Presentation of Geo-Referenced Videos with Google Earth

Lingyan Zhang
School of Computing
National University of Singapore
Singapore 117417
lingyan@comp.nus.edu.sg

Roger Zimmermann
School of Computing
National University of Singapore
Singapore 117417
rogerz@comp.nus.edu.sg

Guanfeng Wang
School of Computing
National University of Singapore
Singapore 117417
wanggf@comp.nus.edu.sg

ABSTRACT

Geo-tagging is becoming increasingly common as location information is associated with various data that is collected from a variety of sources. In the field of media, images and most recently videos, can be automatically tagged with the geographic position of the camera. Geographic location provides an interesting means of browsing through, and "drilling into," large video repositories. At the same time complementary efforts are underway to create so-called mirror-worlds – large-scale environments that are essentially detailed computer-models of our three-dimensional real world. However, these mirror worlds are for the most part static (for example they include buildings and trees). Here we describe our efforts to bring together these two paradigms by enabling Google Earth (and similar tools) to come to life with videos that are geographically and perspectively correctly placed as "viewports" inside the world. In essence this reflects the next step in the evolution of mirror worlds and maps with (panoramic) still images towards a three-dimensional, dynamic environment.

Categories and Subject Descriptors

D.2.2 [**Software**]: Design Tools and Techniques—*User interfaces*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Graphical user interfaces (GUI)*

General Terms

Design, Experimentation

Keywords

Geotagged video, IFrame shim, Google Earth, KML

1. INTRODUCTION

Technological advances have lead to interesting developments in the following three areas:

- Location and direction information can now be affordably collected through GPS and compass sensors. By combining location data to with other information interesting new applications can be developed. Location data also gives rise to a natural organization of information by superimposing it on maps that can be browsed and queried.
- While maps are two-dimensional, three-dimensional mirror worlds have started to appear. In these networked virtual environments, the real world is "mirrored" with digital models of buildings, trees, mountains, etc. Mirror worlds allow a user to explore, for example, a city from the comfort of their home in a very realistic way.
- High quality video camcorders are now quite inexpensive and the amount of user collected video data is growing at an astounding rate.

Our goal with the presented project is to harness the confluence of the above developments. Specifically, we envision a detailed mirror world that is augmented with (possibly user-collected) videos that are correctly positioned in such a way that they overlay the 3D structures behind them, hence bringing the mostly static mirror world to live and providing a more dynamic experience to the user who is exploring such a world. To test the feasibility of this idea we have collected a number of videos that were augmented with compass and GPS sensor information. We then used Google Earth as a backdrop to overlay the acquired video clips in the correct locations.

The rest of this paper is organized as follows. Section 2 describes work related to our research. In Section 3 we present the methods used in the proposed system and in details. Challenges and open questions that require future considerations are discussed in Section 5. Finally, Section 6 provides the conclusions.

2. RELATED WORK

There exist only a few systems that explore the use of geo-located videos. Some earlier work has investigated 2D geo-referenced video acquisition, search and presentation. Here we specifically expand on this into three dimensions. Navarrete and Blat [6] defined a method of indexing and retrieving geo-referenced video sequences based on their geographic content. Although they proposed a valuable method for indexing and retrieving video sequences based on geo-referenced information, our work is based on the viewable direction in 3D which requires extension of the prior work.

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

Simon *et al.* [11] presented a local visibility model with a suitable XML-based prototype implementation. Using an XML-based description is notable and desirable, as it can by applied to many applications. Inspired by this idea, we use KML since KML can be automatically parsed by Google Earth, which is our current implementation environment [8].

