Abstract

This research designs and implements a web-embedded Google Earth application for disaster response field workers to upload images of disaster sites captured by Unmanned Aerial Vehicles (UAVs) and to type in text capturing observations that are not visible in the images, from the field. In addition to providing an interface for those on the field to insert images, it provides a format for viewing tiled imagery of both plan (taken aerially) and elevation (facing the sides of a structure) views, which does not exist at present. The Google Earth application accommodates plan views, and Microsoft's Photosynth application accommodates elevation views, but there exists no application for viewing a combination of the two. The research in this paper focuses on the manual uploading of images without GPS or compass stamps, and the design of a web-embedded Google Earth application for uploading and integrating images into a display. To upload images, all this approach requires of the user is an approximate location and an image file. Feedback from subject matter experts was used to measure success, and is positive.

Introduction

UAVs deployed in search-and-rescue missions such as that of the Berkman Plaza II Parking Garage collapse in Jacksonville, FL in December 2007 gather and return imagery with no GPS or compass information, making the images difficult to interpret both by emergency responders and by structural engineers. The images gathered consist of both elevation views (taken from the sides of a structure, see Figure 1) and plan views (taken aerially, see Figure 2), two types of images that are rarely displayed together cohesively. Displaying them together cohesively requires that they be placed on a model of the site, making their relative positions explicit as well as providing a sense of the "big picture" of the disaster site. Figure 3 indicates the relationship between plan and elevation images.

Additionally, providing an on-site method for uploading and presenting the images can expedite the process of getting information out to remote responders, rather than uploading the images after the mission is over (when perhaps less would be known about the relative locations of detailed images). This research designs a web application that can be accessed on-site to upload and present imagery, using the existing Google Earth software to solve the problems of inserting and displaying images of differing view type.

Related Work

Related work has been done regarding the presentation of UAV imagery for search-and-rescue robotics purposes. Michael Lindemuth has developed a method to automatically upload plan images taken from multiple robots and view them in the Google Earth application; however in his research the images were stamped with GPS and compass data, allowing automation. Kevin Pratt has explored technical requirements for the use of Miniature UAVs (MAVs) in search-and-rescue missions. In addition, Microsoft has developed Photosynth technology for the automated displaying of images taken from different angles and altitudes, and Google is currently exploring automated image tiling for search-and-rescue purposes, which would complement the application designed in this paper.

This research builds on previous work in "Multi-Agent Tracking and Recording in Google Earth" (Lindemuth 2007). Lindemuth's paper develops a methodology for real-

time tracking of multiple robots in Google Earth using a web interface, and implements a UAV-to-Google Earth data flow architecture, allowing the automated tracking of GPS-equipped UAVs. Lindemuth's architecture, in conjunction with the application designed in this paper, would prove very useful in facilitating the uploading of UAV imagery, but can only be used if the UAVs deployed are equipped with GPS (which they are not for the purposes of this paper). A point of differentiation is that in Lindemuth's paper data can only be added automatically, whereas this paper's application aims to allow manual uploading and editing of photographic and text data. Additionally, Lindemuth's web application delivers KML files to be opened in the Google Earth application, whereas in the application developed in this paper KML files can be opened in the web application itself.

"Overview of Requirements for Semi-Autonomous Flight in Miniature UAVs" (Pratt, Murphy, Stover 2006) determines field needs for the operation of MAVs, notably that 3 operators are recommended for each MAV and that GPS capability is unnecessary for operation. The application developed in this paper is intended for such situations in which GPS is not present and there is sufficient presence in the field to perform the task of uploading and displaying data.

The viewing of elevation images of differing angle and altitude is well handled by Microsoft's Photosynth technology, which uses image processing to place images on an existing model of a site. However, since the sites dealt with in search-and-rescue missions are expected to be to a degree unrecognizable from their prior undamaged states, photo recognition cannot be used to place them. Additionally, this technology requires that a model of the site be created long beforehand (a process that can take a matter of days, which response workers do not have). As Photosynth uses publicly available images to create models of sites, it requires that there be a large collection of images of the damaged site available in order to construct an accurate model of the site post-disaster.

Google is currently developing an application to handle the tiling of consistently angled imagery for search-and-rescue purposes. The research outlined in this paper differs from Google's by focusing on a Google-compatible application for viewing and inserting the imagery, in the hope that the two applications can be combined at a later time.

