Analysis, recovery and visualization of encoded Google Fit geo-location forensic artefacts from an Android Wear smartwatch

G. Seba

Cyber Security Technical Consultant, QA Ltd.

George.Seba@qa.com

Abstract

Recovery and interpretation of geo-location data stored on Android wearable devices such as smartwatches is an emerging field of contemporary mobile forensics. This data can include not just GPS coordinates such as latitude, longitude and altitude, but also precise timestamps associated with specific geographical coordinates. Given the growing use of fitness tracking apps on smartwatches that uses the embedded GPS sensor of the device, this data can be very useful in a forensic investigation.

This paper summarises the findings of advanced research into how Google Fit structures geo-location data, where it is stored on the smartwatch, how it is encoded and how synchronization and interaction with a paired phone impacts on the storage of geo-location data on the wearable device. The research has found that deleting all Google Fit history does not delete the GPS coordinates stored on the smartwatch. By the contrary, new records containing geo-location data are appended to the list of old, supposedly deleted records.

Furthermore, we present a method of automating the extraction and decoding of Google Fit geo-location data from an Android smartwatch that generates a KML file, which can be further opened with Google Earth in order to visualise the GPS coordinates as routes with associated timestamps.

1. Introduction

The digital revolution we have witnessed in the last decade alone has enabled the use of personal computing on a global scale, largely supported by the extremely rapid rate of adoption by users of portable digital devices starting with MP3 players then smartphones and currently wearable technology such as smartwatches or fitness trackers. Although this has brought largely positive benefits to the users, there is always a minority that uses this technology to perpetrate crimes and traditionally the emphasis of digital mobile forensics seemed to be on the examination of mobile smartphones due to the fact that this technology is the most ubiquitous. Currently, Mac-based and Android digital portable devices are dominating the market for these technologies and there are a considerable number of forensic tools available to the examiners to investigate these devices. The arrival of smartwatches only two or three years ago resulted in a rapidly growing rate of adoption of these devices which is in stark contrast with the rate of development of specialist forensic tools and software dedicated to investigate smartwatches. As a result, the availability of forensic tools or scientific research into smartwatch forensics is currently very limited.

In 2014 Android Wear has been released as an extension of the Android Operating System with the aim to support the development of a new generation of portable technology – Android smartwatches. The wide range of sensors available in these smartwatches has enabled developers to release fitness tracking applications such as Google Fit. The benefit to the user is that Google Fit enables the smartwatch to be used as a step counter, to track a running session on a map or count the calories burned in a sport activity. From a forensic perspective, the potential benefits to an investigator include the ability to identify unique GPS coordinates recorded by the smartwatch in a Google Fit activity, or build up a timeline based on the unique timestamps of each GPS reading. Furthermore, the Google Fit geo-location data can be used forensically to calculate the speed, distance and direction of travel or even derive whether the user was walking, running or using an additional method of vehicle transport.

This paper will look at related work in Android Wear forensics and also at the background of geo-location forensics in traditional satellite navigation systems and more recently in smartphones, before setting out a methodology of experimental research to determine the location where geo-location data recorded by Google Fit is stored on the smartwatch, under what conditions is recorded, determine the structure of this data, how it is encoded and the byte-offsets for each coordinate before finally presenting a method of automating the extraction of this geo-location data from a Google Fit database with the aim to visualise it further in popular mapping software such as Google Earth. The results of the practical experiments will be presented and discussed in detail with the view to critically evaluate their importance from a forensic perspective after which the main conclusions of the research will be drawn along with a discussion of the areas that could benefit from further research as a result of the findings presented in this paper.

2. Background and Related Work

Geo-location forensics focusing on the examination of traditional commercial GPS satellite navigation systems is an established filed of digital forensics mainly due to the increased popularity of these devices with the general population in the last decade alone. Forensically, the importance of evidence recovered from GPS devices cannot be overstated since it can reveal important information to a forensic investigator, such as very accurate geographical positioning of the device or a route taken by the user of the device. Since the Global Positioning System uses very accurate atomic clock signals to triangulate the geographical position of a GPS device (Lombardi et al., 2001), each GPS coordinate recorded by the device has a unique timestamp associated with it. This is a very useful artefact to have in a forensic investigation because it could be used to build an accurate timeline from multiple GPS readings over a certain time period, hence being able to determine an accurate position of the device and also an accurate time when the device has been present at a certain location (Last, 2009).

The most common forensic artefacts that could be recovered from satellite navigation systems consists of trackpoints, track logs, waypoints and routes, with trackpoints representing one of the most important artefacts (LeMere, 2011) because it consists of a single accurate GPS reading (latitude, longitude, altitude) and a timestamp just as accurate that are both recorded by the device every few seconds. A forensic investigator could then use trackpoints recovered from a satellite navigation device to build the route taken by the device and the exact times when the device has been at a certain location on that route. While trackpoints and track logs are usually generated automatically by the system, it is the user that creates the waypoints and routes. This is very important to the forensic investigator because the trackpoints and the track logs can show where the device has been. In contrast, the waypoints and the routes can only show the location where the user intended to navigate to (ibid.).

