Accurate Georegistration of Point Clouds using Geographic Data

Chun-Po Wang Kyle Wilson Noah Snavely
Cornell University
{cpwang, wilsonkl, snavely}@cs.cornell.edu

Abstract

The Internet contains a wealth of rich geographic information about our world, including 3D models, street maps, and many other data sources. This information is potentially useful for computer vision applications, such as scene understanding for outdoor Internet photos. However, leveraging this data for vision applications requires precisely aligning input photographs, taken from the wild, within a geographic coordinate frame, by estimating the position, orientation, and focal length. To address this problem, we propose a system for aligning 3D structure-from-motion point clouds, produced from Internet imagery, to existing geographic information sources, including Google Street View photos and Google Earth 3D models. We show that our method can produce accurate alignments between these data sources, resulting in the ability to accurately project geographic data into images gathered from the Internet, by "Googling" a depth map for an image using sources such as Google Earth.

1. Introduction

The Internet has become an extremely important source of imagery for computer vision, driving work in many areas, including object recognition and 3D reconstruction. Beyond images, the Internet is also increasingly a source of *detailed geographic information*. By this, we mean accurately georeferenced information about the shape and content of the world, ranging from accurately georegistered imagery from mapping sites (e.g. Google Street View and Microsoft Bing Maps), to 3D city models from Google Earth, to open source vector street maps from OpenStreetMap[1].

In combination with algorithms for estimating the geoposition of an image, this geographic data can be of significant value in computer vision tasks. For instance, if one could compute the exact camera pose (position, orientation, and camera intrinsics) of a photograph in the world, one could imagine simply "Googling" a depth map for that image (or at least the static parts of the scene such as streets and buildings), by rendering the world from that viewpoint using online 3D city models. However, in order to make use of such information, we need a way to determine the geographic pose of an image to near *pixel-level accuracy*. To this end, one could make use of recent work in image localization that accurately registers an image with a 3D structure-from-motion (SfM) point cloud [14, 11]. While this process results in a 3D camera pose, the accuracy of this pose in a geographic coordinate system is limited by how well the point cloud itself is positioned on the Earth's surface.

Our work addresses this problem of accurately and robustly georegistering SfM reconstructions, towards the goal of automatically creating precisely georegistered point clouds for many places around the world. We build each

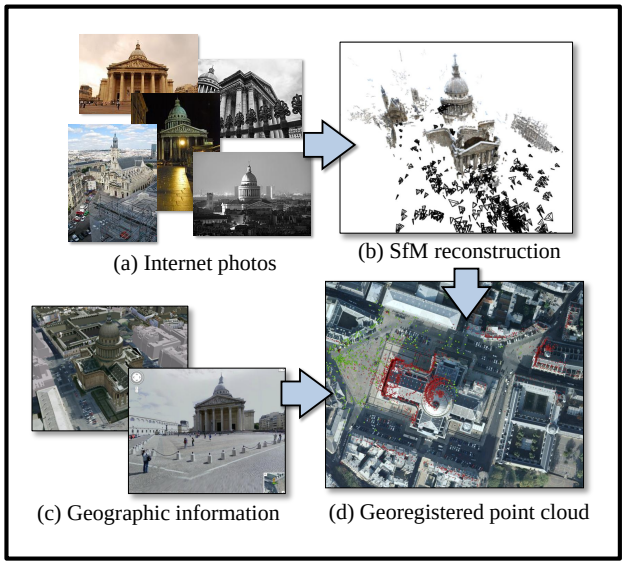


Figure 1. Overview of our work: (a) we take unordered Internet photos of a scene, reconstruct scene geometry using SfM , then (b) align the resulting point cloud with online geographic data, including (c) Street View images and Google Earth 3D models. These georegistered point clouds allow us to project useful information, such as depths of 3D building models, into images (see Figures 5 and 6).

[1] http://www.openstreetmap.org/

SfM model from a large collection of Flickr photos, and georegister it using geographic data itself for alignment. We draw on three sources of geographic information: (1) noisy geotags from the input Flickr photos, (2) nearby Google Street View images with more accurate geotags, and (3) 3D city models from Google Earth. We propose a robust pipeline that leverages this data, along with information estimated from the SfM model such as vanishing points, to produce an accurate georegistration. This process is illustrated in Figure 1.

2. Related work

Location recognition and geographic reasoning. Photo-sharing sites, such as Flickr, have accumulated vast collections of geotagged imagery, and over the past few years such imagery has been used to create databases that enable georegistration of new query images [8, 11]. Of particular note is the `im2gps` work of Hays and Efros [8], which goes further in using this geolocation information to estimate rough properties of a scene, such as how mountainous or populated it is expected to be based on a coarse geolocation. Unfortunately, the geolocation estimates produced by such methods is limited by the accuracy of the geotagged image database. Geotags on Flickr are extremely noisy, and hence only rough attributes can be inferred. In our case, we want to make use of detailed geographic information, such 3D models, and require near pixel-accurate image georegistration.

