Web-Accessible Geographic Integration and Calibration of Webcams

AUSTIN ABRAMS and ROBERT PLESS, Washington University in St. Louis

A global network of webcams offers unique viewpoints from tens of thousands of locations. Understanding the geographic context of this imagery is vital in using these cameras for quantitative environmental monitoring or surveillance applications. We derive robust geo-calibration constraints that allow users to geo-register static or pan-tilt-zoom cameras by specifying a few corresponding points, and describe our web interface suitable for novices. We discuss design decisions that support our scalable, publicly-accessible web service that allows webcam textures to be displayed live on 3D geographic models. Finally, we demonstrate several multimedia applications for geocalibrated cameras.

Categories and Subject Descriptors: I.4.1 [**Image Processing and Computer Vision**]: Digitization and Image Capture—*imaging geometry*; I.4.9 [**Image Processing and Computer Vision**] Applications; H.2.8 [**Information Systems**]: Database Applications—*spatial databases and GIS*

General Terms: Algorithms

Additional Key Words and Phrases: Camera calibration, geospatial web services, participatory GIS, pan-tilt-zoom cameras, social computing, voluntary geographic information, webcams

ACM Reference Format:

Abrams, A. and Pless, R. 2012.Web-Accessible Geographic Integration and Calibration of Webcams. ACM Trans. Multimedia Comput. Commun. Appl. 0, 0, Article 0 (June 2012), 20 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Currently, the global network of webcams offers recent, unique viewpoints from tens of thousands of locations all over the globe. In the past, viewing webcam imagery with respect to its surrounding geographic context has yielded promising results, including large-scale environmental monitoring and estimating local weather conditions. Due to their widespread use and up-to-date imagery, webcams are emerging as useful resources for large-scale ecological studies and surveillance, especially within the context of geographic information systems (GIS). A recent non-exhaustive count has found URLs for more than 17 000 webcams all over the globe, and although they provide useful visual information, these cameras are largely unlabeled [Jacobs et al. 2008; Jacobs et al. 2009]. In this article, we describe our efforts to attach geospatial labels to these images in order to support further studies in using webcam images as worldwide sensors.

The world wide web has an increasingly large, diverse, and interesting set of sensors that provide live updates. While many projects have worked to integrate this sensing data within GIS (and to provide frameworks for querying and visualizing this data), they typically represent streaming video sources as "just another sensor". However, to interpret camera data automatically, it is often vital to know not just where the camera is, but also its orientation and camera zoom. Thus, this paper seeks to address this problem within the context of organizing all publicly-available webcams so that they can be coherently used as visual sensors.

In this article, we also introduce novel camera geocalibration methods which uncover the imaging geometry with respect to the camera's geospatial context. Geocalibration parameters (such as the camera's latitude, longitude, altitude, orientation, and zoom level) are important for understanding how the camera fits into the environment, and camera calibration is often a necessary first step toward more in-depth computer vision applications.

There are several challenges in effectively calibrating all cameras. Fully automated systems for geocalibrating and geo-orienting cameras [Jacobs et al. 2007; Jacobs et al. 2008; Sunkavalli et al. 2008; Lalonde et al. 2008] report geolocation accuracy to the scale of miles, and orientation accuracy to a scale of degrees, which is insufficient to map pixel data onto 3D models. Furthermore, current geo-orientation algorithms assume that the orientation is fixed throughout a long-term time lapse sequence of images, which may be invalid, especially when interpreting popular pan-tilt-zoom cameras. Also, webcams are emplaced by a large collection of different organizations, from private citizens to national parks to businesses of all sizes, and most cameras do not publish accurate calibration information. Even when a camera offers geolocation estimates, they usually reference objects in the scene rather than the camera itself, and to our knowledge, there does not exist a service which offers an estimate of camera altitude. Thus, we believe that in the foreseeable future, human assistance will be necessary to provide reliable geocalibration estimates.

We therefore seek to create tools which make it simple for arbitrary users on the web to contribute to calibrating all the world's cameras. We create a web application allowing anyone to calibrate cameras by specifying a few corresponding points, and to create a web based visualization infrastructure that uses the user's web browser to embed live imagery into GIS. This eliminates the need for any central organization to be responsible for computationally-expensive image manipulation, and makes the system scalable to the large number of cameras by not requiring a central server to touch the live video data.

When using input generated from arbitrary Internet users, we found that a small percentage of their input was arbitrarily wrong (e.g. placing a correspondence many miles away from the ground truth location). As we will show, even a single misplaced correspondence can ruin the prospect of a usable geocalibration. Therefore, we introduce a novel ℓ_1-based calibration method that is robust to these outliers. We show that even under a large amount of gross errors, our method recovers the correct geocalibration parameters.

Pan-tilt-zoom (PTZ) cameras, where a user can control the camera remotely through a web interface, are emerging as a popular choice for live imagery. PTZ cameras may benefit the most from providing the surrounding 3D geospatial context, because their orientation and zoom can change both dramatically and rapidly. We discuss methods of solving for the dynamic calibration parameters from a small set of points clicked under different viewing directions and show our interface where a user can remotely move the physical camera and watch a virtual, textured camera frustum move to mimic the camera motion. We show that our calibration method works well, recovering the ground truth camera geolocation and geo-orientation, even in the presence of erroneous input.

We conclude by demonstrating several applications of integrating live webcam imagery into GIS, and how to augment these applications once the geocalibration of a camera is known. We discuss methods to embed live and historic imagery into 3D GIS, assisting users with calibration-specific constraints, visualizing live PTZ cameras under varying rotation and zoom levels, and remotely controlling PTZ cameras through a geographic interface.

2. RELATED WORK

Creating systems to support the organization, querying, and data integration from distributed sensors with known geolocation has been a major focus of work over the last decade. Classic works include IrisNet [Gibbons et al. 2003] and SenseWeb [Nath 2006; Kansal et al. 2007], which offer frameworks to organize, collect, filter, and combine sensor feeds, to enable distributed queries with reasonable response times. However, these architectures leave a centralized data store responsible for collecting data, which makes it difficult for them to deal with large sets of video feeds.

There have been several other efforts integrating video into a 3D context. Most recently, the authors of [Kim et al. 2009] describe methods to add dynamic information into virtual earth environments, by displaying multiple video streams in a single context. They provide compelling results by merging several video streams into a view-dependent texture that minimizes artifacts of viewing a back-projected texture from extreme angles.

They also augment the virtual environment through object recognition results (e.g., animating a virtual pedestrian whose position is determined by a pedestrian detector) and in-depth image processing (e.g., estimating cloud cover and updating the virtual cloud models). As a first step to these high-level algorithms, they first calibrate their camera with respect to the surrounding geographic environment. In this article, we solve similar geometric computer vision problems, but with the goal of making the calibration process accessible to novice users through a scalable web interface.

Video Flashlights [Sawhney et al. 2002] provided the first instance of projecting textures onto a geo-referenced space, which places several cameras from a network in a single 3D context, which [Sebe et al. 2003] extended by implementing a multi-camera tracking system for surveillance purposes, integrated within a geospatial context. [Gliet et al. 2008] create a GIS-mashup that maps weather data onto the sky part of a webcam image (but ignores the part of the scene below the horizon).

In [Sankaranarayanan and Davis 2008], the authors demonstrate a registration framework for a network of PTZ cameras. They use a spherical ‘fisheye’ panorama to register a PTZ camera to a common geographic space through an affine transformation, and use this geographic space to coordinate the network of cameras and track objects across the different cameras in the network. Although we use a different calibration approach, the motivating ideas and problem statements are similar. In this paper, we show a PTZ registration and calibration framework for a single camera that relates the parameters of a movable camera in terms of familiar camera models from computer vision. Furthermore, we extend the work of Sankaranarayanan and Davis by incorporating geographic altitude and the effects of zooming the camera on the resulting image.

In [Kopf et al. 2008], the authors use an image annotated with the geolocation and orientation of the camera, as well as the 3D geographic coordinates of every pixel in the scene to dehaze, relight, and annotate images. In this paper, we simplify the interface necessary to complete this type of registration and consider applications to live imagery.

This article is an extension of two papers published in peer-reviewed conference proceedings [Abrams et al. 2010; Abrams and Pless 2010].