2.1 Geo-location forensics on Android

The advent of smartphones with embedded GPS sensors and the ability to install third-party applications on these mobile devices resulted in navigation apps becoming increasingly popular because they enabled the smartphone to be used also as a satellite navigation system. Forensically, this is important because now a single device can contain many other geo-location artefacts associated with the device, such as pictures EXIF metadata containing GPS

coordinates of the location where the picture has been taken with the smartphone's camera or the location from where a Facebook post has been made. A noticeable difference in the structure and storage of geo-location data on Android phones as opposed to traditional satellite navigation systems has been observed (Roeloffs et al., 2014), with many applications now storing records of GPS readings in SQLite databases.
In the absence of a GPS sensor on the smartphone or when the GPS sensor is switched off, triangulating the relative distance and signal strength between the device and surrounding mobile cell towers or Wi-Fi hotspots can still estimate the location of the device (Kramer, 2013). This information is stored in two files – *cache.cell* and *cache.wifi* in the */data/data/com.google.android.location/files/* directory (Spreitzenbarth et al., 2012). Apple uses a similar approach to provide location-based services to iPhones when the GPS sensor is switched off and the data containing the estimated coordinates can be found in the *consolidated.db* SQLite database file inside the phone's iTunes backup (Sun, 2012). However, since these are just estimated readings with no exact position of the device at a specific timestamp, they are lacking accuracy. This could make a very important difference in a forensic investigation because the estimation process of determining the approximate location of the device could weaken the evidential value of the artefacts when presented in court, for example.

2.2 Android Wear forensics

One of the most important developments in the Android platform was the introduction of Android Wear API, an extension to the Android 5.0 Lollipop operating system which was released by Google in March 2014 (Pichai, 2014). Wearable devices and in particular smartwatches were designed primarily to extend the functionality of applications with the view to increase efficiency of user interaction by developing a GUI purposefully designed for ease-of-use and convenience (Cuartielles-Ruiz & Goransson, 2015). The versatility of Android Wear is provided by the combination of a wide range of sensors available in the hardware with the software designed to maximise the use of these sensors and offered to users as apps that address a variety of functionalities from location-specific weather information to fitness trackers and navigation directions. A typical smartwatch features on average 4 or 5 different sensors and some devices have even more than that, although the most common sensors are the accelerometer, pedometer, gyroscope, compass and heart-rate monitor.
Despite the fairly recent launch of Android Wear in March 2014, the popularity and growing rate of adoption of smartwatches among users in subsequent months combined with elevated features and functionality brought by wearable technology, has increased the significance of these devices to a forensic investigation. However, the research into Android Wear forensics is relatively limited to a very small number of initial forensic examinations of Android smartwatches, most of them attempting to establish forensic acquisition methodologies adapted from the more general and well-documented field of Android smartphones forensics. Although limited, initial Android Wear digital forensic research by Wilson (2015) had established a methodology of imaging the *system*, *cache* and *data* partitions from a Sony SW3 smartwatch device using the Linux *dd* utility, nevertheless the methodology proposed assumes that the smartwatch is already rooted and does not address the situations when the device's bootloader is locked.

Other findings suggest that all data on the *data* partition is stored in sqlite3 database files containing the traditional shared preference folder *shared_prefs* together with a *config* file that is used by all the applications on the device although this needs further research to establish the detailed process. Additionally, a similar initial research of Android Wear forensics on a Sony SW3 smartwatch found that information about the paired smartphone and Bluetooth data such as the MAC address and the device ID is stored in the */data/misc* folder (Parikh et al., 2015). Furthermore, the authors state that voice commands are stored locally on the wearable device inside the *com.google.android.gms* folder in the */data/data* directory as text strings inside files named in the format *voice_action_xxxxx*, however, this needs further research as it seems that at a closer inspection the voice commands identified are the default Android Wear commands needed to initialise the cue cards.

2.3 Google Fit

Google Fit is a fitness tracking application which uses the device's sensors to monitor various fitness and health activities such as walking, running or cycling while providing a comprehensive range of fitness and health statistics, such as calories burned and steps count, weight tracking or distance and routes history. Google Fit is the response to Apple Health, the iWatch app with similar functionality, and currently it comes installed by default in all Android-powered devices featuring Android 5.0 Lollipop or higher. The strength of this application lies in the ability to be run from the paired smartwatch, although the settings and the main interface reside on the paired phone on which it had been installed. At first launch, Google Fit needs to be configured by providing the gender, height and weight then it will link this information with the Google user account with which the phone has been configured. If an Android Wear smartwatch is paired with the phone on which Google Fit is configured, then app notifications will be sent to the smartwatch such as the total steps taken in a time period or the calories burned in a certain day, etc. The central feature of the application is the *Activity*, a session-based workout choice with options from walking, running, cycling or other fitness activities such as push-ups.

From a forensic perspective, being able to access geo-location information regarding the route taken during a fitness session is likely to be very important in a forensic investigation, especially since GPS data contains precise dates and times associated with every single geographical coordinate recorded. However, due to the small size of the smartwatch's screen, the map showing the route taken during the activity is not available as a statistic on the smartwatch and it can be viewed only on the paired phone on which the main Google Fit app is installed, even though the route could be recorded using the smartwatch's own GPS sensor. This allows hypothesising that since the smartwatch synchronises its geo-location data with the paired phone, then in a forensic context it would be beneficial to investigate further the smartwatch because it is likely that even though it is not shown on a map, geo-location data is stored somewhere on the smartwatch.

3. Methodology

The main aim of the research was to determine what, if any, Google Fit geo-location artefacts can be identified, recovered and visualised from an Android Wear smartwatch and under what conditions is this data stored locally on the smartwatch. The methodological approach for the research has been a combination of a theoretical analysis of the Google Fit API followed by conducting practical experiments in a controlled testing environment. First, the Google Fit API documentation has been analysed in order to determine how and what geo-location data is stored by the application, and also to examine the database used by the application in order to establish the structure and format of the data. Then a range of experiments have been performed in a strictly controlled environment in order to observe how the theory is put into practice by determining how the process of storing and structuring geo-location data on the smartwatch actually works in realistic situations.

The practical experiments consisted of starting a Google Fit activity and following a pre-determined route for a set period of time, at the end of which the smartwatch had been imaged and the Google Fit database had been extracted and analysed. Every test has been carried out on a different route to avoid GPS coordinates being misinterpreted. A GPS control dataset has been generated and recorded at the beginning and end of each experiment using the GPS debug console in the Developer Options on the Android Wear smartwatch.

