Published: 21 June 2010

Citation in BibTeX format

COM.Geo '10: 1st International Conference on Computing for Geospatial Research and Application
June 21 - 23, 2010
Washington, D.C., USA

RESEARCH-ARTICLE

Participatory integration of live webcams into GIS

AUSTIN D ABRAMS, Washington University in St. Louis, St. Louis, MO, United States

NICK FRIDRICH, Washington University in St. Louis, St. Louis, MO, United States

NATHAN BRADLEY JACOBS, Washington University in St. Louis, St. Louis, MO, United States

ROBERT BRYAN PLESS, Washington University in St. Louis, St. Louis, MO, United States

COM.Geo '10: Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application (June 2010)
https://doi.org/10.1145/1823854.1823867
ISBN: 9781450300315

Participatory Integration of Live Webcams into GIS

Austin Abrams, Nick Fridrich, Nathan Jacobs, and Robert Pless *
Department of Computer Science and Engineering, Washington University in St. Louis

ABSTRACT

Global satellite imagery provides nearly ubiquitous views of the Earth's surface, and the tens of thousands of webcams provide live views from near Earth viewpoints. Combining these into a single application creates live views in the global context, where cars move through intersections, trees sway in the wind, and students walk across campus in real-time. This integration of the camera requires registration, which takes time, effort, and expertise. Here we report on two participatory interfaces that simplify this registration by providing applications which allow anyone to use live webcam streams to create virtual overhead views or to map live texture onto 3D models. We highlight system design issues that affect the scalability of such a service, and offer a case-study of how we overcame these in building a system which is publicly available and integrated with Google Maps and the Google Earth Plug-in. Imagery registered to features in GIS applications can be considered as richly geotagged, and we discuss opportunities for this rich geotagging.

Keywords

Camera calibration, geospatial web services, participatory GIS, social computing, voluntary geographic information, Web 2.0 & GIS, webcams

1. INTRODUCTION

The global network of webcams is a powerful yet underutilized source of imagery. There are at least 17 000 publicly available webcam streams, and it is strongly believed that there are tens of thousands more. Most importantly, webcams have an incredibly high refresh rate. Most webcams update at least every five minutes, and a significant amount of those update at sub-minute or even sub-second intervals [4]. While satellite and aerial imagery has been extensively used in GIS, webcam images have not, in part, because of the challenge in georegistering a large number of webcams. In this paper, we describe an interface that enables untrained Internet users to assist in this registration process.

(a) (b)

Figure 1: Using Live3D, inexperienced users can register a webcam image (a) into Google Earth (b), creating a live view of the scene.

Although a webcam offers a live view of a scene, it can be difficult to put that scene into context without surrounding geographical data. Combining live webcam views with GIS applications contextualizes the camera and offers a secondary view of a scene, providing a more comprehensive understanding of the underlying geometry. Currently, GIS applications like Google Earth combine the views by displaying a geolocated placemark that includes a recent image and additional information about the camera. Although this provides some geographic context, it does not provide the altitude or orientation of the camera, and does not directly tie the imagery to the geometry of the scene.

We create a system that correctly projects live video from webcams onto the geometry provided by a GIS system. This requires matching points between the image and the world. While this matching has been demonstrated before (most recently in [6]), we create a system that is scalable to a large number of users, and does not require extensive experience in camera registration. Using the correspondences provided by the user, we automatically calibrate the camera, which reveals not only the position and orientation of the camera, but also the geolocation of every pixel in the image

*email: {abramsa@cse., fridrich@, jacobsn@cse., pless@cse.}wustl.edu

COM.Geo 2010, June 21-23, 2010 Washington, DC, USA

(except for sky). We consider these geolocated images as richly geotagged, and show some potential applications for rich geotagging.

Our work is a new development in the recent trend towards the use of volunteered geographic information, or VGI. Recent examples, such as OpenStreetMap, whose 200 000 users have contributed hundreds of millions of geo-located points across the globe [3], show that participatory geographic systems can provide a wealth of free, verifiable information. However, by opening these systems to users with presumably little GIS experience, it is possible the user-provided data is incorrect. As noted in [2], measuring and monitoring the integrity of VGI is critical to the success of participatory geographic systems. In this paper, we provide quantitative measures of quality and consistency for the scenes that our user-base creates and we show that, through our system, our users provide high-quality information.