3. A SCALABLE SYSTEM FOR PARTICIPATORY WEBCAM GEO-INTEGRATION

While the geometric transformations involved in the mapping of imagery onto 3D geometry are largely well-understood [Hartley and Zisserman 2000], there remain important system architecture issues

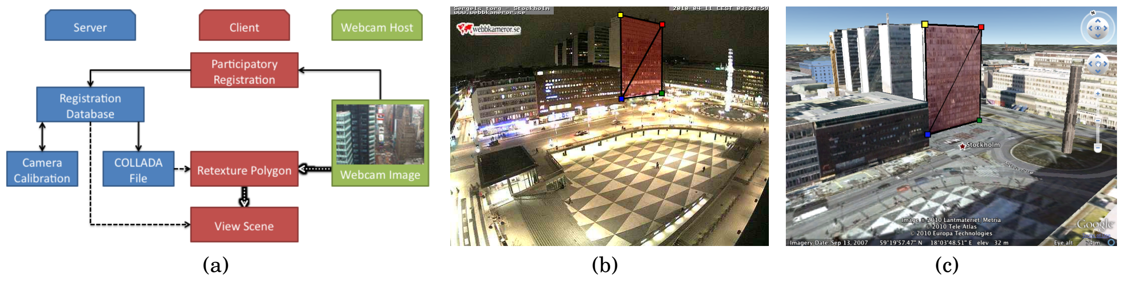


Fig. 1. (a) A data flow diagram for our system. Each solid line is followed once per registration, each dotted line is followed once per scene view, and the thick dotted line is followed many times during a scene view. To register an image to Google Earth geometry, the user drags markers across a 2D webcam image (b), and then creates a correspondence between the 2D image and the 3D geographic space by dragging 3D markers to match the 2D points (c).

in building a system which scales to a large number of simultaneous users and accepts untrained volunteer input.

In this section, we describe our developments in implementing a web application for users to embed live imagery taken from webcam images into 3D virtual environments. Here, we discuss the constraints and difficulties of building this system as a web application, and how we resolved these issues to create a publicly-accessible scalable web service.

Creating a publicly-accessible system to map image data onto 3D geometry is not scalable because the central server would require both significant computational and network resources. The streaming image data must come to the central server, then each image must be warped, and the resulting warped imagery must be sent on to the client.

Our system architecture supports 2 modes of interaction: when a user is calibrating a system and when a user is viewing a system.

The first mode of interaction is the registration stage, where a user marks corresponding points between the 3D world and the 2D image. Towards the integration of webcams into a 3D geographic space, we require a user to find a webcam and click on a small set of correspondences between the image and the 3D GIS data (in our work, we use the Google Earth browser Plug-in). When the user is done marking up a scene, we are left with a set of correspondences between the geographic world (i.e., a latitude, longitude, and altitude) and the image. Figure 1(b)-(c) describes this interface.

The second mode is more challenging, as users may view a camera many times and many users may be viewing the scene simultaneously. If all data were to go through a central server, that would become both a bandwidth and computation bottleneck. In our system, during the view of a live stream, the image data only goes to the client computer and all image manipulation is performed on the client side. Furthermore, this is implemented with JavaScript so that the client computers only needs a web browser. This allows a single central server to provide this service to a very large number of potential clients. Figure 1 shows the data flow diagram illustrating interactions between the user and the system for both the registration process and how a user views a scene.

3.1 Additional Information

As of the time of this writing, our system has been active for over two years, and users from all over the globe (mostly in the United States, Germany, and Romania) have contributed a total of 203 scenes, calibrating 119 cameras from 3129 correspondences. The interface can be demoed at http://www.projectlive3d.com. This website allows a user to register and calibrate both still imagery and PTZ cameras, download scenes in KMZ (the format used by the desktop Google Earth application), download calibration information for each camera, use code to embed a calibration visualization widget into another website, and view the source code running the system.

4. GEOCALIBRATION METHODS

These correspondences between a 3D scene and 2D image form constraints on the imaging geometry that took place when the image was captured. These constraints form the basis for camera calibration, a well-known geometric vision problem which aims to discover how the 3D scene projects onto the image. By solving for the camera calibration, we discover the camera's position, orientation, focal length, and more. In this section, we discuss our methods for camera calibration, particularly with the goal of applying geographic reference to the calibration parameters.

We first describe the classical method to solve the calibration problem, and then extend it in a few ways. We provide a robust variant of camera calibration that can handle arbitrarily-wrong corruptions in the data, which may come from inexperienced or malicious users, and then describe a novel PTZ calibration model that accounts for variations in orientation and zoom.

4.1 Background

In solving for the 3D camera calibration we find the 3×4 projection matrix M that defines a virtual linear camera such that if a world point (X, Y, Z) in 3D projects to some image coordinate (x, y), then

$$\begin{bmatrix} xw \\ yw \\ w \end{bmatrix} = M \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad M = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \\ m_9 & m_{10} & m_{11} & 1 \end{bmatrix} \tag{1}$$

where w is a homogeneous scaling factor. Although only 6 correspondences are required to solve for M, we require users to submit at least 12. Given these correspondences, we then find the optimal M that minimizes the following least-squares problem:

$$M^* = \arg\min_M \left\| \begin{bmatrix} xw \\ yw \\ w \end{bmatrix} - M \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \right\|_2^2 . \tag{2}$$

The above equation can be solved by the following equivalent system of linear equations:

$$M^* = \arg\min_M \left\| \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -X_1x_1 & -Y_1x_1 & -Z_1x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -X_1y_1 & -Y_1y_1 & -Z_1y_1 \\ & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -X_nx_n & -Y_nx_n & -Z_nx_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -X_ny_n & -Y_ny_n & -Z_ny_n \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_{11} \end{bmatrix} - \begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix} \right\|_2^2 \tag{3}$$

$$M^* = \arg\min_M |A \operatorname{vec}(M) - b|_2^2. \tag{4}$$

We then solve for Equation 4 using standard linear least squares.

Then, M can be further decomposed into:

$$M = K[R|T] = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} . & . & . & T_x \\ . & R & . & T_y \\ . & . & . & T_z \end{array} \right] \tag{5}$$

where K contains the intrinsic parameters of the camera, R is the 3×3 camera rotation matrix, and T is the 3-element camera translation vector. Here, f_x and f_y refer to the camera's focal lengths, x_0 and y_0 give the image coordinates of the optical axis (usually in the center of the image), and s is the skew of the camera. Once M is known, we solve for K and R by decomposing the left-most 3×3 submatrix of M using the QR decomposition, which decomposes a matrix into an upper-triangular matrix and a rotation matrix. Finally, we solve for T as:

$$T = K^{-1} \begin{bmatrix} m_4 \\ m_8 \\ 1 \end{bmatrix} \tag{6}$$

Given this calibration we solve for the position and orientation of the camera in geographic coordinates. Then we find the center of the camera in world coordinates, given by $C = -R^{\top}T$. We then convert C back to geographic coordinates (latitude, longitude, and altitude), and express the rotation matrix R in terms of heading, tilt, and roll.

Once a scene is calibrated, we allow the user to move the virtual camera to match the parameters of the actual camera. Some example calibrated scenes are shown in Figure 2.

This linear camera model is a commonly-used approximation of how many real-world cameras take images. However, it does not model barrel or pincushion distortion, which is present in some cameras. This simple model could therefore lead to inaccurate correspondences in some cases. We opted to keep the simpler camera model because it provides an effective approximation to a real camera, while keeping the required number of user-submitted correspondences to a minimum. This has not been a significant source of error.

As an implementation detail, we first convert the user's latitude, longitude, and altitude into a Cartesian coordinate system by representing each point as meters east, north, and up of some arbitrarily-chosen origin point, and this new coordinate is passed into the calibration procedure as (X, Y, Z). This assumes that the ground is locally linear around the origin point, and since the altitudes of the cameras we work with are well above the altitudes where we might experience distortions due to the curvature of the Earth, this is a safe assumption in practice.

4.2 Robust Geocalibration

In informal observation, we notice that novice users are typically experienced at dragging 2D points across an image, but generating high-quality 3D corresponding points remains a challenge (see Section 5.3 for our observations and attempts to alleviate these difficulties). Furthermore, the building geometry provided within Google Earth is itself provided by arbitrary Internet users. Although submissions to Google Earth are monitored, there are still a large quantity of buildings in Google Earth that do not accurately portray their real-life counterparts. Therefore, the set of correspondences generated with the web interface may be of relatively poor quality.

In response to this issue, we propose a calibration procedure that is robust in the presence of these kinds of outliers. The system of linear equations shown in Equation 1 can be solved using linear least squares, which is done in previous work [Abrams et al. 2010; Abrams and Pless 2010]. Although least squares is a popular tool for many data- fitting problems, it is notoriously fickle in the presence of outliers. A small percentage of outlying points can corrupt the entire calibration routine, even if the rest of the correspondences are accurate. This brittleness emerges from the squared error; any outlier will accrue a large error that dominates the rest of the optimization, and least squares solutions typically fit solutions that accrue smaller errors everywhere, rather than large errors at a few sparse locations.