Approach

Three methods for displaying UAV images were compared: the Microsoft Photosynth application (see Figure 4), the Google Earth application (see Figure 5), and a static HTML image layout (as Pratt implemented after Hurricane Katrina, see Figure 3). They were evaluated according to the following criteria deemed necessary for a satisfactory display method:

Criteria	**Photosynth**	**Google Earth**	**HTML**
Ability to insert images from the field with no prior model of site	No	Yes	Yes
Ability to accommodate both plan and elevation images	Elevation images only	Yes	Yes
Ability to textually annotate features not evident from the photographs (such as building sides for elevation views)	Yes	Yes	Yes

Ability to provide a sense of the relative locations of images	Yes	Requires more accurate model of site	No
Ability to insert images with no knowledge of GPS or compass information	Yes	No	Yes
Google-compatible	No	Yes	No
Ability to integrate images into display in 1 minute per image or less	No	Yes	No

The Google Earth method was chosen as the most feasible method for use in the field, and its inadequacies (the need for precise GPS and compass information to automate the integration of images, and the requirement that a more accurate model of the disaster site be created due to its inevitably being unrecognizable from old satellite data) were noted. A subject matter expert evaluated alternative solutions and advised that a Google-compatible display method be chosen, as Google is developing image-tiling software to be used by search-and-rescue field workers. The recent release of the Google Earth Browser Plug-in and associated JavaScript API allowed for the design of a web interface for on-site search-and-rescue personnel to actually upload photographs from the field, and passing this data through PHP to a Google Earth-readable KML file, as opposed to manually scripting KML files for each image.

Implementation

The following section addresses the designed implementation of the application. Design constraints due to the use of the Google Earth Browser Plug-in are explained, followed by the application's functions, external interface as it concerns the user, and additional attributes of the application not addressed elsewhere.

a) *Design constraints imposed on implementation.*

The Google Earth Browser Plug-in requires that a specified JavaScript API be used. Although a constraint on design was that the implementation be Google-compatible to accommodate forthcoming image tiling software, the image tiling software being developed by Google has not been integrated into the application at present time, so the current application does not incorporate the tiled 3d model of the disaster site. The Google Earth Browser Plug-in currently only works with these browsers in Microsoft Windows XP and Vista: IE 6.0+, IE 7.0+, Firefox 3.0x, 2.x or 2.0x, Netscape 7.1+, Mozilla 1.4+, and Flock 1.0+ (see http://code.google.com/apis/earth/documentation/index.html) but will be extended to other operating systems in the near future.

b) *Functionality.*

This software provides a method for on-site responders to upload images and information from UAVs at a disaster site, and allow remote access of that information as well as modification capabilities. The expected users are UAV operators and other on-site emergency responders. A consulted subject matter expert's suggestions included: the ability to insert text and other pictures from the user's personal cameras, the inclusion of GPS coordinates and time & date stamps for forensics, and the ability to go back and augment these entries after leaving the site. The functions are as follows:

1. **Choose Site**: Upon initializing the application, the user will be able to select which site's data to view (for example: "Berkman Plaza II Parking Garage" or "Hurricane Katrina"). Upon selection, the web application will localize to that location and load the data associated with that site.
2. **Create New Site**: User is prompted for the following information:
    a) Site name
    b) Approximate central GPS location for window-viewing purposes
    c) Additional description of activity and events at site

The data are requested via JavaScript and passed via PHP to a new folder containing the information in KML format. Once a site is created user can create and view its Placemarks.

3. **Create Placemark**: User is prompted for the following information:
    a) Image (jpg file type)
    b) File name of image (String type)
    c) GPS data (numeric type: latitude, longitude, altitude, date and time)
    d) Compass data (numeric type: heading, pitch, roll)
    e) Description of feature(s) noticed at site but not visible in image (String type)

The data are requested via JavaScript and passed via PHP to a script that converts them to a KMZ file. The KMZ file is located in a folder of features that are automatically opened when the application is initialized.