Since its release in January, this user-centric view of camera registration has resulted in the community providing thousands of correspondences for hundreds of scenes, resulting in more than 60 calibrated cameras.

2. RELATED WORK

Previous work has presented the ability to integrate video with geographic models. Hanna et al. [8] demonstrate the Video Flashlight system, which provides an interface to display multiple video streams from a camera network in the context of a 3D reconstruction of the scene. Neumann et al. [9] provide a similar service that focuses on tracking moving objects through the 3D scene. In [6], Essa et al. present an application that projects video streams onto the surface of Google Earth, which takes advantage of multiple overlapping cameras to improve the quality of the back-projected texture.

We expand on these works by providing a service where anyone, even someone with minimal GIS experience, can register a video stream within Google Earth and Google Maps. Although there exist methods to register photos into 3D environments, to our knowledge, these methods are either not accessible or not in a geographic context using GIS-provided geometry. To our knowledge, we provide the first publicly available service of this kind.

3. SCALABLE INTEGRATION OF LIVE VIDEO WITH GIS

The image transformations required to map live webcam imagery onto 2D or 3D geometry are well understood. They require understanding the mapping between pixel coordinates on the live image into geographic coordinates, and using these coordinates to warp the current webcam image onto the geo-referenced geometry. However, existing systems perform the registration, warping and display all locally (on one computer).

Creating a publicly accessible service to do this is not scaleable 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. In this section, we describe a system architecture where the central server allows users to specify corresponding points between the image data and the geographic models, and specifies the necessary image warping. However, during the view of a live stream, the image data only goes to the client computer and all warping is done 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.

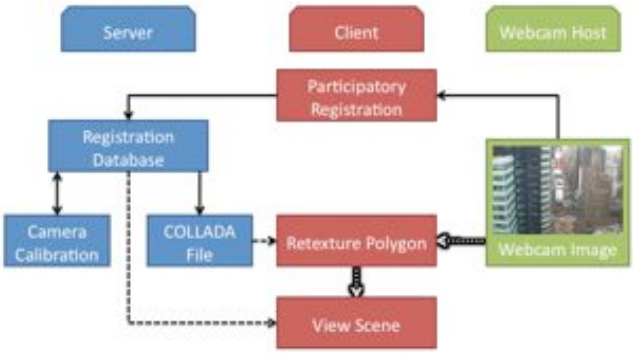


Figure 2: A data flow diagram for our system. Each solid line is followed once per registration, each dotted line is followed once per scene view, and the solid dotted line is followed many times during a scene view.

Figure 2 shows the data flow diagram illustrating interactions between the user and the system for both the registration process and the viewing of the scene. Solid lines represent correspondence data submitted by the user to the database, which, in the 3D case, calibrates the camera and creates a COLLADA file. On loading the viewer for a particular 3D scene, the registration data and COLLADA file are sent to the client, but the image stream flows directly from the image source to the viewer (through the thick dotted line).

In this section we first describe an implementation of this system in 2D, that maps webcam data onto Google Maps, and then an implementation that uses the Google Earth Plug-in to allow users to re-texture 3D building and terrain models with live images.

3.1 Integration of Webcams into 2D Satellite Imagery

The Bird's Eye Viewer (BEV) is a web application that allows users to embed webcam images in Google Maps. Once a user defines the geometric transformation from the image to the satellite view, the webcam image refreshes that area of the map. This is, in effect, a "live satellite", where students walk across campus, cars drive through intersections, and trees sway in the wind, in the context of a satellite image. We chose to use Google Maps because is ubiquitous in the public arena, has low computational requirements and does not require a browser plug-in. Figure 3 shows a typical registration from the BEV.

To register a webcam in 2D, the user finds a webcam and follows these two steps, through the web interface:

1. **Select Image Region in the Webcam.** The user crops away the image above the horizon line.
2. **Define the Geometric Transformation.**

(a)

(b)

Figure 3: A webcam image (a) embedded in the Bird's Eye Viewer (b).

> The user drags image corners on the map so that the warped webcam image matches the satellite view.

As the user drags the pins over the map, the texture continuously warps to match the user's input. Notice that one of the corners in Figure 3(a) is in the middle of a river, and it is not immediately obvious where that point should be on the map. However, since the rest of the image warps quickly enough for interactivity, the user can easily refine the position of this pin until the entire webcam image is aligned.