3.1 Experimental tools

The chosen device to be used in the experimental phase was a Sony SWR50 smartwatch and this decision was grounded in the availability of a wide range of sensors on the smartwatch, including accelerometer, compass, pedometer, magnetometer and gyroscope. It was reasoned that having more sensors would increase the level of accuracy in generating control and real geo-location data used in the testing scenarios as part of the experiments. Furthermore, Sony SWR50 was the only smartwatch at the time when the project started that featured a stand-alone GPS sensor capable of operating independently from the paired phone, which made this device ideal for use in the experiments.

Several smartphone models were considered for pairing with the smartwatch, and the chosen phone to be used in the experiments was LG Nexus 4, which has been developed by Google and manufactured by LG. The justification for choosing LG Nexus 4 was that it features the stock version of Android OS as designed by Google, which does not include any vendor-specific software like most of the other models of mobile phones available. It was considered that using the stock version of Android eliminates the risk of vendor-specific software influencing the functionality of applications, which in turn might compromise the integrity and validity of test results. The phone featured Android OS version 5.1.1 Lollipop, build number LMY48T and the version of Android Wear installed was 1.3.0.2160025 with Google Play Services version 8.7.01 (2590918-534).

3.2 Experimental setup

The smartwatch and the phone have been reset to initial factory settings before each experiment to prevent potential interference from previous tests compromising the integrity of results. It has been decided to not use the smartwatch's Wi-Fi connectivity feature during the experiments because of the effects that this action could potentially have on the data synchronization process. That would involve having to monitor a three-way sync between the devices themselves and separately from each device to and from the Google Fitness Store, which is a cloud-based service enabling fitness information such as geo-location data to be stored and retrieved to and from Google's servers. Due to the potential complexity this could bring to the project, it was reasoned that using the Wi-Fi connection on the smartwatch during experiments would cause the project to deviate from its original aim so it was decided to not use the feature at all. This also made the testing environment more reliable due to having fewer variables to monitor.

The paired phone also had the Wi-Fi connection disabled during the experiments to ensure that Google Fit will not synchronise any data with the Google Fit Store. The only exception to this has been the experiment that explored the effects of deleting all Google Fit history because this required the Google Fit app to synchronise the deletion process with the Google Fit Store in order to observe whether the deletion of geolocation data is synchronised with the paired smartwatch. Similarly, the phone did not have a SIM card inserted, so no mobile internet data connections on the paired phone were active during the experiments. In order to avoid collusion between datasets gathered from two different tests, each experimental setup for tests that required a factory reset of the devices had the phone being configured using a completely new Google user account.

First it had to be established the default configuration of Google Fit database on the smartwatch before the GPS sensor or the Google Fit application being used for the first time on the smartwatch. Once extracted and analysed, this database acted as control and formed the baseline with which other instances of the database have been compared during experimentation. Following each experiment, the SQLite database has been extracted from the smartwatch or the phone and analysed. The focus was on identifying whether geo-location data is recorded in the database and if the records were consistent with the experiments conducted, especially the time and the control GPS coordinates.

4. Results

This section presents the findings of the analysis and experiments carried out and in the first part it will focus on presenting the key findings from the examination of the Google Fit API and the SQLite database. In the second part, the outcomes of the experiments are presented and analysed, before ending with discussing the recovery and visualisation of geo-location forensic artefacts in the last part.

4.1 Examination of Google Fit API

The Google Fit API is publicly available for Android developers at https://developers.google.com/fit/android/ and it details in depth the structure and processes involved in running the application. The examination of the API documentation focused on determining what specific geo-location data does Google Fit record, how that data is structured and where it is stored on the device. The analysis of the API established that Google Fit is accessing an instantaneous reading from one or more device sensors using *data points*, which is another name for GPS trackpoints found in traditional satellite navigation systems. For example, each reading from the smartwatch's GPS sensor is stored in a single data point. The value of each data point is formatted and stored programmatically in specific *data types* defined over the `com.google` namespace. Apart from instantaneous readings, data points could also represent an aggregate with statistics over a time interval, such as the total number of calories burned during a time interval or the number of steps taken, however, this is beyond the scope of the current project. A data point also contains the timestamp information for each instantaneous reading and the values for the fields of a specific data type. A data type can have more than one field and the data points containing values of instantaneous readings from the GPS sensor (i.e. user's current location) are stored in the `com.google.location.sample` data type, which contains four fields:

- Latitude – in degrees
- Longitude – in degrees
- Altitude – in meters
- Accuracy – in meters

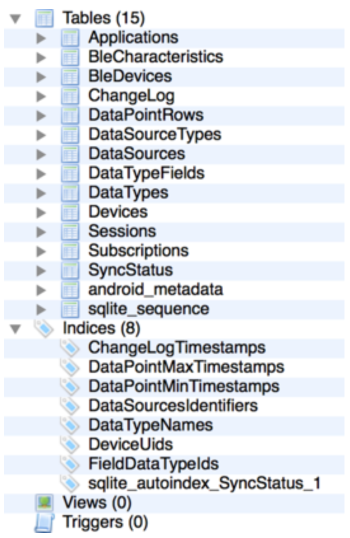
Programmatically, the Google Fit data model is defined over the specific `com.google.android.gms` namespace (Google, 2015c), which is dedicated for applications developed by Google as part of Google Mobile Services (GMS) and following the analysis of multiple disk images acquired during the experimentation stage, it has been found that the namespace is mirrored on physical storage where `/data/data/com.google.android.gms` is a physical directory containing application data and databases for Google-designed applications, including the Google Fit SQLite database.