A common solution is to apply RANSAC, which attempts to isolate inliers and outliers during the fitting step. However, these methods work best when the number of data points far exceed the number

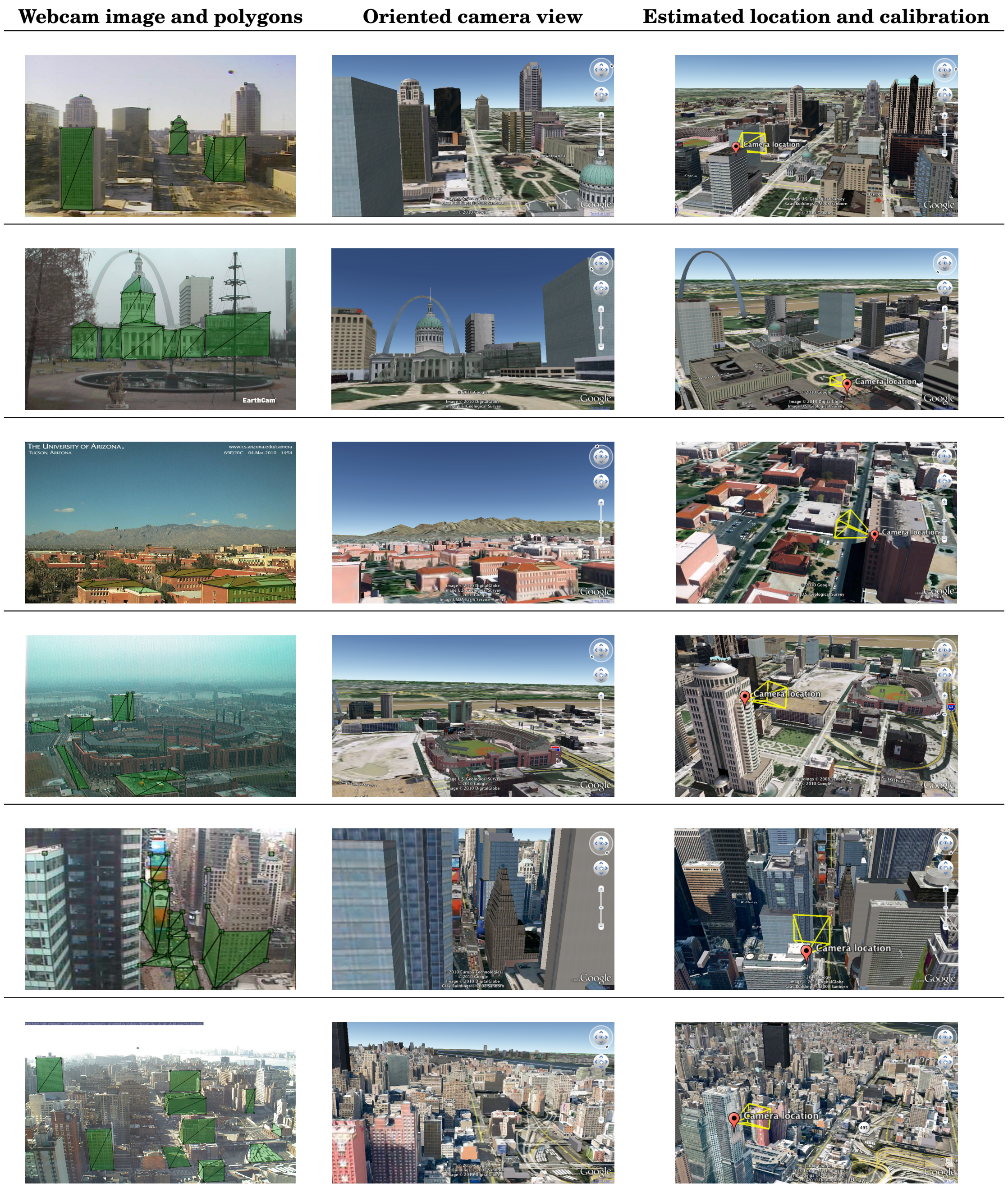

Webcam image and polygons
Oriented camera view
Estimated location and calibration
Camera location
EarthCam
The University of Arizona
Tucson, Arizona
Google

Fig. 2. A selection of webcams and the polygons users have added to scenes (left column), the view from the positioned and oriented camera in Google Earth (center column), and the calculated location and orientation of the camera in the context of the scene (right column). The views in the center column have not been altered from the default calibrated camera view.

of points required for any fitting step (in our case, six). An ideal solution would take advantage of all (potentially noisy) data points to minimize the constraints on how many points a user must submit.

The ℓ_1 norm, or the sum of absolute values, is a robust estimator when we expect some portion of the data to be corrupt. Because the absolute error measures the magnitude of the error rather than the magnitude of the *squared* error, the resulting errors from outliers are on the same order of magnitude as the inliers. Therefore, ℓ_1 methods support solutions that give large errors to a few data points, so long as the errors themselves are sparse. Furthermore, in contrast to RANSAC, we can incorporate all data points, yet maintain robustness properties.

We incorporate the ℓ_1 penalty into the optimization in Equation 4:

$$M^* = \arg\min_M |A\,\mathrm{vec}(M) - b|_{\ell_1}. \tag{7}$$

By denoting A_i and b_i the ith row of A and b, we define $t_i = |A_i\,\mathrm{vec}(M) - b_i|$. Then, the optimization Equation 7 is equivalent to the following constrained optimization:

$$\begin{aligned} \arg\min_{M,t} \quad & \sum_{i=1}^{n} t_i \\ \text{s. t.} \quad & -t_i \le A_i\,\mathrm{vec}(M) - b_i \le t_i \end{aligned} \tag{8}$$

Finally, we express Equation 8 as a canonical linear program:

$$\begin{aligned} \arg\min_{M,t} \quad & \begin{bmatrix} \mathbf{0}_{11} \\ \mathbf{1}_{2n} \end{bmatrix}^\top \begin{bmatrix} \mathrm{vec}(M) \\ t \end{bmatrix} \\ \text{s. t.} \quad & \begin{bmatrix} A & -\mathbf{I}_{2n} \\ -A & -\mathbf{I}_{2n} \end{bmatrix} \begin{bmatrix} \mathrm{vec}(M) \\ t \end{bmatrix} \le \begin{bmatrix} b \\ -b \end{bmatrix} \end{aligned} \tag{9}$$

Where $\mathbf{0}_k$, $\mathbf{1}_k$, and $\mathbf{I}_k$ define the k-element 0 vector, 1 vector, and $k \times k$ identity matrix, respectively. This formulation is then solved using standard linear programming algorithms.

4.3 Geointegration of Pan-Tilt-Zoom Cameras

Here, we extend the results in the previous sections to handle calibration of popular pan-tilt-zoom (PTZ) cameras. These cameras are useful for surveillance applications, because they can view a large portion of its surrounding geographic region, but zoom in to focus on regions of interest.

With dynamic cameras, geographic context is especially important. Even if a PTZ camera has been placed near easily-recognizable landmarks, arbitrary users can easily rotate and zoom the camera into geographically uninteresting locations. Therefore, understanding the geographic position and orientation of any view in the camera's range is critical in attempting to understand the geometric foundations of the scene.

Each camera used in this work advertises its internal pan, tilt, and zoom values through a web interface in terms of three integers, with upper and lower bounds[1]. The goal of the calibration step is therefore to apply geographic meaning to these numbers, by inferring the geographic orientation and focal lengths from an arbitrary (pan, tilt, zoom) triplet.

Here, we describe a camera model that responds to variations in rotation and focal length, and show how we calibrate such a camera given a set of correspondences. As before in Equation 1, we assume that some 3D point (X_i, Y_i, Z_i) projects onto an image point (x_i, y_i) through a 3×4 matrix M. However,

[1]In particular, we used the LiveApplet series of cameras, although our approach is appropriate to any camera for which the internal state is readily available.

since this matrix is dependent on the camera's rotation and zoom, we now represent this relationship as:

$$\begin{bmatrix} x_i w \\ y_i w \\ w \end{bmatrix} = M(p_i, t_i, z_i) \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}, \tag{10}$$

where $M(p_i, t_i, z_i)$ is a function of the pan, tilt, and zoom of the camera at the time the image was taken. Here, we assume that p_i and t_i are between 0 and 1, and that z_i is between z_{min} and z_{max}, the given bounds. In other words, z_{min} and z_{max} are the zoom values at its largest and smallest focal lengths (fully zoomed in and fully zoomed out), respectively. Then, we can represent $M(p_i, t_i, z_i)$ as a function of the matrices $K(z_i)$ and $R(p_i, t_i)$, the intrinsic camera matrix and the camera's rotation matrix for correspondence i:

$$M_i(p_i, t_i, z_i) = K(z_i)[R_i(p_i, t_i)|T]. \tag{11}$$

Notice that, because the position of the camera never changes, the translation vector T is constant across all images.