The 2D transformation that defines the relationship between geographic coordinates and the webcam image is represented as a standard 3×3 homography H:

$$\begin{bmatrix} wu_i \\ wv_i \\ w \end{bmatrix} = H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

where (x_i, y_i) is a point in the image, (u_i, v_i) is its corresponding point on the satellite view, and w is a homogenous scaling factor. Since there are 8 degrees of freedom in H, and each correspondence provides two constraints, the four correspondences at the corners of the image are sufficient.

The BEV also allows for a multi-camera view, by projecting several webcam images onto the same map. In areas with dense camera networks, this offers a comprehensive, live view from many perspectives in a single global context.

Implementation Details.

When developing the BEV, one goal was to perform as much computation on the client side as possible. The BEV warps the texture from the cropped image to the map using a homography through a set of client-side Javascript functions [10]. For any BEV scene, our database keeps track of the horizon line and the location of the 4 pins on the map. As the user views a scene, the client computer directly accesses the webcam image stream, and performs that appropriate perspective mapping (as defined by the homography matrix H). Hence, no image processing is performed server-side; this enables the server to handle many simultaneous users.

Drawbacks.

One disadvantage of the BEV is the limitation of the homographic transform; this simple transformation cannot handle any non-planar geometry in the image. If someone uses the BEV to display a scene with vertical walls, those walls may appear stretched over a large portion of the image. This is shown in Figure 3(b), where the building on the right side of the image is warped incorrectly over the surface of the satellite image.

3.2 Integration of Webcams into 3D Imagery

To overcome the limitations of the homographic transform, we also developed Live3D: a web application similar to the AMOS Bird's Eye Viewer, using Google Earth (and accessible through a browser using the Google Earth Plug-in). Live3D allows live textures to augment the static buildings and satellite images found in Google Earth. Although the geometry of a real-world building rarely changes, its textures are very dynamic. Static textures cannot account for the way real world building faces respond to variations in weather, temporary advertisements, and dynamic lighting.

We present an interface for a user to select some portion of an image and the corresponding geolocation on Google Earth, and a method to create a polygon representing the user's input. Once a user has selected a webcam, the user registers the camera through the web interface as follows (as shown in Figure 4):

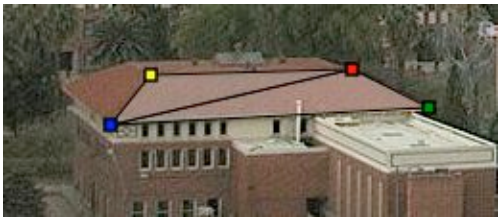
1. **Image Registration.** The user drags a quadrilateral, triangle, or single point around the 2D webcam image.
2. **World Registration.** The user drags the 3D polygon in Google Earth to match the 2D image.
3. **Repeat.** Continue defining corresponding polygons until the camera is calibrated or the relevant parts of the scene are mapped.

Since most of the scenes in the Live3D library come from webcam images, this creates a scene where the ground, the rooftops, and the walls of any object in the scene are updated with textures from a webcam in real-time.

Camera Calibration.

When a user provides at least 12 correspondences through Live3D, the server estimates the calibration of the camera. An accurate camera calibration gives geolocation, orientation, and field of view of the camera. Figure 6 shows some calibrated scenes in the Live3D library.

The user correspondences create matches between 2D image coordinates and 3D geographic coordinates expressed in terms of latitude, longitude and altitude. We use the 3D coordinate of the first correspondence as an arbitrary origin of a 3D orthonormal coordinate system, and change our set of 3D points from (latitude, longitude, altitude) coordinates



(a)

(b)

(c)

Figure 4: To show a webcam stream in 3D, the user defines a polygon on a section of the image (a) and the corresponding surface in Google Earth (b). Then, Live3D uses the specified image region to texture the 3D polygon (c).

into (X, Y, Z) Cartesian space. Here, X is the distance in meters from the origin in the east-west direction, Y is the distance in meters from the origin in the north-south direction, and Z is the altitude in meters above sea level.