Other methods use more accurately geotagged Street View images to localize new query imagery [19, 3]. However, these image-to-image matching techniques do not directly produce an actual camera pose for a query image, and so geographic data such as 3D building models cannot be projected into the photo. Moreover, many points of interest lack such accurately calibrated images (including the photos from the top of the Empire State Building shown in Figure 5). In contrast, our framework is built on SfM models created from collections of Internet photos and georegistered, enabling us to compute explicit camera pose for input photos matched to such models.

In graphics, Kopf *et al*. use 3D city models for photo editing tasks, such as dehazing, relighting, and image annotation [10]. However, they require the pose of an image to be manually specified; our system is completely automated.

Alignment of point clouds to maps. As part of our work, we align SfM reconstructions using georeferenced images and 3D models. In related work, Kaminsky *et al*. georegister SfM models to an aerial image by aligning 3D points to image edges [9]. Similarly, several researchers have sought to align laser scans to aerial images [5, 6]. Strecha *et al*. use geotags and 2D building footprints to help merge multiple small 3D models together [15]. Others have aligned SfM data to digital surface models (DSMs) or street maps, from either aerial input photos [13], or input images with accurate GPS information [18]. Taneja *et al*. align 360 panoramic streetview images to cadastral 3D models (akin to the Google Earth 3D models considered in our work) [17]. The combination of data sources we draw on differs from prior work; we take SfM reconstructions with noisy GPS tags and register them to 3D georeferenced models. While we find that this setting requires robust alignment methods, we also found it to be more accurate than alignment to 2D aerial images.

3. Accurate georegistration of SfM models

This section describes our algorithm for registering a set of input images of a scene to a georeferenced coordinate system, using geographic data from online mapping services.

3.1. Inputs and overview

Our input is an unstructured collection of Internet photos of a scene (e.g., the Empire State Building or Trafalgar Square), some of which may be geotagged, found using Internet search (we gather images from Flickr in our work). As our algorithm runs, it also searches for nearby geographic data sources, in particular Google Street View (GSV) images and Google Earth (GE) 3D building models, operating differently depending on which sources of data are available for the location of interest.

We first run feature matching and structure from motion (SfM) on the input images, using the implementation of [1]. SfM estimates the camera pose of a subset of the images, and reconstructs a set of 3D points. This reconstruction is only known up to a similarity transformation; the scale, 3D rotation, and 3D translation of the model with respect to the Earth's surface are unknown. Our algorithm georegisters this SfM model, estimating the seven unknown degrees of freedom that position it on the globe (Figure 2). We achieve this in several steps: first, we robustly compute the gravity vector for the scene using analysis of vanishing points (VPs). Second, we estimate a rough position, heading, and scale using available geotags for the inputs photos themselves. Next, we query Google Maps for nearby GSV images; if enough are available, we add these to our SfM model, and re-estimate its geo-alignment using these more accurately geotagged images. If GSV images are unavailable, we use a VP and scale matching algorithm to align the SfM model with Google Earth 3D data. Finally, we run an ICP-like process to refine the alignment to the Google Earth data.

Vanishing points. Our method first extracts vanishing points (VPs) from the SfM model to facilitate the estimation of scene orientation. We have found VPs to be more robust than other oriented objects on SfM datasets, such as surface normals or planes; such robustness is key to georegistering a noisy SfM model.

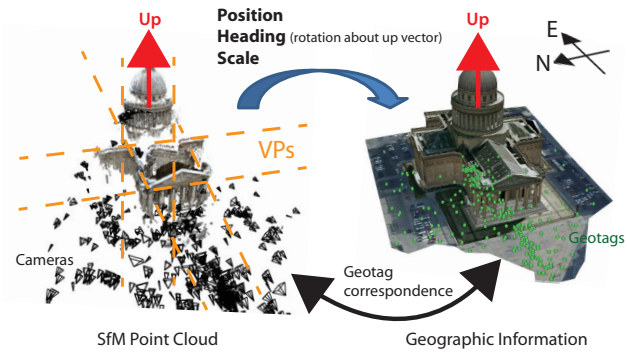


Figure 2. Overview of georegistration process: first the up vector of the SfM model is determined, then the rough position/heading/scale can be found by aligning cameras (black pyramids) with their geotags (green dots). More precise information (e.g., Google Street View images) can then be used to refine the alignment. The VPs of the SfM model are used to find the up vector and dominant orientations.

