Geotagging Software for media content based on EGNOS/GPS Location Based Service (LBS)

Angelos Anagnostopoulos[1], Anastasios Trypitsidis[1], Nikolaos Papadakis, Marc Bonazountas, Despina Kallidromitou

[1] EPSILON International SA | Monemvasias 27, 15125, Marousi, Greece | www.epsilon.gr

Abstract

This paper describes the design and implementation process adopted by Geotagging in c-Space project (http://www.c-spaceproject.eu/). This software module is utilizing different technologies, developed during two EU projects. Mainly:

- Tagging metadata to media content by c-Space project,
- GPS LBS module developed during i-locate project (http://www.i-locate.eu/),
- EGNOS module – Enhances GPS position by c-Space.

Basically, this module governs tagging video streaming or images with location information and informative metadata provided either automatically via a repository such as Wikipedia or by the users themselves.

Keywords

Geotagging; Location Based Services; EGNOS; GPS; Video stream; Images; c-Space; i-locate.

1 Introduction

One of the main objective of c-Space project is to benefit of nowadays trend - 14% of Internet users regularly upload and share media content (photos/videos) in a mobile scenarios – and deliver a low-cost solution to reconstruct 4D scenes (3D plus time) of indoor or outdoor events from casual real-world footage (e.g. captured by mobile phones) through Video Based Rendering (VBR) techniques.

In order to carry through the abovementioned objective many tasks have been defined. In our case we will focus on the development of a software module which will be used to tag media content (i.e. video stream or image) generated by end-user mobile application with meta-data. This meta-data includes spatio-temporal information and more “traditional” static values. It is either produced automatically (e.g. via smart-phone sensors and server-side logic) or is manually entered by the end-user.

The importance of this module is vital for the project because it intends to gather spatial information from mobile devices and store within a video. Afterwards, our intention is to create a standard, as it already stands for

photos.
From a literature standpoint, there are not many references about geotagging of videos – enhance videostreams with spatial information (lat, long, elevation, time). There are very few examples regarding geotaging of videos and in most cases they are just storing a single location as metadata for searching purposes (i.e. geo-referenced queries – range), as can be found (Seo, Cui & Zimmermann, 2012). There are also some interesting thoughs by the Safe Software Blog (blog.safe.com), where actually they are refering to our "problem"/goal . Specifically, Harper S. indicates the importance to have a standard which will store location information in each frame of a video. Specifically, "A standard which allows us to very easily tie together individual frames with location. On a more technical level, I want to be able to programmatically extract the latitude and longitude for any frame in the video. If we had that then it would allow us to do some amazing things" (Harper, 2012). In addition, there are also some commercial solutions where outcomes are unknown (i.e. Garmin, Go Pro) and semi-automatically procedures are required.
In the following paragraphs, data is categorized into types. For each media content type, the standards selected to describe it are explained. The technical process followed is described along with the resulting API. Finally, considerations are reported and future actions are enumerated.

2 Metadata categories

It was determined that meta-data falls under two major categories: Spatio-temporal; Static. Furthermore, in terms of the different media content types generated by the end-user - mobile application, two more categories arise: Image meta-data; Video meta-data.
All four combinations of these two classifications are valid and were considered when designing the technical solutions.

2.1 Spatio-temporal

From project's perspective, spatio-temporal metadata is the most important as it contains all meta-data describing the media content location in space and time. This information is critical for the project since the 4D (3D models + time) content reconstruction is relying on it.
Taking the above into consideration we determine that at minimum the following must be stored in a media file:

1. Spatial meta-data (at the time of shooting)
    a. Location: The camera location, expressed using World Geodetic System coordinates, including elevation (measured in meters).
    b. Bearing: The camera bearing (where it's pointing to), expressed in degrees.
    c. Speed (optional): The camera speed, expressed in km/h.
2. Temporal meta-data
    a. Time: The camera time, measured in millisecond accuracy and expressed in Greenwich Mean Time (GMT/UTC). Special considerations regarding the time are detailed in the following document section.

Temporal Considerations:
During the design of the software module, we came across with specific requirements from the 4D reconstruction engine. Specifically, the media content will come from different devices (smartphone, tablets, etc.), which will definitely not have fully synchronized internal clocks between them. This immediately creates a problem when trying to synchronize media on the server and especially in the 4D reconstruction engine.
The issue was successfully faced by querying a centralized time server (European NTP Pool) and getting back both a common time, but more importantly, the time offset between the device's internal clock and the common time. That offset is then stored into the media, and taken into account during meta-data extraction.