4.2 Analysis of Google Fit database

The analysis has been conducted initially on the default Google Fit database, which was extracted immediately after Google Fit application has been launched but before any Google Fit activities were carried out. This database was used as control because it was generated automatically by the application as a result of launching Google Fit and it was not populated with any geo-location data. Subsequently, after completing a Google Fit walking activity with the smartwatch GPS sensor turned ON, the smartwatch has been imaged then the database was extracted using OSForensics v3.3 and analysed with DB Browser for SQLite v3.8.0 in order to identify any records of geo-location data.
Following the analysis of the databases extracted at the end of each designed experiment, it has been observed that the name of the Google Fit SQLite database file follows a specific pattern, which starts with "`fitness.db`", followed by the username of the Google account used in the initial device set-up, then followed by an underscore which replaces the @ symbol, and finally followed by "`gmail.com`". The whole pattern of the Google Fit database filename is in the format:

`fitness.db.` + *Google Account Username* + `_gmail.com`

The database used in the examination was the default database created automatically by the Google Fit app when it was launched for the very first time. Initially the smartwatch and the phone have been reset to factory settings then the phone was set-up with *test.forensic001@gmail.com* as the main Google account username. The smartwatch was then paired with the phone via the standard Android Wear app and the Google Fit app on the phone was launched. Immediately after this, the smartwatch has been imaged then the `fitness.db.test.forensic001_gmail.com` SQLite database file has been extracted from `/data/com.google.android.gms/databases` directory. From the CREATE DDL statements, it was possible to establish the structure of the database schema, the name of the tables, the fields of each table, together with the data type of each field and the fields that reference other tables, also known as foreign keys.

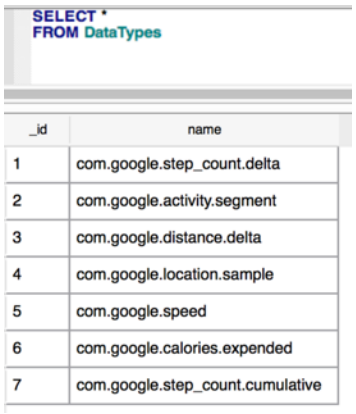


```
Tables (15)
    Applications
    BleCharacteristics
    BleDevices
    ChangeLog
    DataPointRows
    DataSourceTypes
    DataSources
    DataTypeFields
    DataTypes
    Devices
    Sessions
    Subscriptions
    SyncStatus
    android_metadata
    sqlite_sequence
Indices (8)
    ChangeLogTimestamps
    DataPointMaxTimestamps
    DataPointMinTimestamps
    DataSourcesIdentifiers
    DataTypeNames
    DeviceUids
    FieldDataTypeIds
    sqlite_autoindex_SyncStatus_1
Views (0)
Triggers (0)
```

Figure 1 - List of all tables in the default Google Fit database schema

As shown in Figure 1, the default Google Fit database contains 15 tables, however, it was identified that only 6 tables are involved in managing geo-location data and other related information, such as the ID of the wearable device or the sources of data used by the Google Fit application such as the GPS sensor on the smartwatch and the database tables are as follows:

DataTypes
The DataTypes table shown in Figure 2 contains two fields: *_id* and *name*.
The *name* field describes the various data types used by the application and a closer look at record *_id*=4 show that the name of the data type is `com.google.location.sample`, which is consistent with the findings of the Google Fit API analysis as detailed previously in section 4.1

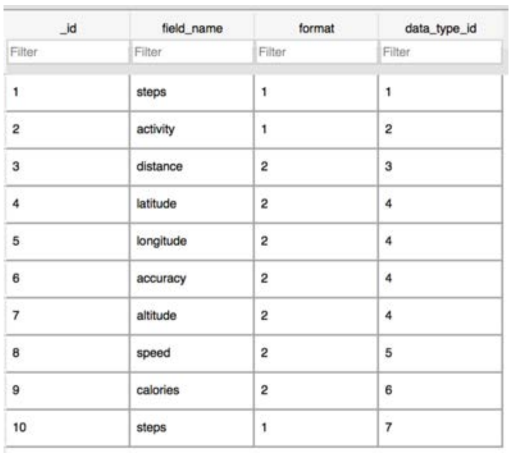


```
SELECT *
FROM DataTypes
```

_id	name
1	com.google.step_count.delta
2	com.google.activity.segment
3	com.google.distance.delta
4	com.google.location.sample
5	com.google.speed
6	com.google.calories.expended
7	com.google.step_count.cumulative

Figure 2 - Screenshot of DataTypes table

DataTypeFields
This table has 4 fields: *_id*, *field_name*, *format* and *data_type_id*. The DDL statement showed that the *data_type_id* field is a foreign key referencing the *_id* field in the DataTypes table, meaning that each value present in the field *data_type_field* must also be present in the *_id* field of the DataTypes table, hence creating a relationship between the two tables. The *field_name* attribute describes the names of the fields belonging to each data type, as shown in Figure 3 below, which displays the result of the query performed on the table. Since the *format* field is not a foreign key, it is difficult to determine without further experimentation what the values contained in this field describe.

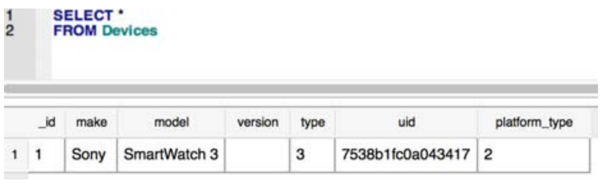


_id	field_name	format	data_type_id
Filter	Filter	Filter	Filter
1	steps	1	1
2	activity	1	2
3	distance	2	3
4	latitude	2	4
5	longitude	2	4
6	accuracy	2	4
7	altitude	2	4
8	speed	2	5
9	calories	2	6
10	steps	1	7

Figure 3 - Screenshot of the DataTypeFields table

As seen in the previous table – DataTypes – the data type with *_id*=4 is `com.google.location.sample` and because the *_id* from DataTypes is a posted foreign key into the DataTypesFields table, it can be observed that the fields corresponding to *data_type_id*=4 are **latitude**, **longitude**, **accuracy** and **altitude**, which is entirely consistent with the findings of the Google Fit API analysis.