Several groups are actively researching videos in 3D environments. First, Google groups have achieved the embedding of photos and videos in Google Maps and Google Earth. These photos and videos are both based on their geolocations, which had to be uploaded manually. GPicSync is a Google project that aims to automatically insert locations into users' photos. Thus, such photos can also be used with any geocode-aware application like Google Earth [10]. In addition, Google Earth supports the simple embedding of videos. However these videos are specified via URLs from YouTube which does not represent a user defined video insertion. As a result, we are interested in a more flexible approach to present videos such that they can be overlaid on a variety of applications. To this end we are currently using an IFRAME shim [5] which can create an IFRAME and a video object overlay dynamically, after – for example – Google Earth is loaded. This method can be applied in various environments because it requires limited integration with the underlying application. As a second example, Ni *el at.* [7] embedded video streaming into the *Second Life* virtual world. They used a fixed screen to study the video quality, which represents quite a different focus compared with our research direction. In other words, they investigated the different quality levels experienced by users, and hence their work is orthogonal to ours and could be applied to our techniques. Third, some existing systems explore the merging of real and virtual worlds [12]. Their major contribution is to investigate a spectrum from completely real environments to a completely virtual ones with augmented reality and mixed reality falling in between. However, these techniques are not based on geo-referenced information.

3. APPROACH

Our objective is to find geo-referenced video segments within a relevant area and then to present them as augmentations in a virtual, three-dimensional environment such as Google Earth. Figure 1 illustrates the overall architecture of our system. The main components are a browser-based web client, a web server with an associated database, a streaming media server, and remote access to Google's Earth and Maps servers.

We require the acquisition of videos that are fused with detailed geo-reference information. In our prior work we proposed a video scene model in which videos are continuously augmented with detailed sensor data such as the current GPS location and camera viewing direction (obtained through a compass heading) [2]. In order to collect geo-referenced videos with the necessary sensor meta-data we have built several acquisition prototypes and software applications. Section 4 describes one of our implementations designed for the Apple iPhone. Figure 2 shows an excerpt from the XML-encoded geo-data that is collected by the iPhone application and uploaded to a suitable search engine that provides the backend functionalities [1]. One of the challenges in virtual environments is that it may not be very easy to specify the user's region of interest (i.e., the query area). For example, currently Google Earth does not

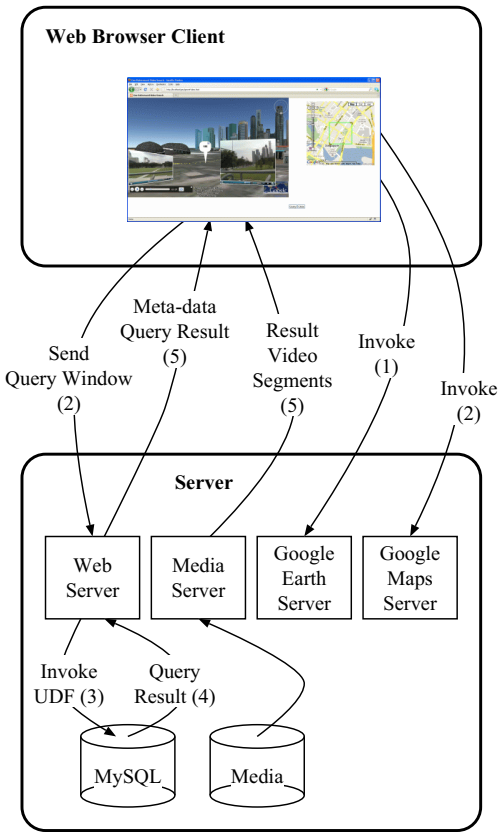

Figure 1: Architecture of the proposed system.

support the specification of a spatial query rectangle to delineate a search area. For this reason – and because a query area is more naturally expressed in 2D – we use Google Maps to let a user select a query window. The search results are then shown properly placed in Google Earth.

Our development environment is based on open source tools and software such as XAMPP (Apache, MySQL, PHP), the Wowza Media Server and the Flowplayer (a video player to render Flash-encoded and other content) and with technologies such as Ajax, IFRAME shim and KML. The Wowza Media Server allows the streaming of video content, similar to the Adobe's Flash media server. Using this combination of media server and player, any segment within a video, specified by start and end timecodes, can be played. We use this feature to extract the most relevant clips from videos which may potentially be very long and cover a large geographical area. The client implementation is browser-based, and hence it is convenient to access from almost anywhere. There are three important components in our system. First, meta-data collection part which is the previous work in our research. Second, the database implementation which contains our 3D meta-data. Third, the Web interface which is the main component in this system. In this part, users can not only search videos through entering a location based rectangle, but also get the query results in the same page.