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

$$\begin{bmatrix} xw \\ yw \\ w \end{bmatrix} = M \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, M = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}$$

where w is a homogeneous scaling factor. With 6 or more correspondences, we use linear least-squares to solve directly for M. Although only 6 correspondences are required to solve for M, we require Live3D to have 12. If M is not over-constrained, then small users errors will greatly reduce the accuracy of the calibration. Then, M can be further decomposed into:

$$M = K[R|T] = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} [R|T]$$

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 image center, 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:

$$T = K^{-1} \begin{bmatrix} m_{14} \\ m_{24} \\ m_{34} \end{bmatrix}$$

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. Anecdotally, we note that most calibrated cameras in the Live3D library are within a few meters of the true location, and a few degrees away from the true orientation.

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. Section 4 discusses other reasons why a camera calibration might give erroneous results.

Integrating Historical Imagery.

In addition to live images, we take advantage of historical imagery through the Archive of Many Outdoor Scenes (AMOS). AMOS is a publicly accessible archive of webcam images, which has been capturing images from over 1 000 cameras every half hour since March 2006 [5]. If the submitted image URL is a camera in the AMOS database, we provide an interface to navigate the 4-year archive of historical imagery through Google Earth and retexture the polygons the user has built with older imagery. The AMOS website provides a set of summary images that describe the set of images from one webcam taken in one year. As users click on this summary image, user-created polygons in the scene are retextured with the corresponding webcam image. Figure 5 describes how the textures in Live3D update as a user clicks around the summary image.

Implementation Details.

Google Earth requires that all textured polygons come from COLLADA files, and so the server creates a file that defines the positional and texture coordinates for every polygon the user creates. When the user views a scene, these files are loaded and the webcam texture is embedded in 3D.

Recall that the textures on these polygons need to refresh quickly to portray a live 3D model. However, repeatedly serving the same polygon file is problematic. From a user standpoint, when a polygon is refreshed, the old polygon first disappears until the file is reloaded. Then, as the webcam textures are being loaded, the polygon is given a default gray texture. Once the texture is finally loaded, it is not long before it blinks out again to be reloaded. This is problematic from the server side as well. Each time the scene is refreshed, the server has to send all polygons in the scene.

When developing the viewer, we took advantage of a "feature" present in the Google Earth framework. Once one polygon refreshed, it refreshed the textures of all other polygons in the scene. So, when the user chooses to view a scene, the server sends out all of the polygons to view. Then, we serve an additional "invisible" polygon with the same texture as the rest of the polygons. As we refresh this invisible

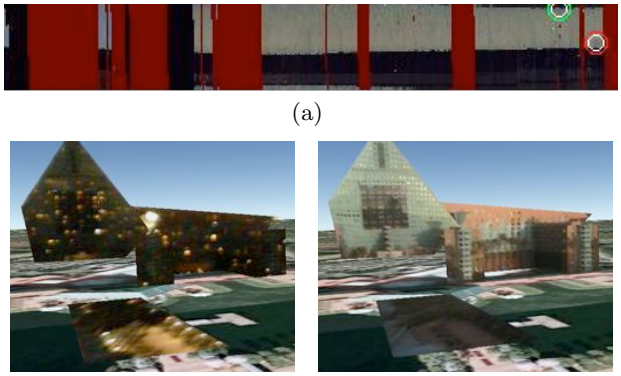


Figure 5: Each pixel in the summary image (a) is the average color of one webcam image taken sometime during 2009 (or red when the image could not be downloaded at that time). Vertical changes in the image are due to variations over a single day, while horizontal changes are due to seasonal variations over the year. When the user clicks on the pixels circled in green and red, Live3D updates its textures to (b) and (c), respectively.

polygon, all the textures in the scene update without disappearing or being re-served. Since we are not hosting the publicly available webcam image, this design choice lets us allow the user to view a live scene, with minimal additional server load and network bandwidth.

4. USER STUDY

To our knowledge, we are the first to offer an openly available geographic camera registration service. This presents us the opportunity to evaluate how an arbitrary user with presumably little registration experience can perform this task. We present a quantitative metric to assess the accuracy of the resulting registration. To date, users have contributed 2273 correspondences to 232 scenes.