Let (x_0, y_0) be the principal point of the image (assumed to be half the width and height of the image), and let the focal length of the camera be a function of zoom, $f(z_i)$. This *focal function* therefore maps otherwise arbitrary zoom values into real-world focal lengths. Then, we estimate the intrinsic matrix as

$$K(z_i) = \begin{bmatrix} f(z_i) & 0 & x_0 \\ 0 & f(z_i) & y_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{12}$$

Previous work [Abrams and Pless 2010] assumed that the focal function was linear and bounded by some unknown minimum and maximum focal lengths. Although the linear model works well for many zoom levels, we noted that there is a dramatic nonlinear change in focal length when the camera is almost fully zoomed in. Measuring the apparent size of several objects in the scene with respect to the zoom parameter suggests that, for the cameras we model, the focal function is an unknown scaling and translation of $f(z) = \frac{1}{z}$:

$$f(z_i) = \frac{\alpha}{z_i - \beta} + \gamma, \tag{13}$$

where $\alpha, \beta,$ and γ are unknowns.

However, due to the singularity when z is close to β, a small change in parameters could have a large change in $f(z)$. Therefore, we instead parameterize this function in terms of the unknown focal lengths f_a, f_b, and f_c at three key zoom levels: $f(z_{min}) = f_a$, $f(z_h) = f_b$, and $f(z_{max}) = f_c$, where $z_h = \frac{z_{min}+z_{max}}{2}$. Representing $f(z_i)$ in terms of these three values is much less sensitive to small changes in the parameter space, and each variable has a well-defined semantic meaning. Given any f_a, f_b, and f_c triplet, we find the corresponding α, β, and γ parameters through the following closed-form solution:

$$d = z_h(f_c - f_a) + z_{max}(f_a - f_b) + z_{min}(f_b - f_c) \tag{14}$$

$$\beta = \frac{z_h z_{max}(f_b - f_c) + z_h z_{min}(f_a - f_b) + z_{max} z_{min}(f_c - f_a)}{-d} \tag{15}$$

$$\gamma = \frac{z_h(f_b f_c - f_a f_b) + z_{max}(f_a f_c - f_b f_c) + z_{min}(f_a f_b - f_a f_c)}{d} \tag{16}$$

$$\alpha = f_b z_h - z_h \gamma - f_b \beta + \beta\gamma \tag{17}$$

These α, β, and γ values then define the focal function as in Equation 13.

Then, we define the rotation of the camera as a function of the current pan p_i and tilt t_i, as well as the rotational offsets p_0, t_0, r_0, which define the default pan, tilt, and roll of the camera in radians (when $p_i = t_i = 0$). Finally, we introduce two unknown scale parameters s_p and s_t which convert the $[0, 1]$ scale into radians (i.e., field of view of the camera over all pan and tilt angles).

$$R(p_i, t_i) = zxz(-(s_p p_i + p_0), s_t t_i + t_0, r_0)$$

where $zxz(u, v, w)$ is the standard 3×3 rotation matrix that rotates u radians around the Z axis, v radians around the X axis, and w radians around the new Z axis. As u increases, the rotation moves counterclockwise around the Z axis. Therefore, we negate the u term in order to move the camera clockwise as p_i increases (i.e., as the camera pans from left to right). This rotation scheme is common among GIS applications.

The last calibration unknown is T as $\begin{bmatrix} T_x & T_y & T_z \end{bmatrix}^\top$, the unknown but constant position of the camera.

For any PTZ camera, the focal function f (and, in our case, the values f_a, f_b, and f_c), the rotational offsets p_0, t_0, r_0, the rotational scales s_p, s_t, and the position of the camera, T_x, T_y, T_z define a set of projective matrices. From this definition of the camera, we can find the projection matrix for any pan, tilt, or zoom value in the camera's range.

Given a geocalibrated camera for which we know the projection matrix M, we solve for the potential points Q that project onto (x, y):

$$\begin{bmatrix} xw - m_4 \\ yw - m_8 \\ w - 1 \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_3 \\ m_5 & m_6 & m_7 \\ m_9 & m_{10} & m_{11} \end{bmatrix} Q \tag{18}$$

where w is an arbitrary homogenous scaling factor.

Then, to compute the error of some assignment of the unknown values f, p_0, t_0, r_0, s_p, s_t, T_x, T_y, T_z, we first compute the projection matrix $M(p_i, t_i, z_i)$ as given by Equation 11, for any correspondence i. Then, we find some ray $\vec{Q}_i$ that goes from the camera center through the ground truth image coordinate (x_i, y_i) as in Equation 18. If the camera is placed correctly, the ray $\vec{Q}_i$ should be exactly the ray $\vec{R}_i$ that passes through the camera center and (X_i, Y_i, Z_i). We normalize both $\vec{Q}_i$ and $\vec{R}_i$ and compute the *angular difference error* for correspondence i as follows:

$$E_i = 1 - \vec{R}_i \cdot \vec{Q}_i. \tag{19}$$

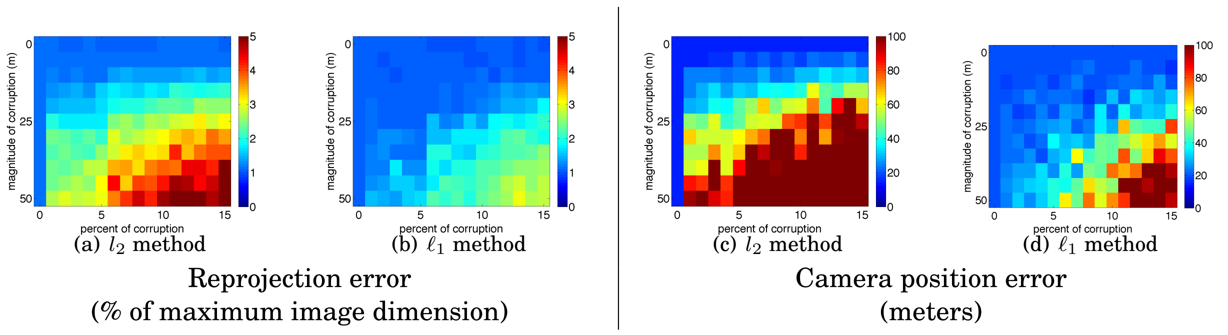
This error measures the angle between the two vectors, and we combine this error for all corresponding points to create the complete cost function, which we minimize over all our parameters:

$$\underset{f, p_0, t_0, r_0, s_p, s_t, T_x, T_y, T_z}{\operatorname{argmin}} \sum_{i=1}^{n} E_i w_i \tag{20}$$

where w_i is a per-correspondence weight that more strictly enforces the angular error term for larger zooms:

$$w_i = \frac{z_i - z_{max}}{z_{min} - z_{max}} + 0.1.$$

Notice that this definition of error depends on $M(p_i, t_i, z_i)$, and hence, every value that contributes to the position, orientation, and focal length of the camera. The total error for some prospective solution is then the sum of all E_i.



(a) l_2 method (b) ℓ_1 method (c) l_2 method (d) ℓ_1 method

Reprojection error (% of maximum image dimension)

Camera position error (meters)

Fig. 3. Experiments to validate the robustness of the proposed calibration method. In each case, the proposed ℓ_1 method is more robust to outliers, both in magnitude and quantity.

Another way to measure the error of some assignment of the unknowns is to instead compute the reprojection error for each point. In our formulation, we found that the reprojection error results in a much more difficult optimization, because the reprojection error can easily cause unbounded error terms which dominate the optimization; if a point projects outside the image frame, it can accrue an arbitrarily large error. However, because we use an angular error, we effectively bound how much error any one point can contribute (i.e., each error term is between 0 and 2).

Given a set of PTZ correspondences (x_i, y_i, X_i, Y_i, Z_i, p_i, t_i, z_i), we solve for the camera's position, default orientation, and focal lengths using the simplex method with the angular difference error. We initialize the position of the camera as the mean of all 3D points, the rotational offsets $p_0 = t_0 = r_0 = 0$, the rotational scales $s_p = 320°$, $s_t = 120°$ (numbers advertised by the camera manufacturers), and the focal parameters as $(f_a, f_b, f_c) = (12.5, 2.5, 1.25)$ times the width of the image.

After calibration, the set of unknowns defines a projective camera matrix M for any PTZ state (p_i, t_i, z_i). In later sections, we show how this formulation allows seamless integration of live webcam imagery from PTZ cameras into geographic information systems.

5. EVALUATION

In this section, we show quantitative and qualitative measures of accuracy for the proposed methods.