3.1 Meta-data Management

The meta-data is stored in a MySQL database to allow for efficient access and search. Our design can accommo-

```
<?xml version="1.0" encoding="UTF-8"?>
<array>
    <array>
        <date>2010-05-28T08:23:22Z</date>
        <real>1.3001484800000001</real>
        <real>103.76716863999999</real>
    </array>
    <array>
        <date>2010-05-28T08:23:59Z</date>
        <real>1.3001484800000001</real>
        <real>103.76716863999999</real>
    </array>
</array>


<?xml version="1.0" encoding="UTF-8"?>
<array>
    <array>
        <date>2010-05-28T08:23:22Z</date>
        <real>167.23540386557579</real>
    </array>
    <array>
        <date>2010-05-28T08:23:23Z</date>
        <real>167.23515872657299</real>
    </array>
</array>
```

Figure 2: Sensor meta-data collected together with geo-referenced video streams by our iPhone application. The XML encoded data includes timestamps, GPS coordinates (top fragment) and compass headings (bottom fragment).

date a variety of sensor meta-data information as is shown in Table 1. Figure 2 illustrates the data that we collect on with the iPhone application, however, some other implementations can provide additional information (e.g., altitude, viewable distance). The most significant 3D meta-data in our current system is the heading (in the database, it is the theta attribute), latitude and longitude. The collected 3D data basically represent the camera direction and location as a vector which describes a 3D field-of-view (FOV). Based on this model, we are able to show videos in the correct positions within the 3D scene view of Google Earth.

$filename$	Uploaded video file
$\langle Plat, Plng \rangle$	<Latitude, longitude> coordinate for camera location (read from GPS)
$altitude$	The altitude of view point (read from GPS)
$theta$	Camera heading relative with the ground (read from compass)
R	Viewable distance
$alpha$	Angular extent for camera field-of-view
$tilt$	Camera pitch relative with the ground (read from compass)
$roll$	Camera roll relative with the ground (read from compass)
$ltime$	Local time for the FOV
$timecode$	Timecode for the FOV in video (extracted from video)

Table 1: Schema for 3D field-of-view (FOV) representation.

In MySQL, we provide the proper functionality through User Defined Functions (UDF) to perform FOV-matching as part of the query execution. The use of UDFs allows us to perform operations on data types that are not natively supported in MySQL. An UDF is executed within the engine of the database, hence being able to take advantage of the execution environment. The UDF was developed and implemented as part of our prior work. One current limitation is that only searches in 2D space are supported. Because of this, the altitude parameter is not implemented. In other words, the search is still performed on the 2D data and the results shown in Google Earth are then displayed as 3D sensor data.

3.2 Web User Interface

Perspective video, i.e., transforming video from a 2D plane into a projected plane in a 3D virtual space in accordance with the user's viewpoint, is one of the major tasks for web-based video overlapping applications. In this domain, there exist several viable solutions:

- Existing plug-in-based Rich Internet Application (RIA) technologies such as Adobe Flash and Microsoft Silverlight support 3D video rendering capabilities. While available for rapid prototyping, these environments require overlapped web services to provide corresponding RIA-compatible APIs.

- Pixel-level image transformation is also a feasible solution, but it requires significant client-side processing power.

- A Cascaded Style Sheets (CSS) 3D transform has been proposed by Apple Inc., and it is now under development by the W3C CSS level 3 [9]. This method transforms the coordinate space of a video element through a simple change of its transform properties.

- An IFRAME shim can establish a layer on top of the Google Earth web browser plug-in (or other web pages). The IFRAME can aid in the process of rendering videos, and is flexible in any environment. Without this technology, we cannot see the videos with an appropriate view point.

Considering both practicality and feasibility, we chose the IFRAME shim approach as our main technique to overlay 3D perspective video. Hence, when the viewing direction changes by a certain angle, the video also changes accordingly. With this notion, the users will get a more intuitive and immersive experience. Additionally, the meta-data will be stored in a KML file which allows to automatically invoke an animated tour through Google Earth. This is a relatively new capability of Google Earth which can help us automatically traverse the environment. Furthermore, the camera trajectory will also be shown in the 3D world. With the presentation of the trajectory, the users will explicitly follow the camera movement associated with the video.