To find the VPs, we detect long, linear edges in each image, and accumulate votes for VP directions using a Hough transform on the space of 2D (θ, ϕ) (elevation, azimuthal) angles. These votes account for the image's estimated camera rotation, so that the votes are for vanishing points in the space of global 3D directions in the scene. We assume that long edges are mostly from man-made objects, such as buildings, and often correspond to vertical or horizontal lines in the scene. (Accordingly, our technique primarily works well for urban scenes.) We identify strong peaks in the Hough transform, and save these as our detected VPs, represented as a list of unit vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$, along with their number of votes, w_i.

3.2. Up vector estimation

As a first step in the georegistration, we estimate the scene's global up vector (i.e., the opposite of the gravity vector) by combining a simple analysis of camera orientations with an analysis of vanishing points. In particular, we first use the method of Szeliski [16] to compute an initial up vector $\mathbf{v}_c$. This method fits a plane to the x-axes of each camera's reference frame, and takes the up vector $\mathbf{v}_c$ to be the normal of this plane, based on the observation that most people take photos with only slight roll (rotation about the viewing direction). We observed that the resulting vector is usually close to correct, but can still deviate from true up by several degrees (especially when most cameras are pointed in a similar direction). Instead, we pick the pre-computed VP direction $\mathbf{v}_i$ that is in best agreement with $\mathbf{v}_c$ (i.e., which maximizes $|\mathbf{v}_i \cdot \mathbf{v}_c|$). We deem this vector to be the vertical VP, and rotate the scene accordingly. We found this combined method to be quite precise in determining which way is up. Other VPs $\mathbf{v}_j$ that are nearly orthogonal to $\mathbf{v}_i$ are then identified as potential horizontal VPs for later steps of our pipeline.

3.3. Georegistration using geotags

The up vector fixes two degrees of freedom of the orientation; what remains are the heading, 3D position, and scale. We now compute an initial estimate for these parameters using available geotags.

Georeferenced photos. Many online photos are now embedded with GPS coordinates, either from hardware or through manual tagging. With these, we can pose the georegistration problem as one of absolute orientation, in which we seek a heading, 2D translation, and scale (i.e., a 4-DoF similarity transformation) that best fits the SfM camera positions to their geotags. (Note that estimating the elevation of the scene is usually ill-posed with geotags alone, since they often lack altitude data.) We use RANSAC to find the 4-DoF transformation T_0 with the most inlier camera positions.

Unfortunately, geotags on Flickr photos are quite noisy, often off by tens or hundreds of meters. We found that for scenes covering a large area, accurate results are possible using geotags alone, but when cameras are quite close together—e.g., on the viewing platform of the Empire State Building—the resulting heading and scale estimates can be quite inaccurate (Figure 4).

Street View photos. To address this problem, we turn to Google Street View (GSV) for panoramas with accurate GPS coordinates distributed around cities. 4.) Given our initial geo-alignment T_0, we find and download nearby GSV panoramas, along with their GPS coordinates. Each panorama is then sampled into seven individual perspective images and registered into our SfM model [11], and a new 4-DoF similarity transformation T_{GSV} is computed using their high-quality geotags.

We find that using GSV photos produces a more accurate georegistration when they are available. Unfortunately, there are still many places around the world where we cannot find Street View photos. For instance, there are no Street View images available for datasets captured well above ground level (e.g., photos taken on viewing platforms or Ferris Wheels), and ground-level views that do exist near such locations usually fail to match. This leads to our second georegistration method: directly aligning point clouds to georeferenced 3D models.

3.4. Georegistration using georeferenced 3D models

For many cities, Google Earth (GE) has a layer of georegistered 3D buildings and terrain. While GE isn't the only source of 3D data—e.g., some cities have been scanned using LIDAR—GE is attractive because it serves as a centralized

storehouse of world-wide data. These 3D models are an alternative to Street View images as a source of data to which we can align our SfM reconstructions.

To do so, we first download from Google Earth a 3D model of the geometry surrounding our SfM model, in the form of a georegistered triangulated mesh. A standard approach to aligning two models to each other is the Iterated Closest Point (ICP) [2] algorithm. Since ICP is a local optimizer, we expect it to work well if provided with a good initial transformation, e.g., for T_{GSV} computed using GSV as described above. However, when such an initialization is not possible (e.g., due to lack of GSV photos), or is otherwise poor, we need an alternative approach to obtain good initial alignment.

We first assume that the first alignment T_0 is centered at the correct latitude and longitude (i.e., the geotags bring us close to the correct mean camera location in the ground plane). Given T_0, we compute a new set of candidate transformations by separately estimating the heading, height, and scale of the SfM model.

Heading. To obtain heading, we align the detected horizontal vanishing points (VPs) of the SfM model with those computed from the GE mesh. For the GE mesh, we compute dominant horizontal VPs by first finding the subset of mesh triangles that are nearly vertical; their normals tend to be perpendicular to VPs in the SfM model. We have each normal vote for the headings ± 90 degrees of it, and then select dominant directions as VPs for the mesh (Figure 3(a)).