5.1 Robustness of ℓ_1 calibration method

Here we report on experiments that describe the sensitivity of the ℓ_1 calibration method to corrupted points. In each of the experiments listed, we corrupt some percentage of the input correspondences and estimate calibration parameters with least squares and the proposed method. For these experiments, we corrupt a correspondence by adding uniform noise to its 3D location.

We first describe a quantitative measure to estimate the sensitivity of calibration methods to larger and more prevalent corruptions. In this section, we explore two different variables of corruption: the corruption magnitude (the amount of uniform noise we apply to the corrupted points' 3D locations, in meters) and the corruption prevalence (the percentage of points which we randomly sample to corrupt). Furthermore, we use two metrics for calibration quality: the reprojection error and camera position error.

The reprojection error measures how well the computed projection deformation conforms to the set of (uncorrupted) input points; for any point (X_i, Y_i, Z_i), we use Equation 1 to determine the location of the projected image point (x'_i, y'_i). The reprojection error is thus the Euclidean distance between

(x'_i, y'_i) and (x_i, y_i), the ground truth input point. Finally, for fair comparison across cameras of differing resolutions, we normalize the reprojection error by the maximum image dimension.

The camera position error measures how far the estimated camera location is from the hand-labeled ground truth camera location, in meters. Although camera position is not the only product of calibration, measuring error with respect to relative position provides a high-level interpretation for calibration quality.

Finally, for any set of corrupted points, we calibrate the camera using the proposed ℓ_1 method and the least-squares method. For any particular corruption magnitude, corruption percent, error metric, and calibration method, we report the median error with respect to twelve scenes across five trials.

Figure 3 shows the results of this experiment. In all cases, the proposed method is more robust to outliers than the least squares method; even under a small amount of corruption, the least squares method returns erroneous results.

Another method to evaluate the quality of a calibration is through qualitative examination of the resulting camera geometry. Each geocalibration corresponds to a full 3D camera frustum. This representation of the camera shows the camera's location, orientation, and focal length (i.e., its zoom level). When a camera is mis-calibrated, the resulting camera frustum is malformed or in some impossible configuration (such as inside a building or underneath the ground). Thus, examining the deformations to this frustum gives a rough estimate of how the various methods react in the presence of outliers.

Figure 4 shows some example calibration results before and after the addition of outliers. Qualitatively, the least squares calibration technique is fairly fickle in the presence of outliers, while ℓ_1 methods give more robust results.

5.2 Evaluation of Pan-Tilt-Zoom Calibration

Here, we describe the quality and robustness of the PTZ calibration model with respect to varying amounts of corruption. Similar to the previous quantitative experiment, we progressively corrupt more and more correspondences by varying magnitudes, and report the average calibration error of the uncorrupted points (in this case, the angular error term used in PTZ optimization) and camera position error. Figure 5 shows our results. Although the PTZ model is not built explicitly for robustness, we found that the proposed method is still robust to large corruptions in the data, a proxy for real-world miscalibrations generated by novice or malicious users. For small amounts of corruption, the PTZ calibration method reliably uncovers the ground truth camera position to within 10 meters, with an average angular error less than 1.3 degrees.

5.3 User Evaluation

Here, we briefly discuss our experiences in working with an arbitrary userbase, some common mistakes, and our efforts in improving our interface to avoid these errors.

If a user has a problem annotating a scene, it is usually due to inexperience in manipulating 3D points with a 2D mouse. Initially, we found that users would ignore the part of the interface dealing with 3D interaction, submitting "correspondences" between the image and the default location of the 3D points. We alleviated this problem by validating that a user had at least clicked on some of the points before submitting their correspondences.

In a similar vein, we found that in previous iterations of our interface, users would become confused as to what the next step was. We introduced an interactive tutorial for our system that explicitly tells a user what steps are required in order to successfully calibrate a camera. In these tutorials, certain areas of the interface are highlighted in order to focus the user's gaze on the elements of interest.

When a user adds points to a brand new camera, the camera position is unknown and so by default, we place the locations of the 3D points across the globe. In the first iteration of the interface, we

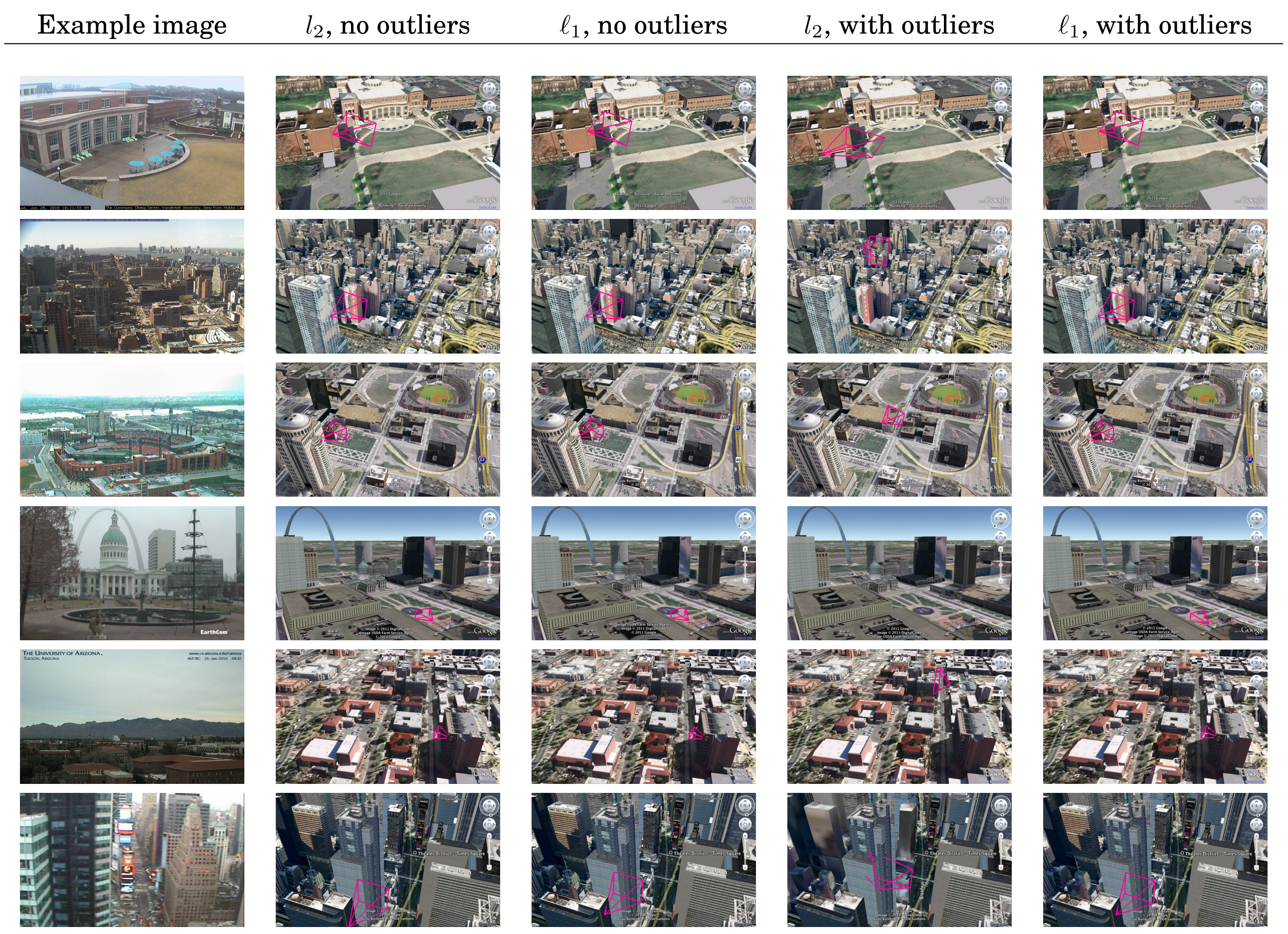


Fig. 4. Each row shows an example image of the camera, and the results from both the least squares approach (l_2) and the robust calibration method (ℓ_1) in the absence and presence of outliers. The proposed algorithm consistently returns more robust calibrations than traditional methods. In the fourth row, the least squares approach placed the camera kilometers away from the ground truth location and is not shown.