In Google Earth, the number of modeled 3D buildings varies among different cities, but overall the number is steadily increasing. When 3D building structures exist, we can more convincingly overlay the captured video with the virtual world. When viewing these buildings we see whether the scene in a video matches the same position in the virtual

world. We can also observe how accurate the these 3D buildings have been modeled. Note that due to the current limitation of Google Earth, which does not allow the drawing of dynamic rectangles, we use Google Maps to enter the query region by applying the Google API. Therefore, our query mode is currently 2D, which may be extended in the future.

There are also a number of other technologies used for our web interface. To embed Google Earth and Google Maps in the same web page, we use Ajax. To achieve the interaction between Google Earth and Google Maps interfaces, we used are the *google.earth namespace*, the *GEView interface*, and the *Maps API GPolygon* [4]. The specific details of each technology are given below.

- The *google.earth namespace* contains global functions to support to the use of the Earth API interfaces. We attach a listener to Google Earth for a specific event, which means that if Google Earth moves, the program will be aware of the movement and simultaneously move Google Maps.
- The *GEView interface* checks the view behavior of the observer camera in Google Earth. There is a function that can return the global view region that is visible. It is noteworthy that the returned region may not be very accurate because it will be larger than what is strictly visible.
- *Maps API GPolygon* is an interface to create a polygon in Google Maps. Through this the users will directly get a view of the query region.

Figure 3 shows the video results using our example Geo-Referenced Video Search engine [1]. As can be seen, the web browser interface embeds Google Earth and Google Maps on the same page. Superimposed on top of Google Earth are our video results, while in the lower left bottom is the tour progress bar. The indicator on the progress bar points out the corresponding position within the time interval.

3.3 Client and Server Communication

In our system we need to exchange data between the clients and server corresponding to the query and result data. The overall architecture is shown in Figure 1. The numbers (1) through (5) indicate the sequence of interactions between a client and the server. Initially the user sends a query window to the server. There, the data will be processed by means of using PHP to invoke the UDF then returning the query results to the PHP code. The query results contain a KML file and video clip information which is used to play back the video clips from the media server. Finally, the results will be sent to the client where they are shown in Google Earth.

The main functionality in this client-server interaction is coded with Ajax. With this technology, web applications can retrieve data from the server asynchronously without interfering with the display and behavior of the existing page. At the same time, because of the properties of Ajax, we can establish an dynamic interfaces the web page [12].

4. PROTOTYPE

We have been implementing the proposed system as part of our ongoing project work. We designed and implemented a prototype geo-referenced video acquisition module on an Apple iPhone 3GS handset, which provides the necessary built-in GPS receiver and compass functionality. Below we describe our current application implementation. Please note that we have started to collect real-world data with this platform for further studies in the future.

4.1 iPhone Geo-Video Acquisition Application

Our Geo-Video App was developed with Apple's *Xcode* development environment for iPhone OS version 3.1.2 or later. Most parts were written in Objective-C, with a few fundamental sections in C. The Geo-Video App is composed of six functional modules: (1) video stream recorder, (2) location receiver, (3) orientation receiver, (4) data storage and synchronization control, (5) data uploader and (6) battery status monitor. Below we will describe each module in more detail.

Video Stream Recorder. This module employs the UIKit Framework of the iPhone OS Cocoa Touch Layer to invoke the built-in camera. Among the three video formats that the iPhone supports, we decided to use the medium quality profile. However, this could be changed based on user needs. Table 2 summarizes the audio and video acquisition parameters.

Parameter		Description
Format		MPEG-4
Format profile		QuickTime
Overall bit rate		861 Kbps
	Video	Audio
Format	AVC	AAC
Format profile	Baseline@L3.0	LC
Bit rate mode	Variable	Constant
Bit rate	794 Kbps	64.0 Kbps
Resolution	24 bits	16 bits
Resolution (pixels)	640 × 480	
Aspect ratio	4:3	
Frame rate	30 fps	
Frame rate mode	Variable	
Colorimetry	4:2:0	
Scan type	Progressive	
Channel(s)		2 channels
Sampling rate		44.1 KHz

Table 2: iPhone audio/video capture parameters.