Next, we match SfM and 3D model VPs to find several candidate 1D rotations. The correspondence between VPs is ambiguous, so we vote for possible 1D rotations based on how well the VPs of the two models align under all possible rotation. In particular, we score a candidate rotation as the sum of the product of votes of pairs of VPs within 4°. All unique rotations with scores within a given fraction of the maximum are returned as candidate headings for the SfM model.

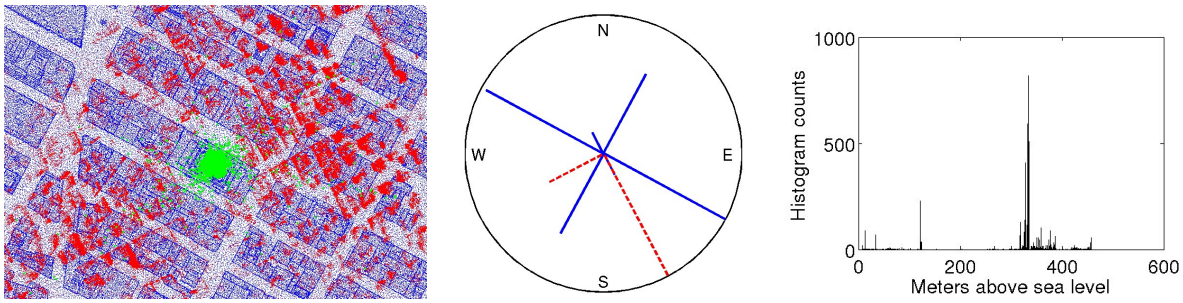
Height. As described above, one usually cannot reliably determine the altitude of a model using geotags alone. In cases such as the Empire State Building viewing platform, there is a difference of several hundred meters between ground and model. We compute a set of candidate heights by observing that on the whole, most images are taken while standing on top of surfaces. Accordingly, for each camera position in our SfM model (under the initial registration T_0), we find the highest point on the GE mesh at that latitude/longitude, forming a histogram of possible heights as shown in Figure 3 (b). Our algorithm selects a set of strong, unique peaks from this histogram, and for each peak generates a candidate transformation that translates the SfM model so that its median camera height matches that elevation. This assumes that photos are not taken from many different elevations at the same location, and that photos were captured from the highest point on the GE surface.

Scale. The scale estimate is most important in cases where cameras are clustered close together; as the radius of a SfM model approaches the level of the geotag error, the uncertainty in scale increases. In challenging situations such as the top of the Empire State Building, we observed that the initial transform T_0 could have a scale off by as much as a factor of two. To estimate the correct scale, we need some way of measuring absolute distance; we propose a method based on distances from cameras to points they observe, in both the SfM and Google Earth coordinate frames. In particular, for each ray from a camera to a visible 3D point in our SfM model, we find the ratio of (1) the length of that ray to (2) the length of the ray from the same camera position to the first object it hits in the GE model (computed by rendering a depth map from the 3D model). From a collection of these ratios, we employ a simple voting scheme and choose the most prominent modes as our candidate scale estimates.

This procedure depends on having a roughly accurate orientation, or else we would be comparing distances between pairs of rays that do not correspond. Hence, the order of computation is as follows: we first compute candidate headings and heights. For each combination of these two, we compute candidate scales. Finally, we compose these transformations, performing rotation and scaling around a median camera, and we use all as initializations to a final ICP stage, selecting the one with the lowest error after ICP as the result.