Devices
This table lists the details of any devices using the Google Fit application. In this case is the Sony Smartwatch 3 as shown in Figure 4 below, however, if additional devices are using Google Fit under the same Google account, they will appear as new records into this table.



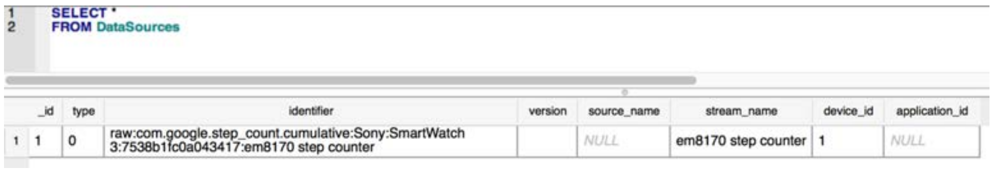
```
SELECT *
FROM Devices
```

	_id	make	model	version	type	uid	platform_type
1	1	Sony	SmartWatch 3		3	7538b1fc0a043417	2

Figure 4 - Screenshot of the Devices table

Apart from the make and model of the device, the *version* field describes the version of the device's software/hardware but this is optional. The *type* field is a reference to the type of the device and Google lists several device types such as "ChestStrap", "Phone", "Watch", "Tablet", etc. (Google, 2015b). The *uid* field contains the serial number or other unique ID for the device's hardware while the *platform_type* field is a reference to the type of the software platform in use.

DataSources
This table stores information about data streams and types of data sources used by Google Fit application, such as device sensors – accelerometer, GPS, step counter, etc.. As shown in Figure 5 below, the table contains 7 fields: *_id* as primary key, *type*, *identifier*, *version*, *source_name*, *stream_name*, *device_id* and *application_id*.



```
SELECT *
FROM DataSources
```

	_id	type	identifier	version	source_name	stream_name	device_id	application_id
1	1	0	raw:com.google.step_count.cumulative:Sony:SmartWatch 3:7538b1fc0a043417:em8170 step counter		NULL	em8170 step counter	1	NULL

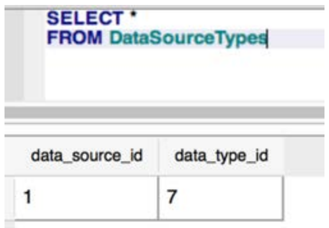
Figure 5 - Screenshot of DataSources table

The value of the *type* field indicates the type of the source of data, which can be either “raw” or “derived” (Google, 2015b), and this is described in more detail in the *identifier* field where it can be seen that in this case the data source is “raw”, meaning that it originates from a device sensor rather than being derived from calculations (Google, 2015a). As the screenshot in Figure 5 shows, the *identifier* field also describes the data type associated with the source of data, the device make, model and ID and the name and ID of the sensor from where the data stream originates.

The *version* field is empty and the NULL property of the *source_name* field also indicates that the value for this field is optional. The *stream_name* field contains the name of the data stream and as can be seen it is the name of the sensor from where the data stream originates. Finally, the *device_id* and *application_id* fields are foreign keys referencing the Device table and the Application table, respectively.

DataSourceTypes

As the name implies, the DataSourceTypes table is an intermediate table - also known as a linking table - meaning that the fields in this table are all foreign keys linking to fields in other tables and thus making it easier to associate different records from two or more tables. Each record in the DataSourceTypes table contains information about what is the data type for a specific data source and as seen in the screenshot of the table in Figure 6, the data originating from data source with _*id*=1 is of data type _*id*=7.



```
SELECT *
FROM DataSourceTypes
```

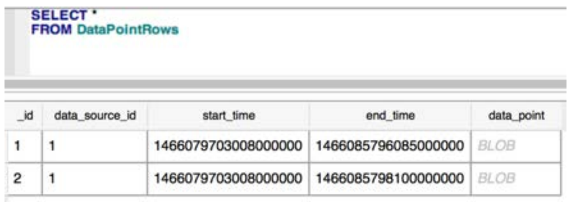
data_source_id	data_type_id
1	7

Figure 6 - Screenshot of DataSourceTypes table

A quick look at the respective records with those identifiers in the DataSources and the DataTypes tables shows that the data type of the data stream originating from the step counter is `com.google.step_count.cumulative`.

DataPointRows

The DataPointRows table contains records about data points and as the name of the table implies, each row in this table is a record of either a single instantaneous reading, which constitutes a data point as described in the Google Fit API documentation, or a cumulative reading such as the total calories burned or the number of steps taken over a time period. The _*id* is the primary key while the *data_source_id* field is a foreign key referencing the _*id* field in the DataSources table. The *start_time* and *end_time* fields as shown in Figure 33 below, store UNIX timestamps as integers representing the start and the end time of when the data point reading occurred.



```
SELECT *
FROM DataPointRows
```

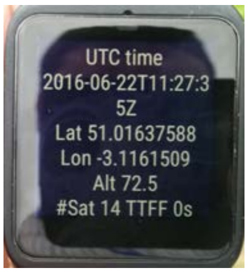
_id	data_source_id	start_time	end_time	data_point
1	1	1466079703008000000	1466085796085000000	BLOB
2	1	1466079703008000000	1466085798100000000	BLOB

Figure 7 - Screenshot of DataPointRows table

The difference in the value of the timestamps from the *end_time* field of the two records in the table is due to the high level of granularity, since the precision of the timestamp is one millionth of a second and this translates into a difference of 2 seconds between the two records. The *data_point* field in the DataPointRows table is formatted as BLOB, which means that the value stored in this field is a blob of data that is stored exactly as it was input and this field is where the raw data readings from the sensors are stored.