3. **View Placemark**: After Placemark is created, user can click on it to view all inputted information in a pre-set format. There will be an option in the Placemark to edit its description.
4. **Edit Placemark**: While viewing Placemark, there is a button to add information to the description. User is prompted for additional description and can insert text and photos using HTML. This is passed through PHP and added into the existing KMZ file containing the placemark.

c) *External interfaces.*

This software is an extension of the Google Earth Browser Plug-in software. Using the Google Earth API, JavaScript and PHP, the software is web-embedded and allows the user to input the following relevant information (as specified above). Not all of the information is required as it is not all expected to be readily available (for example, GPS data are not currently obtainable from the UAV images). Once the information is received it will be converted to a KMZ file, a zipped file consisting of the image and the KML code defining its placemark on the Earth. Users can click on the Placemark to view all information contained in it. Additionally in the Placemark there will be a function to edit the description field, allowing the user to input additional pictures (externally hosted?) and information.

d) *Attributes.*

Building upon the Google Earth Browser Plug-in requires less memory use than running the Google Earth application. It additionally requires less application development time since Google Earth Browser Plug-in has an API for interactivity. Additionally using the Browser Plug-in allows remote access from computers not pre-

loaded with the application. Downloading the Browser Plug-in is still necessary but uses less memory than downloading the Google Earth application.

For security purposes, the web application should be hosted in a directory that is username- and password-protected, to prevent unauthorized individuals from inserting incorrect or inaccurate information. To allow external individuals to access the information, a smaller version of the application, without the Create Placemark/Edit Placemark functions, can be hosted in a public directory.

Demonstrations of Display Possibilities

Display possibilities in the Google Earth application were first demonstrated, followed by an HTML image map implementation. The final demonstration was the development of the web-embedded Google Earth application designed in the Implementation section. The objective of the demonstrations was to isolate additional requirements of the application, and to determine whether the method clearly displayed images and was a reasonable method to be used in the field.

a) *Google Earth*

This experiment built upon Lindemuth's approach of using Google Earth Placemarks to display detailed imagery. However, unlike in Lindemuth's case, the imagery worked with contained no GPS stamps (therefore its placement on the Earth could not be automated but had to be manually estimated) and consisted of both plan and elevation views (as opposed to just plan views). As the viewer can change the viewing angle and altitude in Google Earth, one can view both kinds of images contextually, provided that there exists a model of the site that accurately reflects its appearance in individual photographs (as opposed to existing models created before the disaster in question). To indicate detailed images of a particular location, Google Earth Placemark features were used due to the ability to view their contents simultaneously with the site model. Two models for the Berkman Plaza II Parking Garage site were evaluated for the viewing of Placemarks:

1) *Google 3D Model*: A publicly available 3D model for the structure was visible by selecting the "3D Buildings" layer in the Google Earth application. The model reflected the building pre-collapse and was therefore not an accurate representation of the disaster site's appearance.

   This model proved useful in displaying elevation views, as it allowed images to be viewed with a sense of where on the structure they were located (see Figure 6). However, the model was not useful for displaying plan views as it looked completely different from the disaster site, making it hard to visualize the extent of the damage and achieve a complete picture of the site from the collection of plan images. In addition, the 3D model was "solid," meaning that a Placemark with an image taken from the inside of the structure could not be viewed while the structure was set as visible.

2) *Manual image tiling*: To more accurately represent the site and display the plan view imagery, our own model was built using Google SketchUp. A blank model of the structure's dimensions was publicly available in the

Google 3D Warehouse. By tiling the elevation views taken from consistent sides, and the plan views (see Figure 7), and setting these tiled images as the faces of the model, the user would see a more accurate representation of the site. The plan images were manually tiled and set as the top face of the model in the experiment (see Figure 8).

The tiling took about 3 hours to complete manually and the result received positive feedback from subject matter experts, however the process was deemed not feasible for use in the field. Current work is being done by Google to address the tiling of consistently angled imagery in the field.

The final display of the images and tiled model in Google Earth was clear and received positive feedback, but the methods employed were limited by factors making them difficult to use in the field. The creation of an accurate model was time-consuming, and as the images had to be placed by hand (since they were not stamped with GPS information) the user had to hand-code large KML files to insert them into the display. Additionally, as Google Earth is a somewhat "large" application requiring lots of power, it may run slowly on the laptops used in the field.

b) *HTML Image Map*

Another method for displaying the imagery, without Google Earth, was explored by developing an HTML and JavaScript web-based image map (see Figure 9). The tiled plan view was used as a clickable image map, with detailed photographs of the selected area appearing in a pop-up window.