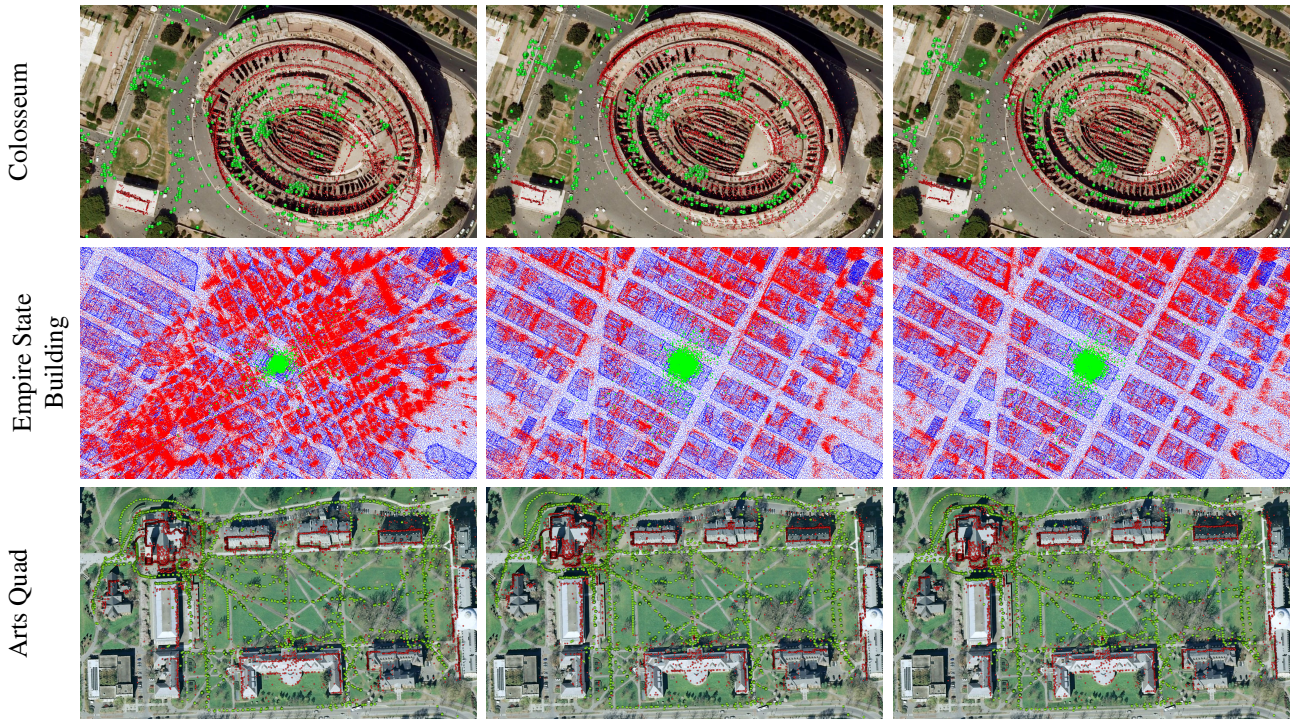
Final ICP registration . We complete the alignment using ICP between the SfM model and the GE mesh with a few modifications to aid in robust convergence. We compute the cost function over only the 70^{th} percentile inliers since the SfM models tend to have a small set of spurious outlier points (and sometimes there are missing buildings in the GE models). Additionally, we constrain the median camera position to be above the ground. Finally, we find that trying to estimate a full rotation all at once can lead to unstable alignments until ICP has gotten close to the optimum, so we begin by optimizing over translation, scale, and rotation about the zenith, and then run a second pass of ICP that estimates a 3-DoF rotation.

Other alignment techniques . If we sample a set of points from the GE model, then the georegistration problem can be reduced to a point cloud alignment task, for which existing more global techniques can be applied. We tried one such recent technique, the automatic point cloud registration method proposed by Makadia *et al*. [12]. This method assumes that point normals are available, and uses these to



(a) (b) (c)