4.3 Analysis of geo-location data encoding

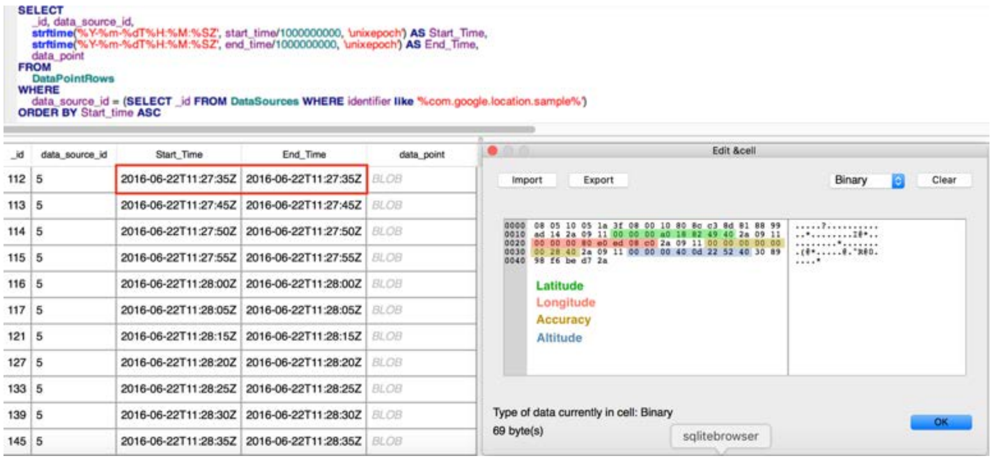
In order to perform the analysis, a dataset containing control geo-location coordinates had been generated in the experimentation stage. To ensure accuracy and validity of outcomes in the database analysis, at the start of the test the debug console on the smartwatch has been switched on to show the live stream of geo-location data generated by the smartwatch's GPS sensor that was also recorded in parallel by Google Fit and stored in the DataPointRows table. A detailed record of the Coordinated Universal Time (UTC) of the live GPS stream output on the smartwatch's debug console has been made in order to allow identification of specific table records in the DataPointRows table, as shown in Figure 8 below.



UTC time
2016-06-22T11:27:35Z
Lat 51.01637588
Lon -3.1161509
Alt 72.5
#Sat 14 TTFF 0s

Figure 8 - Screenshot of the smartwatch's debug console showing the UTC time and the live stream of GPS coordinates as were being accessed by Google Fit

Aligning a specific date and time shown in the smartwatch's live debug console with the same date and time of the data point stored in the database record, ensured that the control coordinates shown in the debug console are also stored within the binary data of the *data_point* field of that specific record. Once a record has been selected based on a specific time, the *data_point* cell has been analysed to identify the lengths and the byte offsets of each GPS coordinate (Figure 9).



```
SELECT
   _id, data_source_id,
   strftime('%Y-%m-%dT%H:%M:%SZ', start_time/1000000000, 'unixepoch') AS Start_Time,
   strftime('%Y-%m-%dT%H:%M:%SZ', end_time/1000000000, 'unixepoch') AS End_Time,
   data_point
FROM
   DataPointRows
WHERE
   data_source_id = (SELECT _id FROM DataSources WHERE identifier like '%com.google.location.sample%')
ORDER BY Start_time ASC
```

_id	data_source_id	Start_Time	End_Time	data_point
112	5	2016-06-22T11:27:35Z	2016-06-22T11:27:35Z	BLOB
113	5	2016-06-22T11:27:45Z	2016-06-22T11:27:45Z	BLOB
114	5	2016-06-22T11:27:50Z	2016-06-22T11:27:50Z	BLOB
115	5	2016-06-22T11:27:55Z	2016-06-22T11:27:55Z	BLOB
116	5	2016-06-22T11:28:00Z	2016-06-22T11:28:00Z	BLOB
117	5	2016-06-22T11:28:05Z	2016-06-22T11:28:05Z	BLOB
121	5	2016-06-22T11:28:15Z	2016-06-22T11:28:15Z	BLOB
127	5	2016-06-22T11:28:20Z	2016-06-22T11:28:20Z	BLOB
133	5	2016-06-22T11:28:25Z	2016-06-22T11:28:25Z	BLOB
139	5	2016-06-22T11:28:30Z	2016-06-22T11:28:30Z	BLOB
145	5	2016-06-22T11:28:35Z	2016-06-22T11:28:35Z	BLOB

```
0000  08 05 10 05 1a 3f 08 00 10 80 8c c3 8d 81 88 99   ......?...........
0010  ad 14 2a 09 11 00 00 00 a0 18 82 49 40 2a 09 11   ..*.........I@*..
0020  00 00 00 80 e0 ed 08 c0 2a 09 11 00 00 00 00 00   ........*.......
0030  00 28 40 2a 09 11 00 00 00 40 0d 22 52 40 30 89   .(@*.....@."R@0.
0040  98 f6 be d7 2a                                      ....*
```

Latitude
Longitude
Accuracy
Altitude

Type of data currently in cell: Binary
69 byte(s)

Figure 9 - Screenshot of the data_point cell in the DataPointRows table showing the byte offsets for each recorded GPS coordinate at exactly the same time as shown in the smartwatch's debug console.

As shown in Figure 9, the *start_time* and *end_time* values of record *_id*=112 matches the recorded date and time from the output of the smartwatch's debug console as shown previously in Figure 35. The analysis of the binary data in the *data_point* cell consisted of sequencing through consecutive records with the aim to try to identify the bytes that are changing in each subsequent record. As shown previously, the analysis of the Google Fit API already established that the longitude, latitude and altitude coordinates are stored as floats, which means that in binary these values are 8 bytes long.