2.2 Static

For this set of meta-data we refer to information that apparently does not change over time/space and characterizes and media content. Basically part of this meta-data is automatically added to any media file by the device whist another part can be manually added by the user.
Specifically, automatically added values include:

- Image content: File type (i.e. JPG in the c-Space scope); File size; Image resolution (width and length); Bits per sample; Colour space;
- Video content: File type (i.e. MP4 in the c-Space scope); File size; Frame resolution (width and length); Number of frames; Compression format (e.g. H.264)

Manually added values include

- Image content: Title/description; Author/creator/artist; Document name; Copyright; Maker note; Keywords; User comment
- Video content: Title/description; Author/creator/artist; Keywords; Album info; Copyright; Rating.

3 Standards-Compliance

3.1 EXIF Standard

Well known standards were used for storing meta-data in media content as for image content EXIF Standard, as can be found (Huggel, 2014). The Exchangeable image file format is an established, commonly used, standard that specifies meta-data for images and sound. It is based on tags, following a specific structure, borrowed from TIFF files.
Various tools exist for quick EXIF data inspection on any given file, such as the free ExifTool tool, available for Linux, Windows and MacOS.

3.2 ISO base media file format (ISO/IEC 14496-12)

The primary reason for selecting the ISO/IEC 14496-12 standard was its capability of containing time-based media file formats such as video. Specifically, it is able to specify the structure and use the ISO base media file format (ISO, 2012).
In addition, the file structure is object-oriented; and as a consequence “a file can be decomposed into constituent objects (or “boxes”) very simply, and the

structure of the objects inferred directly from their type". The standard is extended to cover the MP4 and 3GP container formats, used by Android video capture (and, consequently, in the c-Space context), based on International Organization for Standardization (2013).

4 technical Implementation

The technical implementation presents solutions used the standards and basically it consisted of two parts: the client – Android and the server side implementation.
For the Android side, research was performed in order to locate open-source software libraries and/or tools fit to cover the project's needs. The final result is an API style set of code (including usage example mini applications) which can be included in any third-party Android application, such as in our development case. For the server side, an HTTP REST web service was developed, responsible for meta-data extraction and/or enhancement after uploading the content to the server.

4.1 Metadatalibrary

In this subsection, the main Android classes and methods are described. The figure below illustrates a step by step tagging process activity, both for images and video.