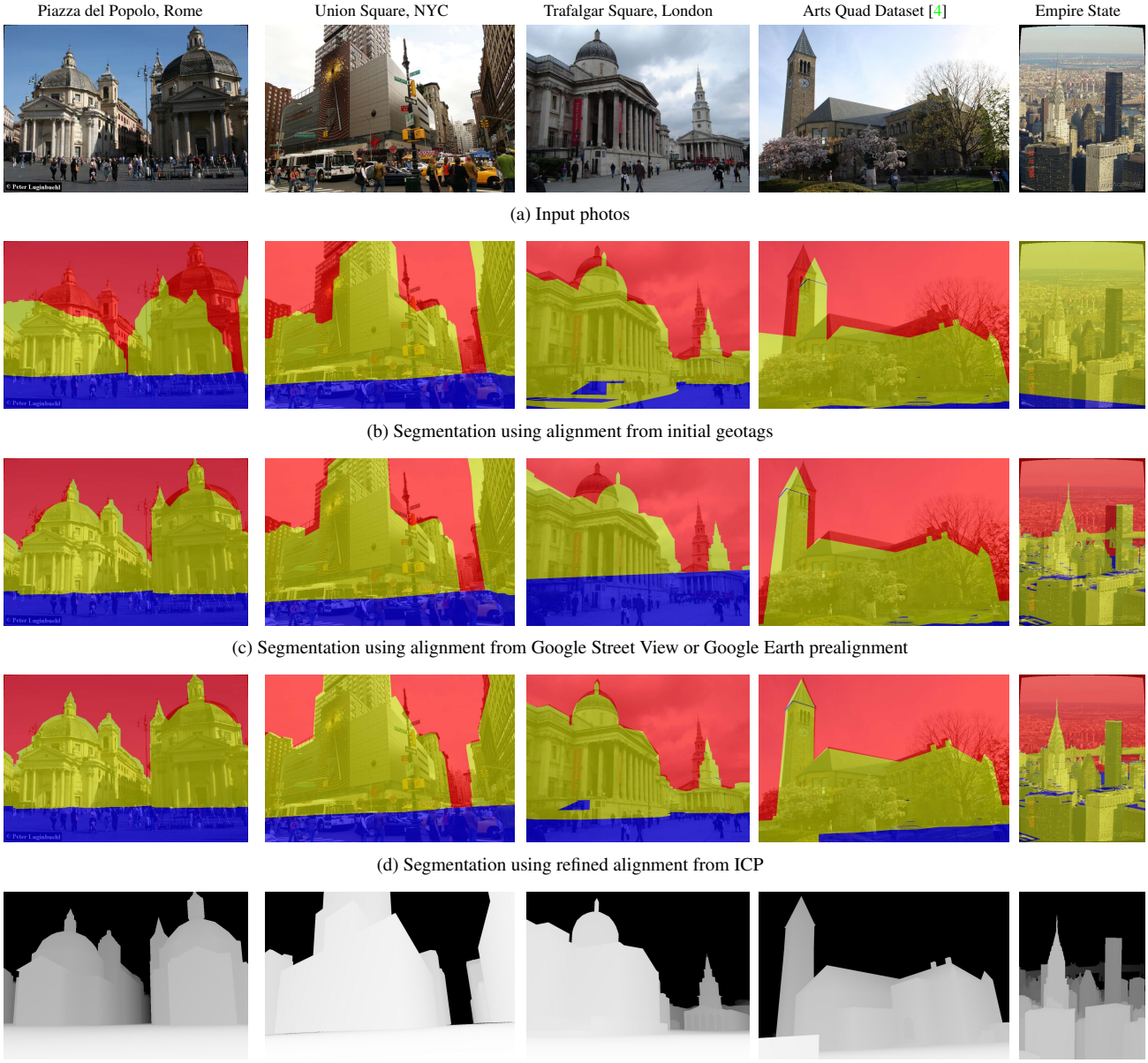
Figure 3. Initial orientation and height alignment for the SfM model from the Empire State Building viewing platform. (a) Initial SfM (red) and Google Earth (blue) point clouds overlaid. Note the incorrect orientation of the SfM model. (b) Detected horizontal VPs for the SfM model (dashed) and GE 3D model (solid). (c) Histogram of estimated heights. Note that the mode of the height histogram corresponds with the actual height of the observation deck, 320 meters.



(a) Georegistered (b) Georegistered using GSV or GE (c) Refined with ICP

Figure 4. From top to bottom: alignment of the Colosseum, the top of the Empire State Building, and the Arts Quad SfM datasets. In column (b), the Colosseum and the Arts Quad were aligned using Google Street View (GSV) imagery. Since this data is unavailable for the top of the Empire State Building, we align based on Google Earth (GE) data as described in Section 3.4. Column (c) shows the results after further applying ICP to refine the alignment.

generate an extended Gaussian image (EGI) for each point cloud. These EGIs are then efficiently aligned, providing a good initialization for ICP. We tried two methods to estimate normals for our SfM point clouds: (a) fitting normals to neighborhoods of points, and (b) using multiview stereo to reconstruct dense points with normals [7]. However, the EGI alignment failed on both sets of normals; we suspect this is due to the points in SfM model being too noisy and sparse. (This is particularly true for datasets such as the Empire State Building, where the camera baselines are quite small relative to the distance to the scene.) Our heuristics, on the other hand, use VPs detected from images for rotational alignment, which we found to be more robust than estimating normals from reconstructed 3D points. However, in cases where VPs cannot be estimated, our technique would likely fail as well.



(a) Input photos

(b) Segmentation using alignment from initial geotags

(c) Segmentation using alignment from Google Street View or Google Earth prealignment

(d) Segmentation using refined alignment from ICP

(e) Depth map created from final refined alignment

Figure 5. Geometric information computed for images using aligned 3D models. Each image in the middle three rows is segmented into geometric categories using the Google Earth 3D model projected into the image (sky: red, vertical surface: yellow, ground: blue). (a) Input photographs. (b) Geometric segmentation using baseline georegistration from geotagged images. (c) Geometric segmentation using improved alignment using Google Street View images (or, in the case of the Empire State Building (last column), the Google Earth pre-alignment. (d) Geometric segmentation using final ICP-refined georegistration. Note that the visual accuracy of the alignment improves from (b) to (d). (e) Depth map created automatically by projecting the Google Earth 3D geometry into the image using the final georegistration.

4. Alignment Results

We ran our algorithm on several SfM datasets from different parts of the world including Trafalgar Square (London), the Pantheon (Paris), Union Square (NYC), and Piazza del Popolo (Rome). We first show qualitative performance of our methods by overlaying aligned SfM models on Google Map satellite images or Google Earth point clouds (Figure 4), and by rendering Google Earth models from the estimated point of view of images in the SfM model (Figure 5). Each figure shows a subset of our datasets; more results can be found in the supplemental material. In general, the alignments using Google Street View or Google Earth 3D models

show clear improvement over the initial alignment computed using only georeferenced online photos. While still not perfect, in general the alignments are qualitatively quite good; we discuss sources of error later in this section. More results of overlaying 3D information on georegistered photos are shown in Figure 6.