Location and Orientation Receiver. To engage and control the built-in GPS receiver and magnetometer, we make use of the Core Location Framework in iPhone OS Core Services Layer. Location data consists of longitude and latitude and we can regard the position of the mobile phone exactly as the position of the camera. For the orientation information, however, we discovered an interesting difference between the true pointing direction and the device heading. Therefore, our system also fetches the accelerometer data from the UIKit Framework to determine an adjustment and ensure that the data that is recorded represents the camera's direction, even when the phone is held vertically.

An interesting aspect in sensor data acquisition is the sampling frequency. In our application we set the update frequency based on a distance filter for the location data and a fixed sample rate for the orientation information. A location update is triggered whenever a distance movement of more than 10 meters is detected. The sampling rate for the

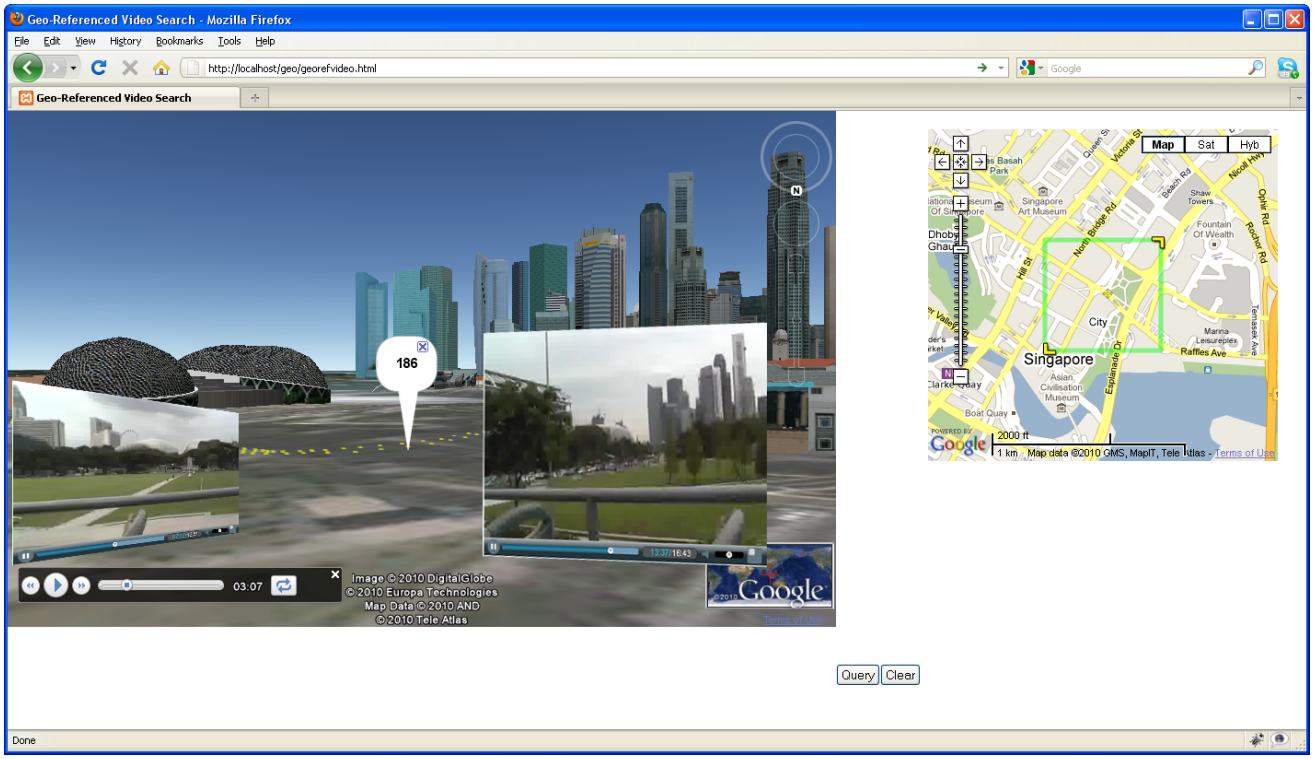

Figure 3: Geo-Referenced Video Search Results in Google Earth.

compass is set to 30 per second. Experimentally, with these settings we can discover viewable scene changes well while saving battery energy as much as possible. Furthermore, we set an expiration deadline for every data item obtained. If the location coordinates were obtained longer than 5 seconds ago, we consider this data as stale. For orientation data, we set its lifetime to 2 seconds because of its higher variability.