This method took approximately 3 hours to implement, not counting the prior time tiling the base image, and was therefore deemed not feasible in the field. The result received positive feedback due to the clarity of the display but ultimately did not allow for the dynamic uploading of additional photographs.

c) *Web-Embedded Google Earth Application*

Using the Google Earth Browser Plug-in and its associated JavaScript API, a web-embedded version of the Google Earth placemarks display was created. To use the API a unique access key had to be obtained from Google Maps (this is publicly obtainable at http://code.google.com/apis/maps/signup.html). The application, located at http://issrt.usf.edu/~ncweber/Jacksonville/map.html, automatically localizes to the Jacksonville site and loads a KML file containing the Jacksonville Placemarks data. The user can click on any Placemark to view its contents (see Figure 10). Additional functions, such as those to create a site, choose another existing site, and create an additional Placemark for the current site, are to be implemented using PHP and XML as detailed in the Implementation section.

The benefits of the web interface are its accessibility by both on-site and remote personnel, and that it provides the same flexible viewing of images as the Google Earth application, while requiring less computing power. It also, however, includes the similar problem of requiring a model that is time-consuming to tile.

d) *Discussion*

Of the three methods explored, the final web-embedded Google Earth application provided the best fulfillment of requirements as outlined in the Approach section above. The main area to be worked on is that of image tiling to create accurate site models, to be addressed by Google. Additionally, the estimation of image locations for placement in all three methods would be greatly aided by GPS-equipped cameras on the UAVs. With GPS, the placement could be automated which would greatly reduce the amount of time spent inputting images.

Conclusions

The application designed in this paper provides on-site search-and-rescue personnel with a web-based method for uploading and displaying UAV images without knowing their exact coordinates or compass information. The web-embedded Google Earth application designed allows field workers to upload, estimate the location of, and integrate into a display images of the disaster site. Additionally, field workers can create a new site for each new mission, view all images from prior sites, and insert textual information indicating features not evident from the imagery. It is a reasonable preliminary solution to the problem that can be utilized on-site by a laptop.

However, main areas for improvement are the generation of site models, the process of which could be greatly aided by automated image tiling, and the process of inputting locations for uploaded images, which could be easily automated if the UAV cameras provided GPS stamps (as is the case in Lindemuth's implementation). Further, if the UAV cameras provided GPS and compass information in their images, the images could be uploaded automatically as well. In the case that the uploading were automated, all that would be left to do is generate an image-tiled model of the disaster site.

Acknowledgements

This research was funded by the Computing Research Association's Committee on the Status of Women in Computing Research, as part of the Distributed Mentoring Program. Subject matter experts Sam Stover (FEMA US&R Indiana Task Force 1) and Michael Lindemuth provided valuable input regarding requirements and design of the application. Kevin Pratt and Jeff Craighead provided access to images and technical assistance. Finally I would like to thank my mentor, Dr. Robin Murphy, for her guidance and advice throughout the project.

References

Michael Lindemuth. Multi Agent Tracking and Recording in Google Earth. Senior Honors Thesis, Tampa, FL: University of South Florida, 2007.

K. Pratt, R. R. Murphy, S. Stover, and C. Griffin, "Requirements for Semi-Autonomous Flight in Miniature UAVs for Structural Inspection," AUVSI Unmanned Systems North America, 2006.

Noah Snavely, Steven M. Seitz, Richard Szeliski, "Photo tourism: Exploring photo collections in 3D," ACM Transactions on Graphics (SIGGRAPH Proceedings), 25(3), 2006, 835-846.