4.1 Pixel Projection Error

The camera calibration process solves for the transformation from world coordinates to image coordinates. Under most conditions, this transformation is over-constrained; there are more correspondences than are necessary to solve for the transformation. We find the transformation M that minimizes the least-squared error, and so a natural measure is the *average pixel projection error*. Once we solve for M, we compute the pixel projection error ϵ_i for some correspondence $(x_i, y_i) \rightarrow (X_i, Y_i, Z_i)$ and the expected pixel projection error E_p in a scene of n correspondences as follows:

$$\begin{bmatrix} x_i'w \\ y_i'w \\ w \end{bmatrix} = M \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

$$\epsilon_i = \sqrt{(x_i' - x_i)^2 + (y_i' - y_i)^2}$$

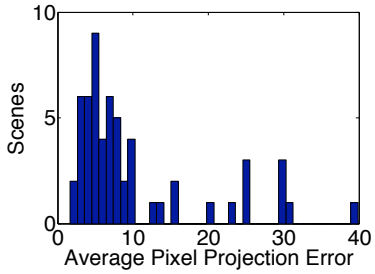


Figure 7: The expected pixel projection error per scene. Approximately one-third of all calibrated scenes have a pixel projection error greater than 40 pixels, and are absent on this plot.

$$E_p = \frac{1}{n} \sum_{i=1}^{n} \epsilon_i$$

As E_p goes to zero, the closer the set of correspondences agree with the camera model, and thus, the better the user did on calibration. Figure 7 shows the projection error of scenes in Live3D. Pixel projection error can come from a variety of sources. A user can simply mis-register some part of the image. The underlying Google Earth geometry could be inaccurate, or the camera could be suffering from barrel or pincushion distortion, not considered in the linear camera model. Figure 8 shows the expected pixel projection error before and after a mislabeled correspondence is removed. We found a median expected pixel projection error of 13.6 among all scenes (where the mean image resolution is above 640×480).

Figure 9 shows that scenes with more user-contributed correspondences give lower pixel projection error. This may be because a single error contributes less to the average error, or because the scenes with accurate point correspondences are easier and this encourages users to add more correspondences.

4.2 User Observations

The Live3D application shares challenges with all systems that use 2D mouse controls to specify 3D points, with the added constraint that we attempt to be accessible to novice users. Our initial experience with users has led us to include:

- A short video tutorial on system use.
- Accepting submissions only where users had changed the default positions for both the image and 3D points.

As of the time of this writing, Live3D has been active for three months, and users from all over the globe (mostly in the United States, Hungary, and Sweden) have contributed a total of 232 scenes, calibrating 77 webcams. Of these, 44 present a feasible calibration. Infeasible calibrations usually come from users who do not understand the interface or its intended purpose, sometimes due to a language barrier.

5. DISCUSSION

5.1 Rich Geotagging

Geographic location is a powerful contextual cue for interpreting images. Our work to register imagery allows for a much deeper connection than just a standard latitude and

Webcam image and polygons	Oriented camera view	Estimated location and calibration