Data Storage and Synchronization Control. This module manages the storage of the sensor data on the device's flash disk. The goal is to utilize a flexible data format that can be easily ingested at a server. In this situation we choose a *Property List* in the Core Data Framework as our structured data representation. The *Property List* provides an XML-based abstraction for expressing simple hierarchies of data.

To provide synchronization, we extract the duration, encoded date and time from the video via the MOV multimedia framework. We then add timestamp information to every sensor data record to establish the relationship between a video clip and its corresponding geo-sensor information. Time is represented in Greenwich Mean Time (GMT), to avoid time zone issues. Files include the timestamp as part of their filename to allow for easy disambiguation.

Data Uploader. This module employs an open source wrapper around the CFNetwork API in iPhone OS Core OS Layer, named *ASIHTTPRequest*. This third-party class makes some of the more tedious aspects of communicating with a web servers easier and it is suitable for performing basic HTTP requests and interacting with REST-based services (GET/POST/PUT/DELETE). The Data Uploader transparently utilizes Wi-Fi, 3G or 2G cellular networks to transmit data files. Importantly, this module implements our two different strategies: (1) both video and sensor files are uploaded concurrently and (2) only the sensor files are uploaded first, while the video files may be transmitted later. Video files on the flash disk are tagged whether they still need to be uploaded.

4.2 User Interface

Figure 4 shows two screenshots of our Geo-Video App. When the user launches the software, he or she will first see a welcome view (left side of Figure 4). A user can either choose to start capturing a new video, or continue to upload video clips whose sensor data was previously uploaded. If the user touches the START button, a camera viewfinder will be displayed (Figure 4 right) and the user can then record, stop, cancel or edit a video clip via this interface just like they usually do in the iPhone's default camera view. However, our system additionally starts to record geo-referenced information from the GPS and the digital compass. The sensor data is stored to the device at the time when the video is saved to the camera roll and flash disk. Next, an uploading screen guides the user through the next step. A destination URL is displayed (which can be changed) and either the sensor information only or both the sensor and video files can be uploaded. As mentioned earlier, saved videos can be uploaded at a later point in time directly from the welcome screen.

5. CHALLENGES

Our study presents a novel approach to automatically positioning and displaying videos in 3D environments. While the results are very encouraging, we also found a number of challenges that will need to be addressed in our future work.

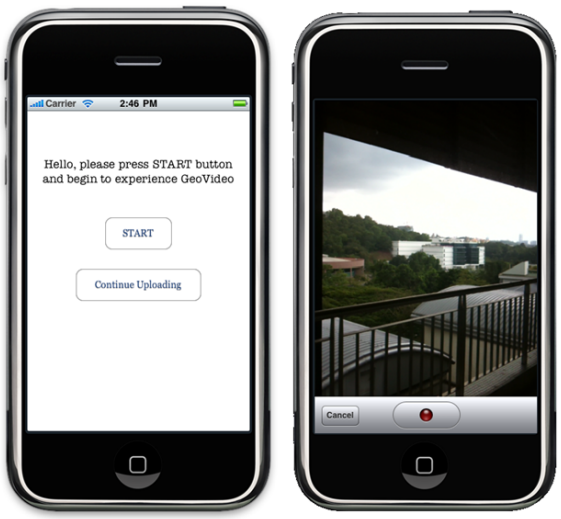

Figure 4: Geo-Video iPhone application prototype.

Below we provide a description of four specific issues that we faced and that require further investigation.

First, the acquired sensor data in our case was not using the same coordinate system as Google Earth or Google Maps. Therefore, the data needs to be converted so that it is compatible with systems such as Google Earth. Our experimental GPS sensor data information is based on a format of degrees, minutes, and seconds. However, the longitude and latitude in Google Earth uses a decimal degree format as represented by the WGS84 coordinate system [3]. The broader issue here is that multiple coordinate systems need to be supported and data needs to be correctly identified and converted to support large-scale video acquisition and applications.