The Colosseum dataset shown in Figure 4 is the same as that described in the work of Kaminsky *et al*. [9], which uses maps alone to perform alignments. While their technique failed to produce an accurate registration on this dataset due to the complexity of detected edges in the overhead map view, our method produces a more visually accurate alignment. (Please see supplemental material for a qualitative comparison to their result.) Note, however, that we require different inputs (Street View and Google Earth data vs. an overhead image).

Quantitative evaluation. The desired accuracy of geo-alignment is somewhat task specific; if we care about using Google Earth data, we may favor accuracy of alignment to that data, rather than a more absolute measure.

Nonetheless, we would still like an objective error measurement. To this end, we manually aligned the point clouds with Google Map satellite images to obtain approximate groundtruth camera locations. In addition, for the Arts Quad dataset, we have 216 (out of 5,229) photos with precise geotags from a differential GPS. (These geotags are not used by our algorithm.) Using this ground truth, we measured average errors in camera positions for all three alignment methods: using the noisy camera phone geotags (GRP), using Street View images (GSV), and refining the alignment using ICP to Google Earth 3D models (ICP). The results are shown in the following table:

Average error (meter)	GRP	GSV	ICP
Colosseum	14.03	7.81	7.16
Union Square	13.69	12.10	2.06
Piazza del Popolo	34.55	5.62	5.60
Pantheon	20.05	5.62	5.41
Trafalgar Square	9.42	10.81	8.69
Empire State Building	595.20	–	17.93
Rockefeller Center	245.49	–	6.44
Arts Quad	3.41	2.56	1.86

As the table shows, the use of Street View images and Google Earth models improves the results (except in Trafalgar Square dataset where the large number of georeferenced photos already yields a good result). Note that there are no Street View images available for the Empire State Building and Rockefeller Center datasets; the initial alignment for ICP is obtained using the heuristics in Section 3.4.

Limitations. Despite working well on many datasets, our method has several limitations. First, the georegistration process does not alter the SfM model beyond a similarity transform; if the reconstructed model is incorrect, the georegistration will fail. For example, our method cannot correct low-frequency errors due to drift. Second, ICP highly depends on having a good initialization. For places where we have no good geotags (e.g., no Street View images and the geotags are too noisy) and our heuristics for pre-alignment fail to find good hypothesis, ICP can easily get stuck in local minimum. Finally, our method relies on the quality of the available geographic information. The quality of current Google Earth models seems highly variable; some cities, such as Paris, have well-georegistered models, while others, such as London, seem much more haphazard. However, we believe that high-quality models will only increase in availability over time.

5. Conclusion

To make full use of the rich geographic information that is becoming available through the Internet in vision, accurate geolocation of imagery is critical. Such geographic data can be used both as the means and an end: worldwide datasets, such as Google Street View and Google Earth 3D buildings, are valuable resources for accurately locating SfM models in the world. Street View images and online 3D models. And at the same time, we believe that accurately georegistered datasets can yield new, geography-aware applications in computer vision that leverage geographic data. As future work, we plan to explore ways that additional forms of such information, such as vector street maps, can be used as powerful priors for scene understanding.

Acknowledgements This work was supported by the NSF under grants IIS-1149393 and IIS-1111534, and by Intel Corporation, Google, and Microsoft. We thank Kevin Matzen with his assistance with the experiments in this paper.

References

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building Rome in a day. In *ICCV*, 2009. 2

[2] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *PAMI*, pages 239–254, 1992. 4

[3] D. M. Chen, G. Baatz, K. Köser, S. S. Tsai, R. Vedantham, T. Pylvänäinen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. City-scale landmark identification on mobile devices. In *CVPR*, 2011. 2

[4] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *CVPR*, 2011. 6, 8

[5] M. Ding, K. Lyngbaek, and A. Zakhor. Automatic registration of aerial imagery with untextured 3d lidar models. In *CVPR*, 2008. 2

[6] C. Früh and A. Zakhor. An automated method for large-scale, ground-based city model acquisition. *IJCV*, 60(1):5–24, 2004. 2