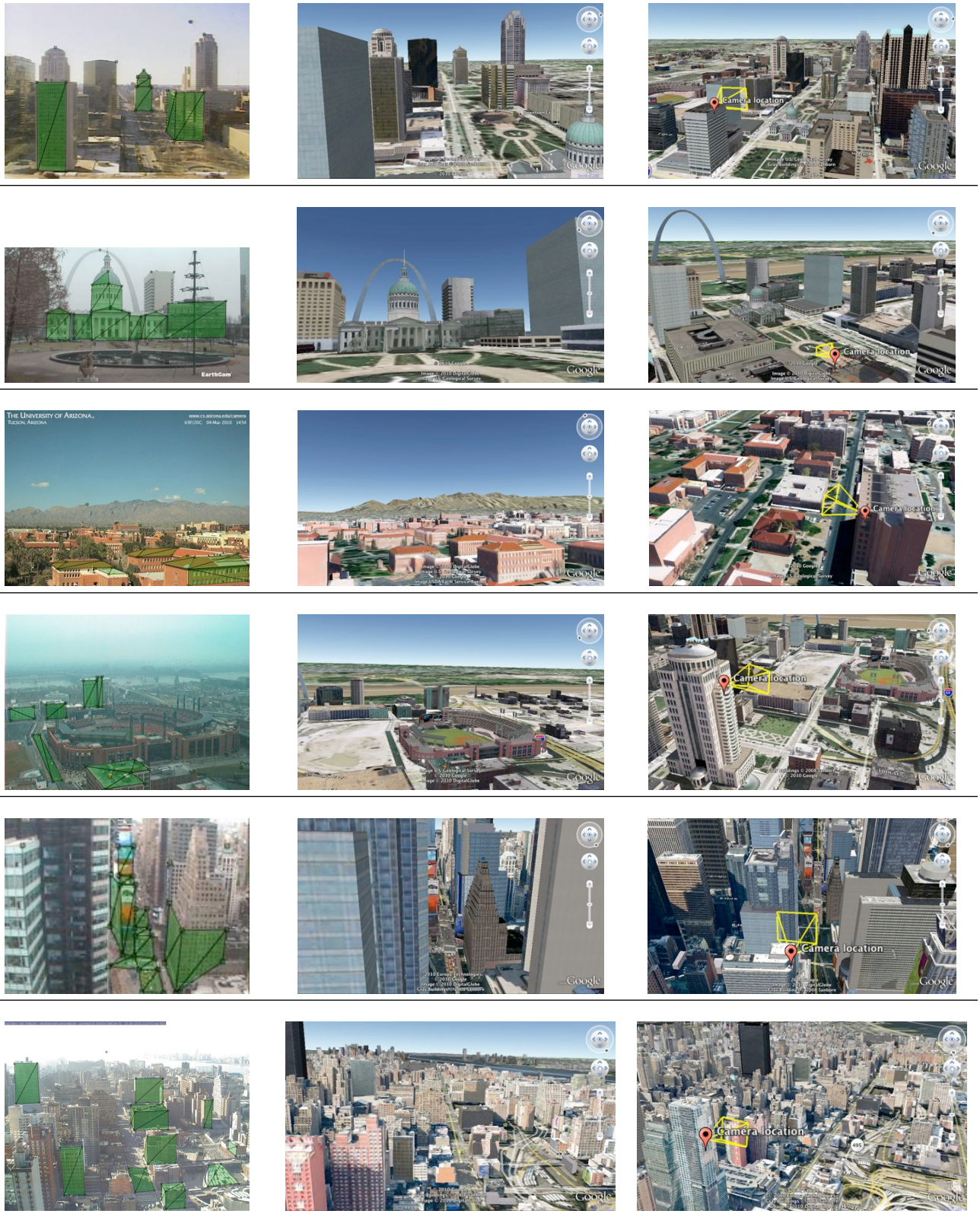
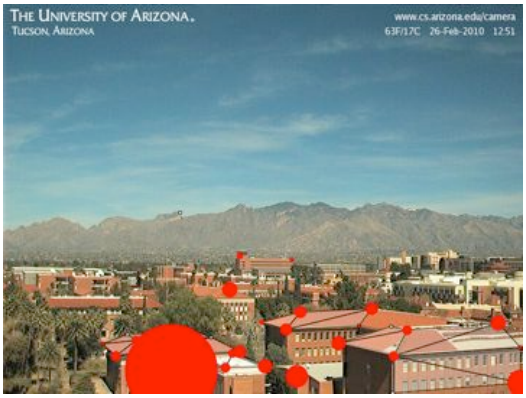
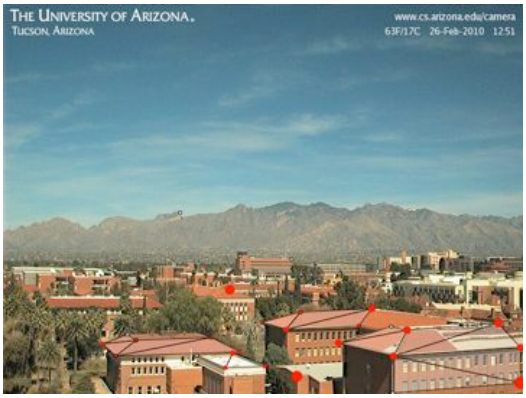


Figure 6: A selection of webcams and the polygons users have added to scenes (left column), the view from the positioned and oriented camera in Google Earth (center column), and the calculated location and orientation of the camera in the context of the scene (right column). The views in the center column have not been altered from the default calibrated camera view.