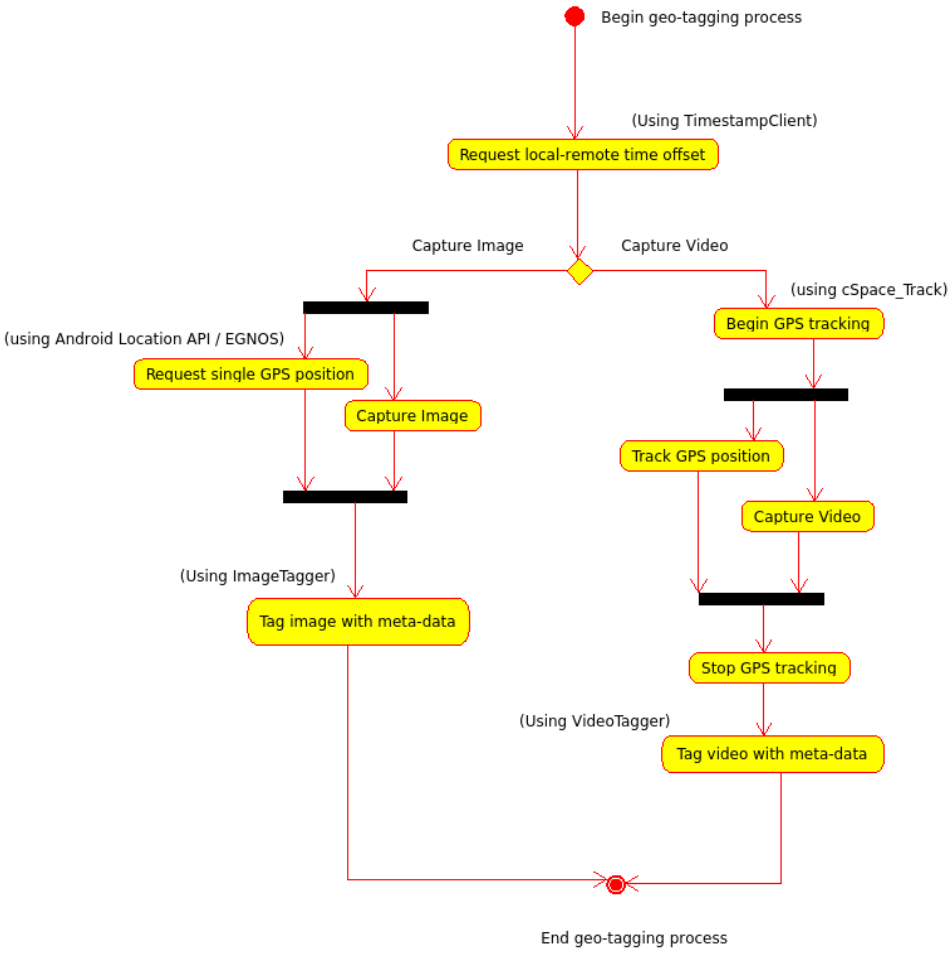

Figure 1: Geo-tagging process activity diagram.

4.2 The TimestampClient class

This class is used to access the remote Europe NTP Pool (via the internet) and fetch any remote time related data. Mainly, it queries:

- the europe.pool.ntp.org time server and returns it's time in UTC format (dd/MM/yyyy HH:mm:ss.SSS z), in millisecond accuracy,
- Queries the 'europe.pool.ntp.org' time server and returns the offset (in milliseconds) between the local device internal clock and the NTP server.

4.3 The ImageTagger class

At first and for image tagging, we built into the Android SDK the ExifInterface (Android, 2015). Since, it provides access to a limited subset of the EXIF tag set; therefore, we build upon the Android Exif Extended library (Crungola, 2014). At the end an Android API was developed providing easy to use methods, via the ImageTagger class, for setting values to specific tags. The methods provided along with the parameters and their descriptions, include:

- setTitle: A text title that sets the image description.
- setAuthor: A text that sets the image author/creator.
- setLocation: Coordinates (in WGS84) which sets camera’s location during image capture.
- setBearing: Sets the camera bearing, in degrees.
- setSpeed: Sets the device's speed at the time of capture.
- setDateTimeCaptured: sets the date and time (in millisecond accuracy) when the image was captured. See section 'Temporal Considerations' for more details on how to set a proper value.
- setTimeOffset: The time offset, measured in milliseconds

4.4 The VideoTagger class

For the video content tagging we used the open-source mp4parser library, which allows full control over the MP4 container format, including retrieving/setting meta-data values (or boxes, as they are called in the ISO/IEC 14496-12 standard) (ISO, 2012; Annies, 2014). Build upon this library and choosing specific tags from the full set (pertaining project needs), the VideoTagger class contains the following methods:

- setTitle: Sets the video title/description.
- setAuthor: Sets the video author/creator.
- setKeywords: Sets an array of keywords related to the video
- setLocation: Sets a "general" location describing the entire video. Not suitable for obtaining tracking info. For this reason we are using also 3GPP technical specifications (3GPP, 2005)
- setTrackingData: Parses a GPX GPS/EGNOS tracking file and embeds it into the video.
- setTrackingData: Parses a GPX GPS/EGNOS tracking file and embeds it into the video. See the relevant section for a detailed explanation of GPS/EGNOS meta-data tracking.
- setTimeOffset: Sets the time offset between the Android device's internal clock and the online Europe NTP time server, as described in

section “Temporal Considerations”.