It was also logically deducted that the value of a specific coordinate recorded a few seconds apart while walking a short distance (seconds as unit of time, for disambiguation), is likely to be the same in the degrees and minutes part of the binary number (minutes as subdivisions of arc degrees, for disambiguation), because the short distance of a few meters covered in a very short time of 4-5 seconds is way too small to have an impact on the value of arc degrees or arc minutes. Furthermore, since the value of the control GPS coordinates recorded at the same time as the time of the data point record analysed was already known, the next step was to convert the latitude, longitude and altitude from their known decimal float-point value to a 64bit hexadecimal representation using the IEEE Standard for Binary Floating-Point Arithmetic (IEEE Computer Society, 2008).

The end results of the geo-location data encoding analysis is presented in Table 1 below, where are listed the byte offsets for each individual coordinate identified inside the *data_point* field of the DataPointRow table, the length in bytes for each coordinate, their HEX value reversed and displayed in big endian, together with the decimal value for each coordinate following the IEEE-754 Floating-Point conversion and the corresponding value for each coordinate as it was shown in the debug console of the smartwatch during the experimentation stage. The small difference between the two values is due to the margin of error of the smartwatch's GPS sensor, since the accuracy was 12 meters, as shown in the table.

Table 1 - Table showing the byte offsets of each coordinate, the stored HEX values and a comparison between the converted decimal values and the values shown on the debug console of the smartwatch

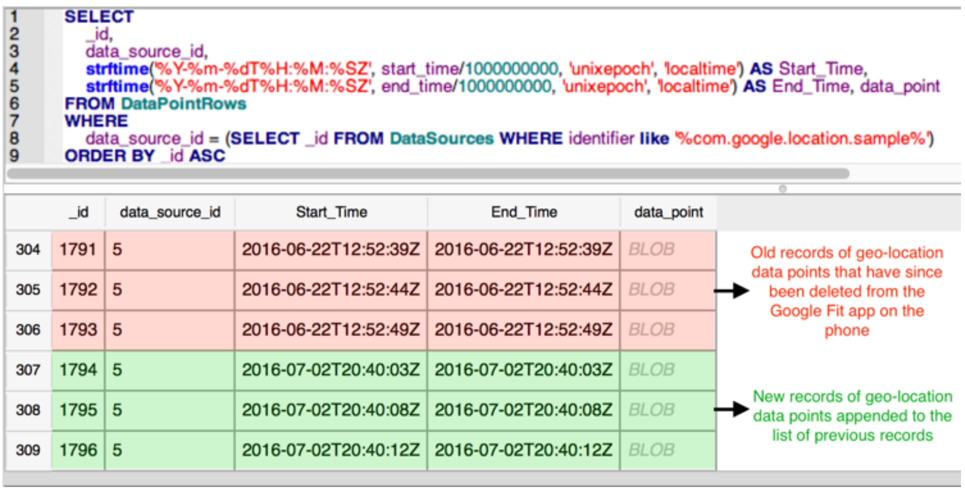
Byte Offset (decimal)	**Length** (bytes)	**Coordinate**	**HEX value** (big endian)	**Decimal value** (IEEE-754 floating point conversion)	**Decimal value** (shown on smartwatch)
21 - 28	8	Latitude	40498218A0000000	51.01637649536133	51.01637588
32 - 39	8	Longitude	C008EDE080000000	-3.1161508560180664	-3.1161509
43 - 50	8	Accuracy	4028000000000000	12	-
54 - 61	8	Altitude	4052220D40000000	72.53205871582031	72.5

4.4 Consequences of deleting all Google Fit history

This experiment was designed to mimic typical user behaviour and consisted of deleting all Google Fit history and then investigate the effects of this action on the Google Fit database. The aim of this experiment was to establish what happens to the geo-location data on the smartwatch if all Google Fit history is deleted from the paired phone, which could be very important in a forensic investigation to establish a timeline of past events and locations.
Since the Google Fit interface on the smartwatch is a simplified version of an info-display of the main app installed on the phone, the only access to the application's settings was through the main Google Fit app installed on the paired phone and the entire Google Fit history has been deleted.

Once the Google Fit application history has been deleted from the paired phone, the Wi-Fi connection on the phone has been switched-on in order to enable synchronisation of the history delete activity with the Google Fit Store, a cloud-based service storing Google Fit data related to the specific Google user account that was used to configure the phone. Throughout all this procedure, the smartwatch stayed connected with the paired phone to ensure that the history delete action has been also synchronised between the wearable device and the paired phone. The Wi-Fi has been switched-off again after 30 minutes, a period of time that was considered to be long enough to allow the synchronisation with the Google Fit Store to take place, and then the phone has been switched off.

Once all the Google Fit history has been deleted and the action synchronised with the Google Fit Store and the smartwatch, a new Google Fit walking activity has been started on the smartwatch and a pre-determined walking route has been followed for approximately 15 minutes. At the end of this walking activity the smartwatch has been immediately imaged. The default directory `/data/com.google.android.gms/databases` has been browsed again and it was observed that the Goole Fit database *`fitness.db.test.forensic001_gmail.com`* was present in the directory. The Google Fit database has been extracted and the DataPointRows table has been queried to show only the records containing data points generated by the smartwatch's GPS sensor (Figure 11).



```
SELECT
    _id,
    data_source_id,
    strftime('%Y-%m-%dT%H:%M:%SZ', start_time/1000000000, 'unixepoch', 'localtime') AS Start_Time,
    strftime('%Y-%m-%dT%H:%M:%SZ', end_time/1000000000, 'unixepoch', 'localtime') AS End_Time, data_point
FROM DataPointRows
WHERE
    data_source_id = (SELECT _id FROM DataSources WHERE identifier like '%com.google.location.sample%')
ORDER BY _id ASC
```