(a)

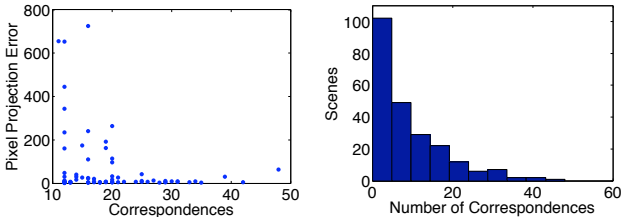


(b)

Figure 8: In each image, each correspondence is shown as a red circle, whose radius is the pixel projection error. The scene in (a) has a mislabeled correspondence, which increases the expected pixel projection error. Once this correspondence is removed and the scene is re-calibrated, the expected pixel projection error reduces drastically (b).

longitude, found in most geotags. From the calibration of the camera and the 3D geometry provided in Google Earth, we find not only the location and orientation of the camera itself, but the latitude, longitude, and altitude of *every pixel* in the camera's view. This "rich geotagging" provides much more complete information about an image, and this opens the opportunity for more in-depth analysis of a scene. For example, the Deep Photo system uses a series of geolocated points on an image to remove haze, annotate the image with geographic landmarks, and recolor the image to simulate the scene in different lighting conditions [7].

As one example, we demonstrate marking the region of the ground viewable by a calibrated camera. In most cameras, a significant portion of the image is on the ground plane. We use RANSAC [1] to fit a plane to the geographic data, which usually results in the set of points that lie on the ground. We use a Delaunay triangulation of these points and remove any triangles with excessively long edges. Since the camera is calibrated, we know the projection from 3D points to image points, so finding the image coordinates for each point in the mesh is trivial. This results in a textured mesh that approximates the ground plane of the scene, as



(a) (b)

Figure 9: The pixel projection error by number of correspondences is shown in (a). This suggests that scenes with a high number of correspondences have low pixel projection error. The number of correspondences per scene is shown in (b).

in Figure 10(c).

This method can be modified slightly to get the coverage of the camera, with obstruction from buildings. While most camera coverage maps only show camera position and possibly orientation, a coverage map from a richly geotagged image gives back the latitude and longitude of every point on the ground that the camera can see, as in Figure 10(d).

In addition to these initial applications, there are many potential uses for a richly geotagged image, including the following:

- In addition to the user-submitted correspondences, retexture the rest of the scene automatically to reflect current conditions.
- Create a network of camera coverage maps and index each location on the map with the set of cameras that can see that point. This would give a geospatial context to an entire network of cameras, rather than a single camera.
- As a source of validation data to evaluate algorithms that estimate depth.

5.2 Conclusions and Future Work

We present two publicly accessible, user-centric web interfaces that ease the process of camera registration so that an inexperienced user can quickly and easily register a scene and retrieve the full camera calibration. Not only does this give the scene live textures, but it gives the webcam image geospatial context, and offers a more complete understanding of the scene. We show that the client-server model that maximizes client-side computation offers a scaleable and realizable solution for large user-centric GIS applications. We discuss some characteristics and quality measures for the scenes the user-base has contributed, and show that this method of camera registration provides high-quality results. Furthermore, through the camera calibration, we can reveal the true position and orientation of the camera to create a richly geotagged image, where the geolocation of every pixel in the image is known. In the future, we plan to explore more possibilities for rich geotagging.

Our results suggest that, although participatory markup and integration of video streams into GIS applications may be challenging, it is ultimately feasible, given a user-friendly interface and sufficient documentation.