4.5 Tracker Library

The task of tagging video content is more complicated than an image. For images is simply a matter of adding a simple set of coordinates and date/time info at the time of capture.
For video content, however, the device position is very likely to change during capture (consider, for example, a UAV drone recording a concert). Positional tracking must, therefore, take place throughout the entire duration of the video recording, in order to provide an adequate quantity of information.
Since the MP4 video container (or any other container, for that matter) does not provide a built-in mechanism for recording & storing location tracking data (aside for a single 'location' set of coordinates, as described previously) a custom solution have been implemented.
In respect to the GPS/EGNOS tracking mechanism implemented, it was based on the open-source (licensed GPLv3) library Open GPS Tracker (Open GPS Tracker, 2013). This library offers an Android Service component able to track and store GPS data in real-time, including any device sensor data (such as speed and bearing). The service runs in the background, so any other Android activity can take place at the same time, in our case a recording video.
The stored data is saved in XML following the GPX Exchange Format Schema, a common GPS data format for software applications. GPX allows for easy reviewing of captured data, via a large number of available GPX viewing applications.
Open GPS Tracker captures GPS data using Android LocationProvider. Additionally, we integrated with the EGNOS module (wherever available and applicable) an Android MockLocation, that was implemented and registered with the LocationProvider.
The resulting GPX file can then be easily embedded into a video file, by using the suitable VideoTagger method, as described previously.
At the end, a specific, re-factored, version of the library was created, cSpace_Tracker, which can be included in any other Android project (along with its dependencies). An excerpt from a GPX file is as follows:

```
<metadata>
      <time>2014-10-09T08:05:32Z</time>
</metadata>
<trk>
      <name>Track 2014-10-09 11:05</name>
      <trkseg>
            <trkpt lat="37.979397447779775" lon="23.783715711906552" >
                  <ele>256.29998779296875</ele>
                  <time>2014-10-09T08:05:38Z</time>
                  <extensions>
                        <ogt10:accuracy>5.0</ogt10:accuracy>
                        <gpx10:course>42.0</gpx10:course>
                  </extensions>
            </trkpt>
            …......
      </trkseq>
</trk>
```

4.6 Workflow for third-party consumers

The steps below summarize the workflow for creating a video recording Android application. The application initiates the GPS/EGNOS tracking service when the user starts the application. The application requests the Europe NTP time offset and stores it along with the current local device timestamp. Then, the user through the application begins recording a video and after while stops recording. Thereafter, the application deactivates the GPS/EGNOS tracking service application and it exports the captured tracking data to a GPX file. At the end, the application embeds the GPC file to the captured video, along with any other metadata desired.

Results

The following figures illustrate a sample application which we developed for testing reasons. The first figure depicts the interface of the application. We can either start the tracking service using GPS or use the EGNOS. The second figure illustrates the status of the receiver after the mobile device has been connected to a receiver.

Finally, the last figure depicts the tracker result, namely the GPX file. The GPX file is visualized through a well-known application (GPX Viewer).

Figure 2: Tracker application.

Figure 3: Tracker application-2.

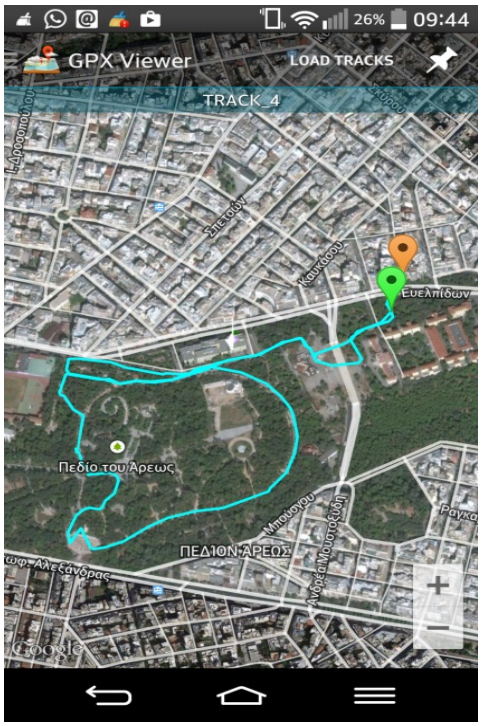

Figure 4: Tracker result - GPX file.

5 GPS & EGNOS Technical Specifications

As indicated the application supports both GPS and EGNOS as the reference location based service.

5.1 GPS APIs

With respect to the GPS LBS we developed a native Android library that retrieves the required location information from the Android OS.
Additionally, the system provided the functionality for a single location update, from the GPS (if available), or alternatively from the Network provider, and to notify the given instance. For this purpose, a locationListener “parameter” instance is used notifying on location updates. A LocationProvidersUnavailableException is thrown if neither one of the GPS/NETWORK nor other Location providers are available.
Below are the most important APIs:

- The getLocationByProvider endpoint is used to obtain last known location from a specific provider (network/gps). The location information is returned as a String object.
- The getLastKnownBestLocation endpoint is used to obtain the “best” last known location selected from all providers. A threshold in milliseconds is used (defaults to 5000 ms) to determine whether a Location is to be considered old. The endpoint returns a Location instance. The following assumptions have to be made: a location update is considered 'old' if its older than the configured; GPS is preferable than network location; If GPS data is old it cannot be considered reliable and the system will opt for network location; If both data are old then the system should return the newest of those two.