Second, sensor values by their very nature are sometimes noisy and errors and drop-outs may occur in practice at times. This problem of data quality will require further study, for example, to investigate interpolation and error correction methods. Another issue may be the accurate registration of 3D buildings in Google Earth (or other virtual worlds). Furthermore, the 3D datasets are far from complete and only a few cities have extensive 3D structures in Google Earth. When buildings are missing then naturally there will be a visual mismatch between any video and the 3D world in that area. This may disrupt a user's navigational experience. However, we expect that in time most 3D buildings will be modeled.

Third, as mentioned earlier, current display technology is mostly 2D and this makes it difficult for the user to specify a 3D query region through, for example, mouse interactions. In our prototype we use Google Maps to aid in the query area definition, but eventually a full 3D query input would be desirable.

Finally, there is the practical challenge of overlaying videos on an application such as Google Earth. Some interfaces exist to deal with images and videos. Although they have rudimentary support for geo-location information, they are still not suitable for our research. For example, existing applications in Google Earth only show *YouTube* videos which are specified by some URL information. We require more flexibility which led us to the selection of the IFRAME shim method instead using the Google API. In addition, we use our own media server which can manipulate the source video clips by extracting segments, or perform other operations. A current limitation is related to 3D perspectives. With the technology of IFRAME shim under Mac OSX, we may be able to implement 3D perspectives in Google Earth with the latest webKit.

6. RESULTS AND CONCLUSIONS

In our prototype system we are able to demonstrate the automatic placement of videos into the three-dimensional coordinate system of Google Earth and the result is very promising. There exist some challenges that still need to be overcome, such as the sensor accuracy of our collected dataset because of weather conditions and other environmental effects. However – very significantly – most of the data can be placed well and fully automatically in our experiments. For large-scale datasets such an automatic processing is of critical importance.

Acknowledgments

This research has been funded in part by Media Development Authority (MDA) grant NRF2007IDM-IDM002-066 and we acknowledge the support of the NUS Interactive and Digital Media Institute (IDMI).

7. REFERENCES

[1] S. Arslan Ay, L. Zhang, S. H. Kim, H. Ma, and R. Zimmermann. GRVS: A Georeferenced Video Search Engine. In *MM '09: Proceeding of the 17th ACM International Conference on Multimedia*, pages 977–978, New York, NY, USA, 2009. ACM.

[2] S. Arslan Ay, R. Zimmermann, and S. H. Kim. Viewable Scene Modeling for Geospatial Video Search. In *MM '08: 16th ACM International Conference on Multimedia*, pages 309–318, 2008.

[3] Google Earth Staff. Google Earth User Guide, 2007.

[4] Google Earth Staff. Google Earth API Samples, 6 August 2009. URL: http://earth-api-samples.googlecode.com/svn/trunk/examples/bounds.html.

[5] J. King. How to cover an IE windowed control (Select Box, ActiveX Object, etc.) with a DHTML layer, 21 July 2003. URL: http://www.macridesweb.com/oltest/IframeShim.html.

[6] T. Navarrete and J. Blat. A Semantic Approach for the Indexing and Retrieval of Geo-referenced Video. In *1st International Workshop on Semantic-Enhanced Multimedia Presentation Systems (SEMPS)*, 6 December 2006.

[7] P. Ni, F. Gaarder, C. Griwodz, and P. Halvorsen. Video Streaming into Virtual Worlds: the Effects of Virtual Screen Distance and Angle on Perceived Quality. In *MM '09: 17th ACM International Conference on Multimedia*, pages 885–888, 2009.

[8] C. Reed and Google Earth Staff. KML 2.1 Reference - An OGC Best Practice, 2 May 2007.

[9] C. Reed and Google Earth Staff. CSS 3D Transforms Module Level 3, 20 March 2009. URL: http://www.w3.org/TR/css3-3d-transforms.

[10] F. Schnell. GPicSync: Automatically Geocode Pictures from your Camera and a GPS Track Log, 13 April 2009. URL: http://code.google.com/p/gpicsync/.

[11] R. Simon and P. Fröhlich. A Mobile Application Framework for the Geospatial Web. In *16th International Conference on World Wide Web (WWW)*, pages 381–390, 2007.

[12] Wikipedia. Wiki contents: Ajax and Mixed Reality, 2010. URL: http://en.wikipedia.org/wiki/.