6. REFERENCES

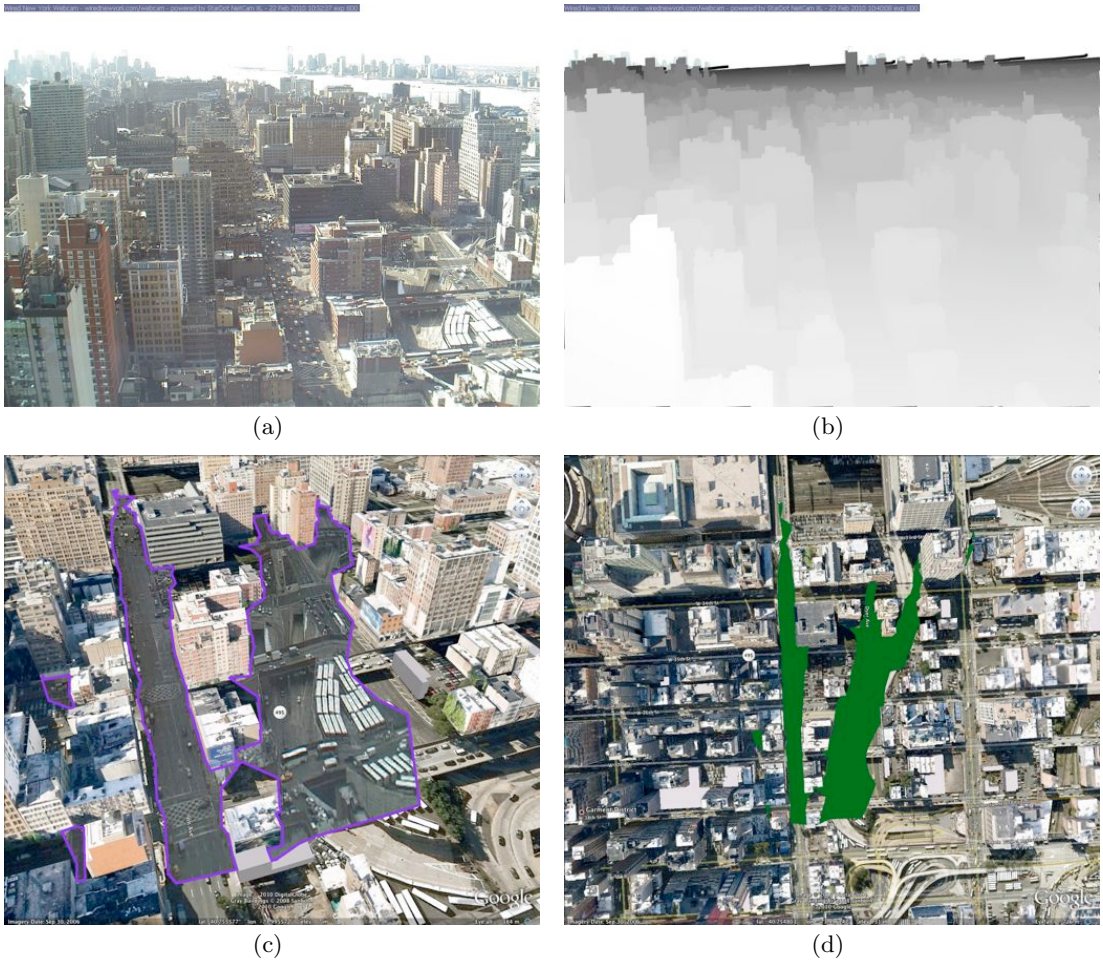


Figure 10: A calibrated webcam image (a) gives a depth image (b). From the set of geolocated points, we approximate a polygon that retextures the ground plane (shown with a violet border)(c), and gives a camera coverage map with occlusion from buildings (d).

[1] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[2] A. J. Flanagin and M. J. Metzger. The credibility of volunteered geographic information. *GeoJournal*, 72(3-4):137–148, 2008.

[3] M. Haklay and P. Weber. OpenStreetMap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12 –18, Oct.-Dec. 2008.

[4] N. Jacobs, W. Burgin, N. Fridrich, A. Abrams, K. Miskell, B. H. Braswell, A. D. Richardson, and R. Pless. The global network of outdoor webcams: Properties and applications. In *ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL GIS)*, Nov. 2009.

[5] N. Jacobs, N. Roman, and R. Pless. Consistent temporal variations in many outdoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2007.

[6] K. Kim, S. Oh, J. Lee, and I. Essa. Augmenting aerial earth maps with dynamic information. In *IEEE International Symposium on Mixed and Augmented Reality*, Oct. 2009.

[7] 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. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2008)*, 27(5):116:1–116:10, 2008.

[8] H. S. Sawhney, A. Arpa, R. Kumar, S. Samarasekera, M. Aggarwal, S. Hsu, D. Nister, and K. Hanna. Video flashlights: real time rendering of multiple videos for immersive model visualization. In *Thirteenth Eurographics Workshop on Rendering*, 2002.

[9] I. O. Sebe, J. Hu, S. You, and U. Neumann. 3d video surveillance with augmented virtual environments. In *First ACM SIGMM International Workshop on Video surveillance*, 2003.

[10] S. Wittens. Projective texturing with canvas. `http://acko.net/blog/projective-texturing-with-canvas`, 2008.