5.2 EGNOS APIs

We based the EGNOS integration on the EGNOS SDK. EGNOS SDK is a complex tool which basically comprises a plurality of APIs. That module is, in essence, a 'wrapper' around the EGNOS core source code, hiding any “unnecessary” EGNOS complexity from the module consumer, whist combining at the same time the features of the various EGNOS toolkits, as they become available.
In order for the SDK core algorithms to apply corrections to the EGNOS signals received from satellites or SISNeT, pseudorange/ephemeris GPS input data is required. At the moment, smartphone GPS chipsets (as well as the Android Location API) do not provide such information. It can only be obtained using an external GNSS chipset. Thereunder, basic API’s will be presented.

- The getReceiverType endpoint is used to provide the type of the receiver that the mobile is connected. This class returns the connected receiver type
- The getEDAS function informed if EDAS service is on or off. EDAS is the technical core of the EGNOS Commercial Data Distribution Service (CDDS) and provides the opportunity to deliver EGNOS data to users who cannot always view the EGNOS satellites (such as in urban canyons) or to support a variety of other value added services, applications and research programs. This class returns 0 or 1 based on EDAS ON or OFF from settings.
- The getPosition function computes the position as a table of 7 x 1 values. Rows 0 to 9 as GPS Latitude, GPS Longitude, GPS Altitude, EGNOS Latitude, EGNOS Longitude, EGNOS Altitude, HPL, R&D Latitude, R&D Longitude and R&D altitude respectively. The class returns the current position.
- The getinitialEGNOSPosition function computes the initial position acquired. The position in the Earth-Centered, Earth-Fixed (ECEF). Used as a first estimation to obtain a position. This class returns the initial EGNOS position.
- The getCurrentLocation function: sets mode of data source for EGNOS position through EGNOS Signal in Space or SISNeT; Checks for network in case data source is SISNeT; Gets the GPS Latitude, GPS Longitude, GPS Altitude; EGNOS Latitude, EGNOS Longitude, EGNOS Altitude and HPL from the EGNOS SDK. This class returns GPS and EGNOS position from EGNOS SDK.
- The checkNetwork function checks if the mobile device is connected to a network via 3G or Wi-Fi. This class returns “0” if no network is available, “1” if it is available.

6 Future steps

Next steps will include enhancing the meta-data library, to include additional meta-data tags/fields, as needs multiply. We are really interested to integrate also with the upcoming Galileo Open Service (OS), when it will be released. Meta-data extraction on the server side will be fully implemented. Finally, if the need arises, more sources will be added to the automatic data extraction.

References

- ✔ 3GPP TS 26.244 V6.4.0 , (2005), "Technical Specifications: 3rd Generation Partnership Project", Retrieved from: http://www.qtc.jp/3GPP/Specs/26244-640.pdf
- ✔ Android, (2015). ExifInterface", Retrieved from: http://developer.android.com/reference/android/media/ExifInterface.html
- ✔ Annies, S. (2014). "mp4parser on GitHub", Retrieved from: https://github.com/sannies/mp4parser
- ✔ Crungola, A. (2014). "Android Exif Extended on GitHub", Retrieved from: https://github.com/sephiroth74/Android-Exif-Extended
- ✔ Harper, S. (2012). "GPS Meets Video. Do We Need a Standard for Geotagging Videos?", Retrieved from: http://blog.safe.com/2012/04/gps-meets-video-do-we-need-a-standard-for-geotagging-videos/
- ✔ Huggel, A. (2014). "Exiv2 Image metadata library and tools", Retrieved from: http://exiv2.org/tags.html
- ✔ International Organization for Standardization, (2012), "Information technology -- Coding of audio-visual objects -- Part 12: ISO base media file format", Retrieved from: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=61988
- ✔ International Organization for Standardization/IEC, (2013), "Sustainability of Digital Formats. Planning for Library of Congress Collections", Retrieved from: http://www.digitalpreservation.gov/formats/fdd/fdd000079.shtml
- ✔ Open GPS Tracker, (2013). "Tracker, Open GPS by Google Code", Retrieved from: https://code.google.com/p/open-gpstracker/
- ✔ Seo, B., Cui, W., & Zimmermann, R. (2012). "An experimental study of video uploading from mobile devices with HTTP streaming", Retrieved from: http://dl.acm.org/citation.cfm?doid=2155555.2155589