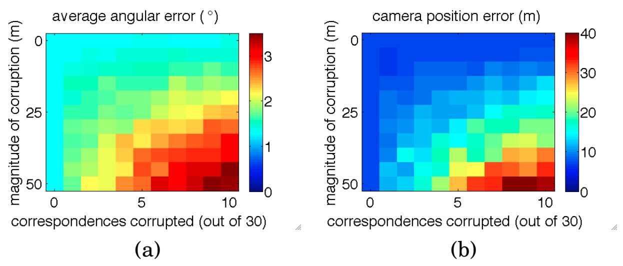


Fig. 5. Quantitative Evalutation of the proposed PTZ calibration algorithm in the presence of large corruptions.

provided a "Center Points" button that would recenter the 3D points on the current virtual camera view, so that once a user had moved the Google Earth camera to the real-world camera's location, he

Scene ID	Total Time (minutes)	Interface Time	Clicks	Correspondences	Final Calibration Error (degrees)
1	49	32m 46s	435	36	1.9
2	47	33m 12s	509	35	8.7
3	39	28m 17s	432	35	3.2
6	21	20m 31s	267	37	1.2
8	25	24m 06s	342	35	0.1

Fig. 6. Statistics of novice users using the interface.

or she could relocate the positions of the 3D points without having to manually drag them from their default location. However, we noted that untrained users were unclear about the purpose of the 3D correspondences, and wouldn't even click the button. We changed the behavior so that the 3D points were always in the center of the virtual camera's view as the user geolocates the camera. This follow-the-user behavior stops when a user moves the first point. After adding this element to the interface, we found that users were more aware of the 3D aspect and had a stronger idea of the goal.

In practice, the user-provided registration process for is approachable to novice users. As an informal test of the ability for users to use our interface, we took a convenience sample of 4 novice users as they registered points to 5 scenes. We gave each user access to the PTZ registration interface, where the camera had been previously geolocated (but the scene was otherwise unlabeled). We gave each user a short tutorial on how to use the interface, and asked each user to submit at least 35 registrations. During this time, we measured several factors: How long the user spent from the end of the tutorial session until all 35 registrations were submitted (total time), how long the user spent during each registration step (interface time), the number of times the user clicked on the interface, the final number of correspondences added to the scene, and final calibration error. Figure 6 shows the results of this informal study.

While monitoring the users, we noticed a few errors. In one test, the user admitted to accidentally creating misregistrations by incorrectly placing several 3D markers on the wrong side of a 3D building, which adversely affected the results of the calibration step. Some scenes have little to no GIS-provided building geometry. In these cases, users sometimes mistakenly registered the roof of a building in 2D to the footprint of the building in 3D. In the cases where the user did not make these mistakes, the resulting calibration quality is good enough to create superimposed images in a geographic setting, as in Figure 7.

6. INTEGRATING IMAGERY INTO GEOGRAPHIC INFORMATION SYSTEMS

In this section, we describe some potential applications that take advantage of geocalibrated cameras.

6.1 Automatic Scene Completion

In order to texture the scene a static camera covers, we require the user to mark off sections of an image and register them to the 3D environment. However, once a camera is accurately calibrated, the transformation that maps the 3D structures and terrain in a scene to an image is known. If the geometry of the scene is also provided (in our case, through the buildings and ground altitude provided by Google Earth), then the mapping for the rest of the scene can be automatically completed.

To automatically generate a scene, we follow these steps.

(1) Generate image annotated with 3D geotags for each pixel.
(2) Triangulate set of georeferenced 3D points.
(3) Remove triangles that cover depth discontinuities.

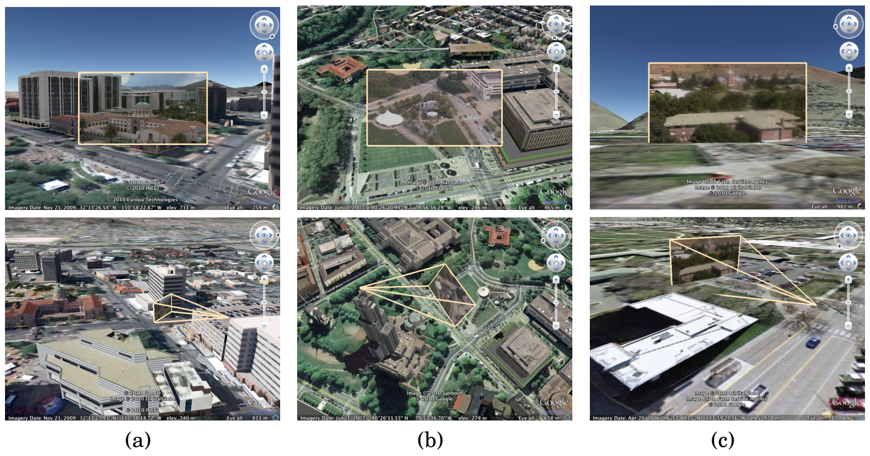


Fig. 7. A series of superimposed camera views and camera frusta generated entirely by correspondences from novice users. The user in (c) had mistakenly selected 3D points on the ground plane, and thus the calibration is incorrect.

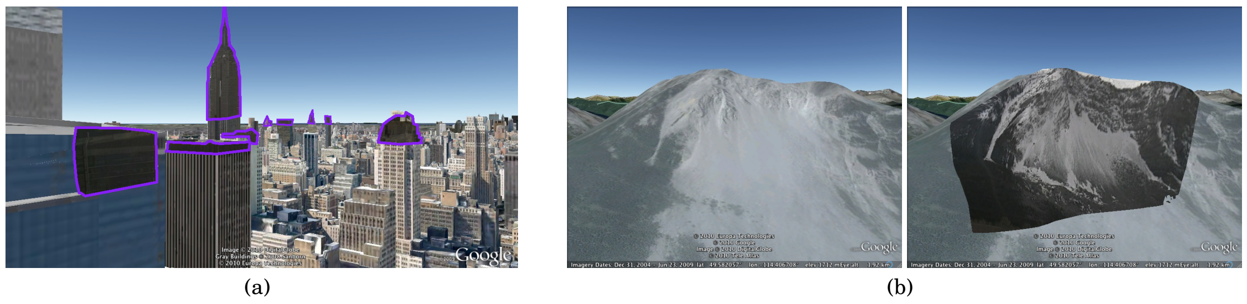


Fig. 8. Some results from automatically-generated scenes. Notice how the depth discontinuities in (a) are preserved, where the mesh is outlined with a violet border. (b) shows the Google Earth view before and after updating the texture with a webcam image from automatically-generated scene. If created by hand, the mesh in (b) would require tens of thousands of perfectly-aligned correspondences.

(4) Texture the mesh using camera calibration.

Given the fully geocalibrated camera for which we know the projection matrix M, we can solve for the potential points Q that project onto (x, y) by Equation 18. From here, we use GIS-provided methods to find the intersection of this ray into the scene's geometry. We repeat this process for every (x, y) to generate a full depth map.

In most images of outdoor scenes, there can be large depth discontinuities between two adjacent pixels, where the geometry of the scene abruptly changes (such as the edge of a building). Although the pixels themselves are very close, the locations of their corresponding 3D points can be drastically different. A simple triangulation of the scene therefore results in triangles with very long edges, where a triangle crosses a depth discontinuity boundary. However, if the background of an image has a cohesive structure (such as a mountain), then it is expected for two adjacent pixels in an image to be somewhat far away, even if there is no depth discontinuity. In other words, the further away an object is, the 3D distance of two adjacent pixels on that object should inherently increase. To prune only the triangles with high depth discontinuity, we score as the length of its longest edge divided by the mean depth of its vertices, and cull away the triangles with large scores (see [Abrams et al. 2010] for details).

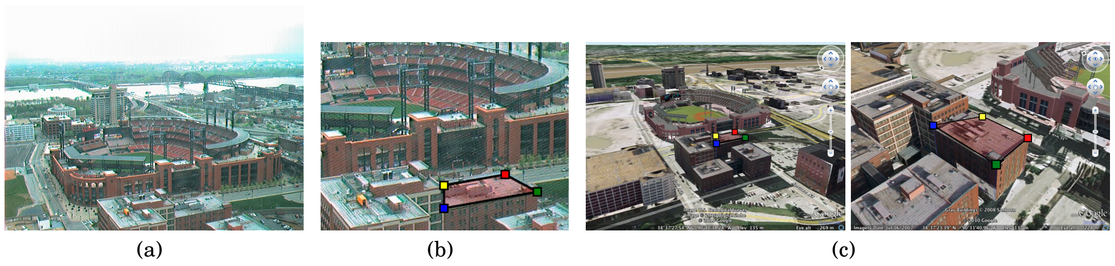


Fig. 9. (a) A webcam image which has been calibrated through the web service. (b) A user-provided selection from the image. (c) Two views of the 3D locations of the selection, determined automatically by the calibration-assisted registration process.

Thus, through a web interface, a user can provide a small set of correspondences (at least twelve) between an image and the geographic scene from which the system will automatically perform camera calibration. Given known scene geometry (provided by GIS applications), this calibration provides the transformation required to retexture a large and complicated scene, such as the skyline in Figure 8(a). In the previous system, users would have to provide a large number of correspondences related to the complexity of the scene geometry. This is particularly useful for scenes in which the geometry of the scene is not planar. For example, Figure 8(b) shows a scene in which the side of a mountain range has been mapped with a live texture. If this scene were to have been created manually, the user would have had to create several thousand correspondences by hand.