Figures Referenced in the Text

Figure 1: Example of an elevation image that would have to be displayed

Figure 2: Example of a plan image that would have to be displayed

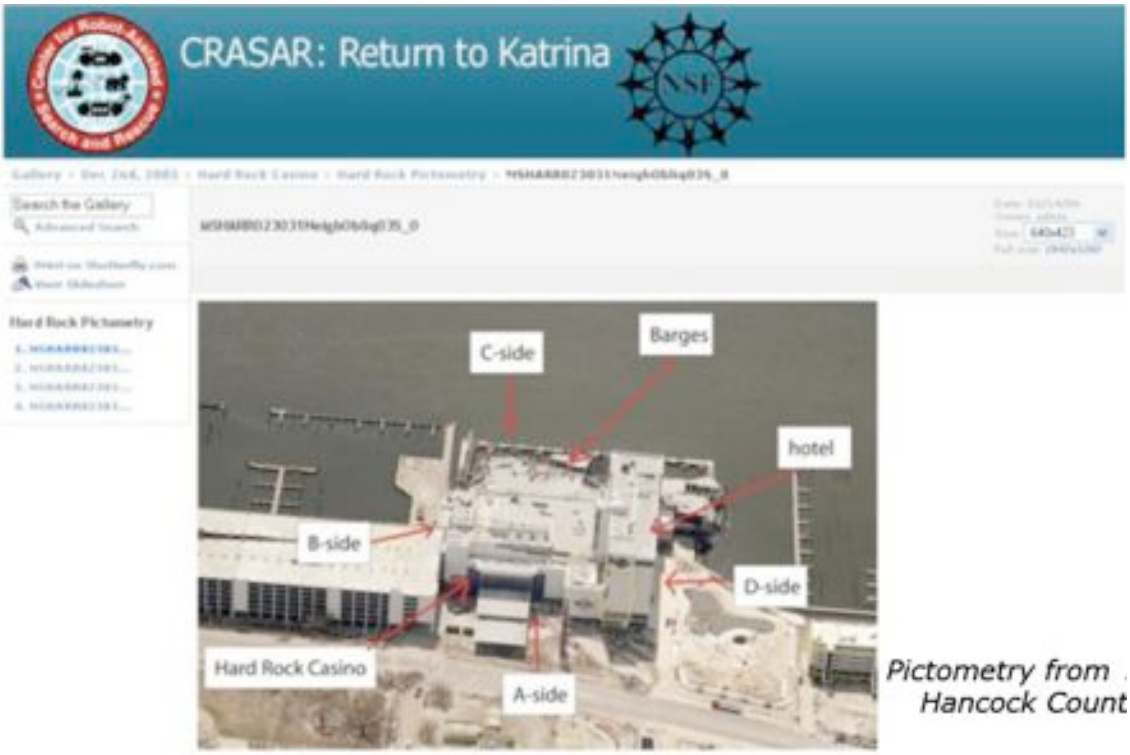

Figure 3: A plan image indicating areas where there would be elevation images taken