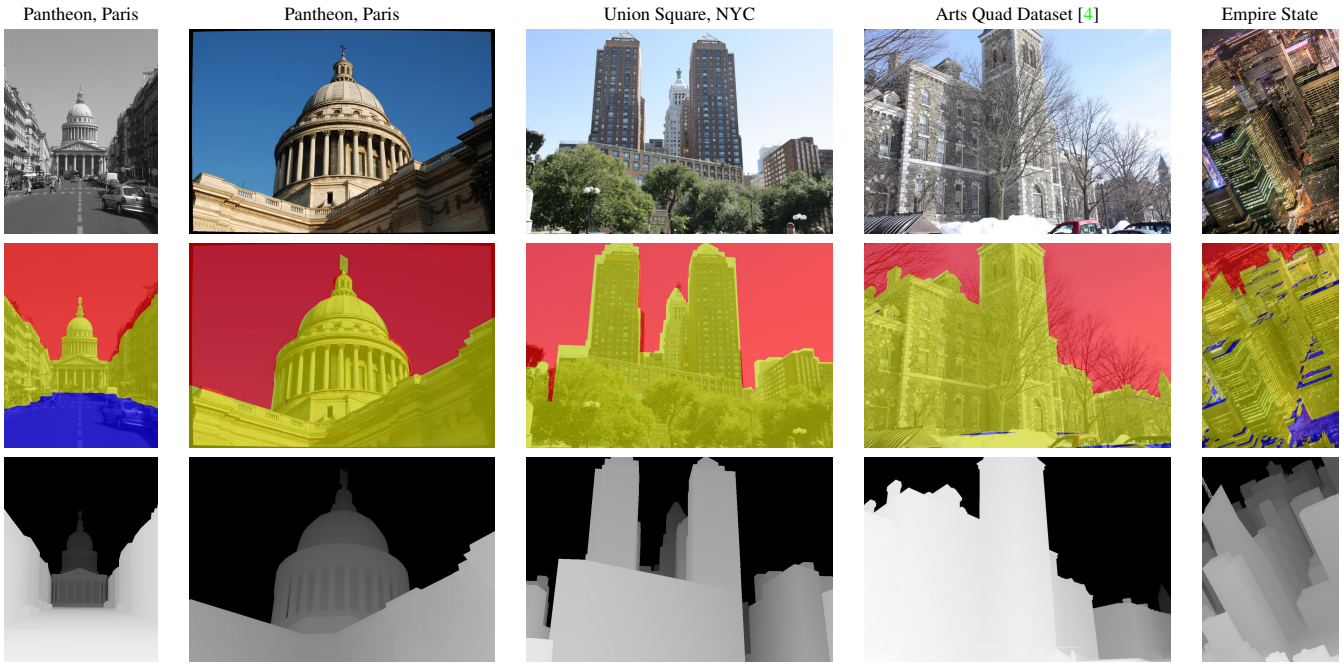


Figure 6. More results of projecting geographic information onto images. First row: input photographs. Middle row: photos with overlaid segmentation from 3D model (sky: red, vertical surface: yellow, ground: blue). Bottom row: depth map automatically created from rendering Google Earth 3D data.

[7] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *CVPR*, 2007. 5

[8] J. Hays and A. A. Efros. IM2GPS: estimating geographic information from a single image. In *CVPR*, 2008. 2

[9] R. S. Kaminsky, N. Snavely, S. M. Seitz, and R. Szeliski. Alignment of 3d point clouds to overhead images. In *Proc. CVPR Workshop on Internet Vision*, 2009. 2, 7

[10] J. Kopf, B. Neubert, B. Chen, M. Cohen, D. Cohen-Or, O. Deussen, M. Uyttendaele, and D. Lischinski. Deep photo: model-based photograph enhancement and viewing. In *SIGGRAPH Asia*, 2008. 2

[11] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide pose estimation using 3D point clouds. In *ECCV*, 2012. 1, 2, 3

[12] A. Makadia, A. I. Patterson, and K. Daniilidis. Fully automatic registration of 3d point clouds. In *CVPR*, 2006. 4

[13] M. Maurer, M. Rumpler, A. Wendel, C. Hoppe, A. Irschara, and H. Bischof. Geo-referenced 3d reconstruction: Fusing public geographic data and aerial imagery. In *Int. Conf. on Robotics and Automation*, 2012. 2

[14] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2D-to-3D matching. In *ICCV*, 2011. 1

[15] C. Strecha, T. Pylvänäinen, and P. Fua. Dynamic and scalable large scale image reconstruction. In *CVPR*, 2010. 2

[16] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Computer Vision*, 2(1), December 2006. 3

[17] A. Taneja, L. Ballan, and M. Pollefeys. Registration of spherical panoramic images with cadastral 3d models. In *Proc. 3DIMPVT*, pages 479–486. 2

[18] A. Wendel, A. Irschara, and H. Bischof. Automatic alignment of 3d reconstructions using a digital surface model. In *Proc. CVPR Workshop on Aerial Video Processing*, 2011. 2

[19] A. R. Zamir and M. Shah. Accurate image localization based on Google maps street view. In *ECCV*, 2010. 2