6.2 Calibration-assisted 3D point locations

As mentioned earlier, during informal observation of users, we have found that registering points in a 3D geographic space is challenging for novice users, who may be inexperienced with manipulating objects in a 3D environment. While a user may be used to moving 2D objects around the screen (as they must do when specifying correspondences on a webcam image), moving 3D points with 2D mouse controls is challenging.

Once the camera is calibrated, we allow users to take advantage of the geometry in the scene to aid the 3D point manipulation process. As users drag a 2D point around the calibrated webcam image, the corresponding 3D point moves to its most likely 3D location, effectively eliminating user manipulation for the 3D case. This *calibration-assisted* interface solves both the issue of manipulating objects in a 3D environment and creating contextually-significant meshes in the foreground of webcam images.

For any calibrated camera, the relationship between 3D world coordinates and image coordinates is known. Any 2D image point (x, y) in the image has an associated ray in space (defined by Equation 18), so we intersect that ray with the 3D scene geometry and place the corresponding 3D geographic point there. However, this assumes that the initial calibration has been done correctly and the underlying geometry of the scene is consistent with the webcam image. We allow the user to adjust the geolocations of the 3D points if either of those assumptions fail. Figure 9 illustrates how calibration-assisted registration offers an intuitive, accurate alternative for novice users to create live textured meshes of a scene.

6.3 Integrating Historical Imagery

In addition to viewing the current webcam stream in a 3D environment, we allow users to navigate historical webcam archives to view the scene from weeks, months, or years ago. In our work, we use the Archive of Many Outdoor Scenes, a collection of over 100 million images taken periodically from from tens of thousands of publicly available webcams [Jacobs et al. 2007].

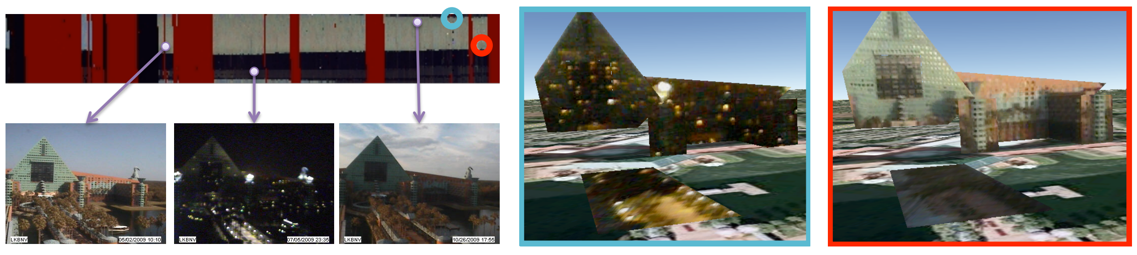

(a) (b) (c)

Fig. 10. Each pixel in the summary image (a) is the average color of one webcam image taken sometime during 2009 (or red when the image could not be downloaded at that time). Vertical changes in the image are due to variations over a single day, while horizontal changes are due to seasonal variations over the year. When the user clicks on the pixels circled in blue and red, the interface updates its textures to (b) and (c), respectively.

To navigate a year's worth of webcam imagery, we use a summary image visualization, which describes variations across a year's worth of archived images [Jacobs et al. 2009]. Here, each pixel in the summary corresponds to a single image taken during that year, and as a user clicks on various parts of the summary image, the image corresponding to the click point is loaded and embedded into the 3D geometry. In this way, a user can quickly navigate historical webcam imagery. Figure 10 describes the historical interface.

6.4 Visualizing PTZ geocalibrations

To texture the scene in the static case (as in Figure 8(b)), it is assumed that the orientation of the camera is known. In the PTZ case, the camera can change its rotation and zoom, which will make any texture coordinates no longer valid. Therefore, any attempt to texture the scene with static texture coordinates will be met with failure when the camera moves, since the texture won't be addressing the same geographic regions across different orientations.

For PTZ scenes, we opted to use a textured rectangular polygon that intersects the camera frustum as the camera rotates and zooms. If the GIS application's virtual camera is placed in the same geolocation as the physical camera, then this gives the effect of the live webcam image retexturing a large field of view in the scene. As the physical camera rotates and zooms, the virtual frustum changes accordingly, matching the webcam image to a geospatial context. A similar approach was demo-ed by [Agüera y Arcas 2010], which superimposes live images from a mobile phone onto street-level maps applications (although their demo and georegistration process has not been released to the public as of this writing). Figure 11 shows an example PTZ camera geointegrated into Google Earth, showing live updates of the estimated view frusta, and the image texture shown in proper geographic context.

6.5 Geographic camera control

The proposed PTZ calibration method discussed takes as input a (pan, tilt, zoom) triplet, advertised by the physical camera, and returns a geocalibration that reflects the camera's current geographic state. However, given a geocalibrated pan-tilt-zoom camera, it's feasible to solve the inverse problem: given some geographic point of interest, provide a (pan,tilt,zoom) state that places that point in the center of the resulting image. Such an interface would be useful for surveillance applications where specifying a geographic point is more intuitive than specifying the pan, tilt, and zoom parameters by hand.

Given some query point of interest (X, Y, Z) and a desired zoom level z, we solve for the camera pan and tilt (p, t) that minimizes the reprojection error from the query point to the center of the resulting

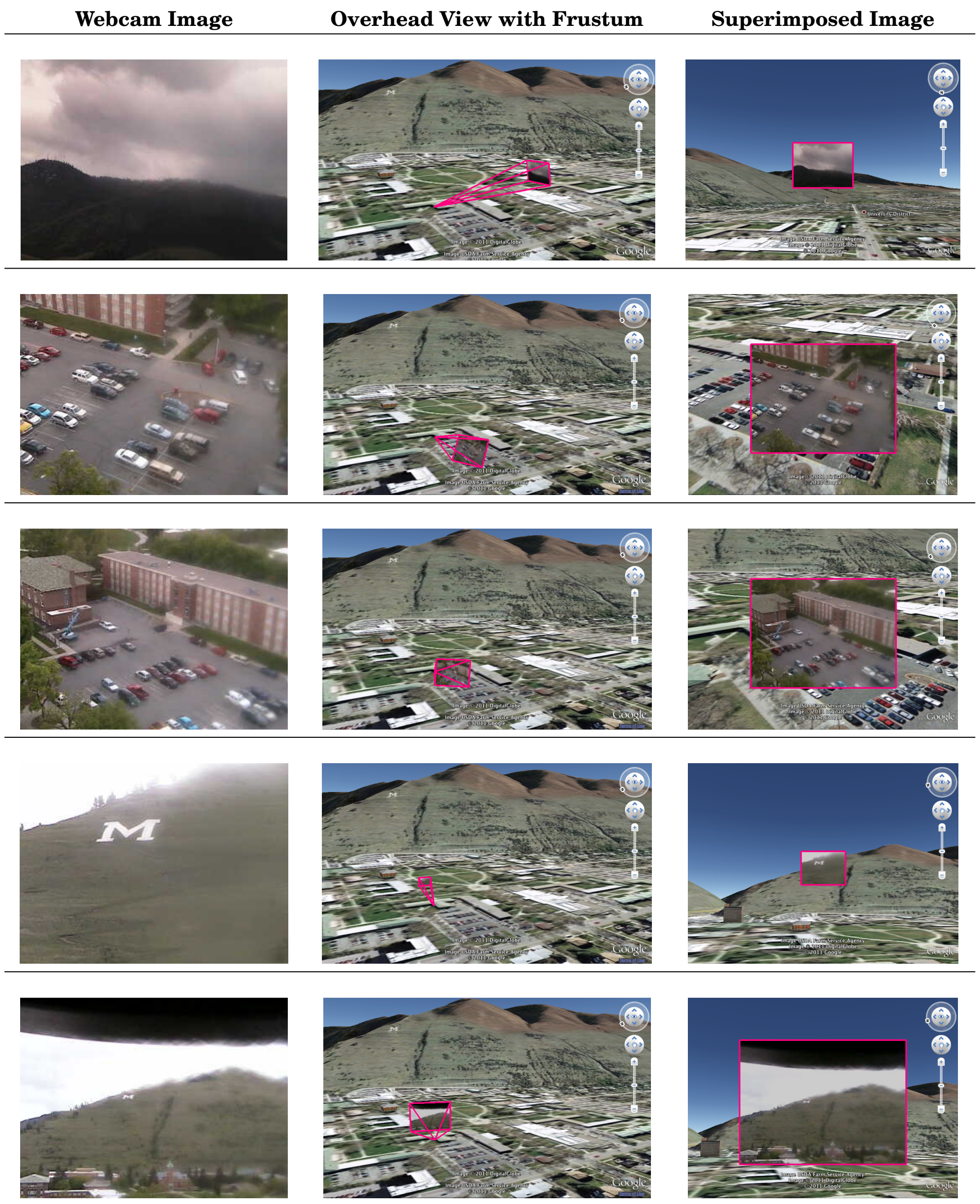


Fig. 11. A series of webcam images taken from a PTZ camera (left column), oriented camera frusta, shown in pink (center column), and the current webcam image superimposed on the virtual camera within Google Earth (right column). Through this interface, the camera frustum and superimposed image change as the physical camera rotates and zooms.