Figure 4: Using Microsoft Photosynth to place angled images on a site model

Figure 5: Displaying a detailed image on a model in Google Earth

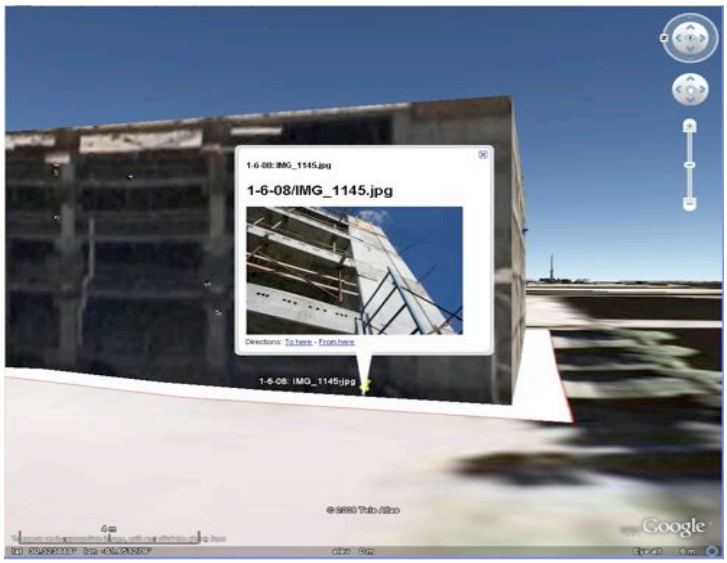

Figure 6: Viewing an elevation image in Google Earth

Figure 7: Manually tiled plan view images

Figure 8: Tiled plan view on model

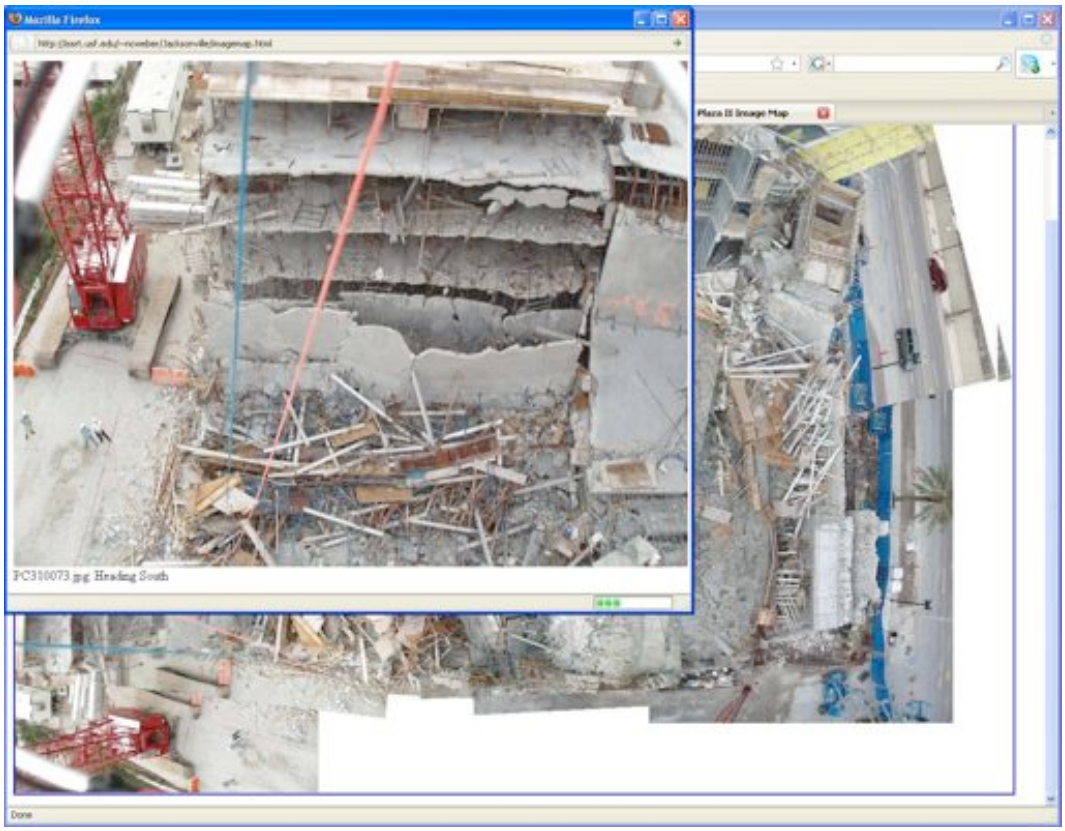

Figure 9: HTML Image Map

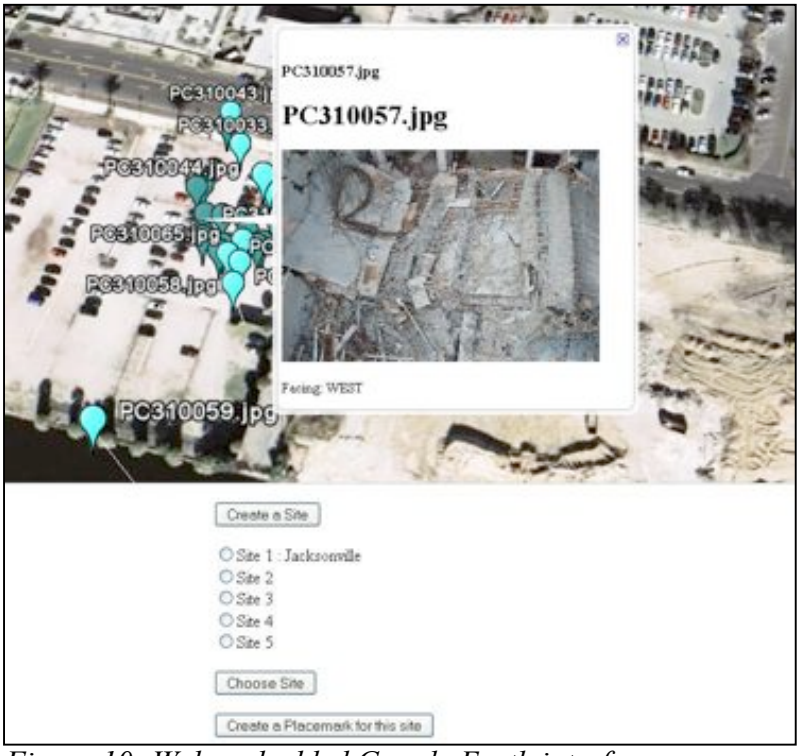

Figure 10: Web-embedded Google Earth interface