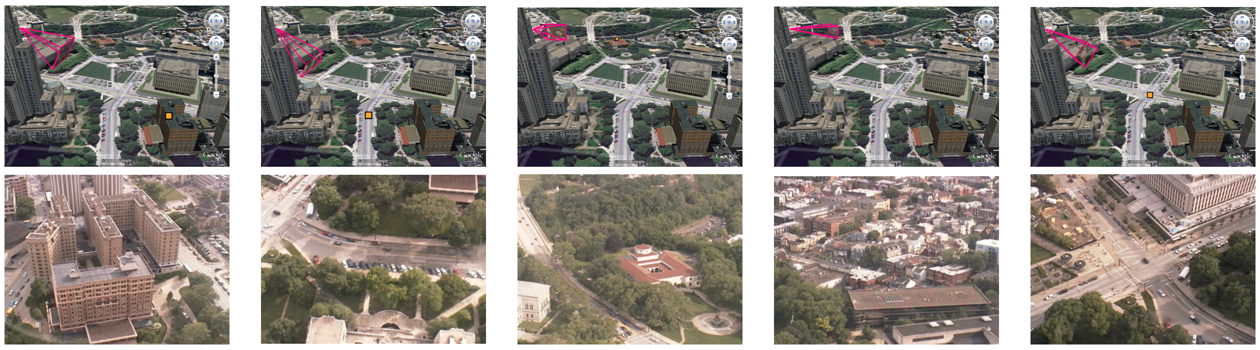


Fig. 12. Examples of a geographically-aware PTZ interface. (Top row) A user drags the orange query point around the virtual 3D environment. We find the optimal pan and tilt values that orients the camera toward the query point and (bottom row) remotely control the physical camera to match those orientation parameters. In each example, the camera is controlled entirely by the manipulation of the orange point in Google Earth.

image. Defining $x(p,t), y(p,t)$ as:

$$\begin{bmatrix} x(p,t)w \\ y(p,t)w \\ w \end{bmatrix} = M(p,t,z) \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{21}$$

we solve for the best pan and tilt as:

$$p^*, t^* = \arg\min_{p,t} (x(p,t) - x_0)^2 + (y(p,t) - y_0)^2, \tag{22}$$

where (x_0, y_0) is the center of the image. Since the number of variables is small and the objective function has a closed form, a simple grid search over p, t is efficient and accurate. Finally, in the same way that we can query for the camera's internal state, we can set the camera's state through a set of web API calls, which remotely controls the physical camera to match the optimal pan and tilt values.

This simple optimization results in an interface where a user can drop a pin that defines an (X, Y, Z) location, and the camera automatically rotates so that the query location is in the frame of the resulting image. So, a user only needs to manipulate the location of a 3D point to change the camera's position. In our interface, we set z to be proportional to distance from the camera center to the (X, Y, Z) point, so that the camera zooms in on distant query points and zooms out on close points. Figure 12 shows an example of the interface.

7. CONCLUSIONS

In this article, we describe our efforts to create a publicly-accessible tool for arbitrary internet users to provide the correspondences necessary to geocalibrate a webcam. By providing geographic context to cameras we interpret their images more deeply and treat them as more than "just another sensor", which has immediate applications to surveillance and environmental monitoring. Because we decentralize our server and push image processing tasks to the end user, we are able to maintain a scalable service with minimal computational overhead. We offer a robust geocalibration method that is robust in the presence of large errors caused by naive users, and a dynamic pan-tilt-zoom calibration method that can handle changes in camera orientation and zoom. These methods work well, even in the presence of large corruptions in the data. Finally, this work supports a variety of tools that take advantage of geographic reference, such as embedding live textures into 3D GIS and controlling a dynamic PTZ camera through geographic constraints.

7.1 Acknowledgements

This work was supported in part by the National Science Foundation (IIS-0546383, EF-1065734, DEB-1053554, IIS-1111398).

REFERENCES

ABRAMS, A., FRIDRICH, N., JACOBS, N., AND PLESS, R. 2010. Participatory integration of live webcams into GIS. In *COM.Geo*. ACM, New York, NY, USA.

ABRAMS, A. AND PLESS, R. 2010. Webcams in context: Web interfaces to create live 3D environments. In *ACM Multimedia*. ACM, New York, NY, USA.

AGÜERA Y ARCAS, B. 2010. Blaise Aguera y Arcas demos augmented-reality maps. *TED Talks*.

GIBBONS, P. B., KARP, B., KE, Y., NATH, S., AND SESHAN, S. 2003. Irisnet: An architecture for a worldwide sensor web. *IEEE Pervasive Computing 02,* 4, 22–33.

GLIET, J., KRÜGER, A., KLEMM, O., AND SCHÖNING, J. 2008. Image geo-mashups: the example of an augmented reality weather camera. In *AVI '08: Proceedings of the working conference on Advanced visual interfaces*. ACM, New York, NY, USA, 287–294.

HARTLEY, R. I. AND ZISSERMAN, A. 2000. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049.

JACOBS, N., BURGIN, W., FRIDRICH, N., ABRAMS, A., MISKELL, K., BRASWELL, B. H., RICHARDSON, A. D., AND PLESS, R. 2009. The global network of outdoor webcams: Properties and applications. In *ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL GIS)*. ACM, New York, NY, USA.

JACOBS, N., BURGIN, W., SPEYER, R., ROSS, D., AND PLESS, R. 2009. Adventures in archiving and using three years of webcam images. In *IEEE CVPR Workshop on Internet Vision*.

JACOBS, N., ROMAN, N., AND PLESS, R. 2007. Consistent temporal variations in many outdoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*.

JACOBS, N., ROMAN, N., AND PLESS, R. 2008. Toward fully automatic geo-location and geo-orientation of static outdoor cameras. In *WACV*. IEEE Computer Society, Washington, DC, USA.

JACOBS, N., SATKIN, S., ROMAN, N., SPEYER, R., AND PLESS, R. 2007. Geolocating static cameras. In *ICCV*. IEEE Computer Society, Washington, DC, USA.

KANSAL, A., NATH, S., LIU, J., AND ZHAO, F. 2007. Senseweb: An infrastructure for shared sensing. *IEEE MultiMedia 14*, 8–13.

KIM, K., OH, S., LEE, J., AND ESSA, I. 2009. Augmenting aerial earth maps with dynamic information. In *IEEE International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, Washington, DC, USA.

KOPF, J., NEUBERT, B., CHEN, B., COHEN, M., COHEN-OR, D., DEUSSEN, O., UYTTENDAELE, M., AND LISCHINSKI, D. 2008. Deep photo: Model-based photograph enhancement and viewing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2008) 27,* 5, 116:1–116:10.

LALONDE, J.-F., NARASIMHAN, S. G., AND EFROS, A. A. 2008. What does the sky tell us about the camera? In *ECCV*. Kluwer Academic Publishers, Hingham, MA, USA.

NATH, S. 2006. Challenges in building a portal for sensors world-wide. In *In First Workshop on WorldSensor-Web, Boulder,CO*. ACM, New York, NY, USA, 3–4.

SANKARANARAYANAN, K. AND DAVIS, J. W. 2008. A fast linear registration framework for multi-camera GIS coordination. In *IEEE International Conference On Advanced Video and Signal Based Surveillance*. IEEE, Washington, DC, USA.

SAWHNEY, H. S., ARPA, A., KUMAR, R., SAMARASEKERA, S., AGGARWAL, M., HSU, S., NISTER, D., AND HANNA, K. 2002. Video flashlights: real time rendering of multiple videos for immersive model visualization. In *Thirteenth Eurographics Workshop on Rendering*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland.

SEBE, I. O., HU, J., YOU, S., AND NEUMANN, U. 2003. 3D video surveillance with augmented virtual environments. In *First ACM SIGMM International Workshop on Video surveillance*. ACM, New York, NY, USA.

SUNKAVALLI, K., ROMEIRO, F., MATUSIK, W., ZICKLER, T., AND PFISTER, H. 2008. What do color changes reveal about an outdoor scene? In *CVPR*. IEEE Computer Society, Washington, DC, USA.