	_id	data_source_id	Start_Time	End_Time	data_point
304	1791	5	2016-06-22T12:52:39Z	2016-06-22T12:52:39Z	BLOB
305	1792	5	2016-06-22T12:52:44Z	2016-06-22T12:52:44Z	BLOB
306	1793	5	2016-06-22T12:52:49Z	2016-06-22T12:52:49Z	BLOB
307	1794	5	2016-07-02T20:40:03Z	2016-07-02T20:40:03Z	BLOB
308	1795	5	2016-07-02T20:40:08Z	2016-07-02T20:40:08Z	BLOB
309	1796	5	2016-07-02T20:40:12Z	2016-07-02T20:40:12Z	BLOB

Old records of geo-location data points that have since been deleted from the Google Fit app on the phone
New records of geo-location data points appended to the list of previous records

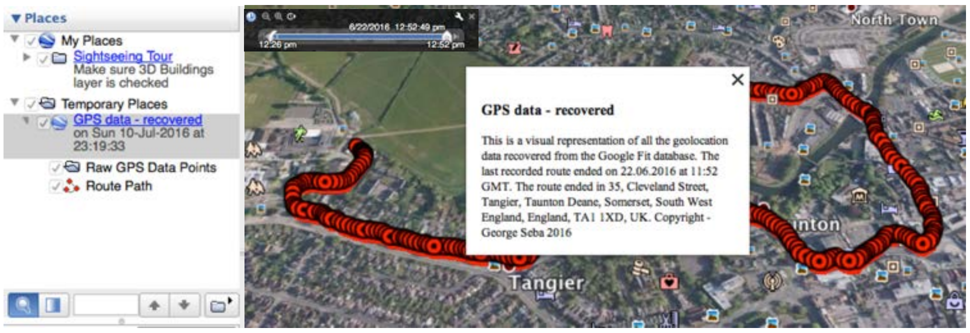
Figure 10 – Screenshot of the DataPointRows table showing new records of data points being appended to the old records that supposed to have been deleted

As seen in Figure 11, the results returned by the SQL query showed that the new records of data points recorded from the smartwatch's GPS sensor while undertaking the walking activity were appended to the list of old records of data points which have since been deleted. This is a very important finding of the research because it shows that even though the Google Fit history has been permanently deleted and this deletion action has been synchronised with both the Google Fit Store and the smartwatch, the Google Fit database on the smartwatch did not delete any historical records containing geo-location data.

4.5 Automation of decoding and visualisation of geo-location data

Developing a method of automating the process of decoding this geo-location data and presenting it in a more readable format could be very important to a forensic investigator examining an Android Wear device and trying to visualise the geographical coordinates stored in the Google Fit database. To automate the process of decoding and extracting geo-location data stored in the Google Fit database a Python script has been developed to parse the binary data and generating a KML file that can be opened with Google Earth.

The script was developed to parse the DataPointRows table and start a loop to iteratively extract the GPS coordinates stored as 8-bytes HEX values in the *data_point* field using the byte offsets presented previously in Table 1. Once extracted, the script converts the HEX values into decimal float-point numbers representing latitude, longitude, altitude and accuracy then it generates a KML file, which is populated with the formatted and newly converted coordinates. The KML file generated by the script can then be opened in Google Earth and the individual data points stored in the DataPointRows table are displayed on the map as red dots (Figure 11).



GPS data - recovered

This is a visual representation of all the geolocation data recovered from the Google Fit database. The last recorded route ended on 22.06.2016 at 11:52 GMT. The route ended in 35, Cleveland Street, Tangier, Taunton Deane, Somerset, South West England, England, TA1 1XD, UK. Copyright - George Seba 2016

Figure 11 –Screenshot of Google Earth displaying the KML file generated by the Python script

Furthermore, as seen in Figure 11 the script has been developed to resolve the last recorded GPS coordinates into a UK postal address, complete with street name, number and postcode which is very important in a forensic investigation. Google Earth also provides further tools that can be very useful in a forensic investigation, such as displaying a timeline based on the dates and times stored in the *start_time* and *end_time* fields of the DataPointRows table or displaying the elevation profile, which shows the speed or the elevation stored in the database table (Figure 12).

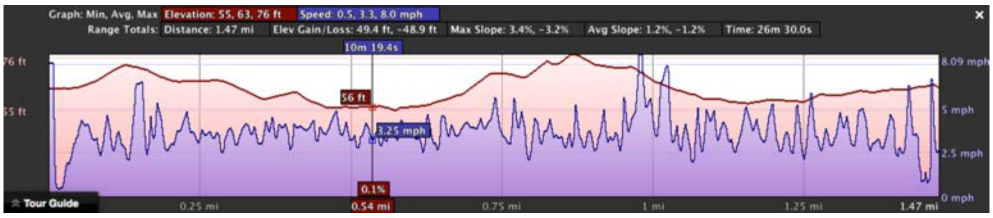


Figure 12 –Screenshot of the Google Earth elevation profile window showing speed and elevation information based on timestamps and altitude coordinates

5. Evaluation

An important finding of the research was that the filename of the Google Fit database follows a specific pattern in which the Google account username used to configure the paired phone with is integral part of the filename. This is particularly important from a forensic perspective because this artefact can link specific geographical coordinates recorded by the device with a specific Google user account, or even more, it can link the smartwatch itself with a specific Google user account since the smartwatch can only be paired with a single phone at any given time and to pair it with another phone the smartwatch would require a factory reset. In contrast, such artefacts could not be found in traditional satellite navigation forensics.

During the analysis of the Google Fit API documentation it was found that Google Fit is accessing an instantaneous reading from one or more device sensors using *data points* and that the value of each data point is formatted and stored programmatically in specific data types defined over the `com.google` namespace. Apart from instantaneous readings, it was also found that data points could also represent an aggregate storing statistics over a time interval, such as the steps cadence or the total number of calories burned during a time interval. Due to the aim of this research, which was focused solely on researching geo-location data, researching the aggregate data points was considered to be beyond the scope of the project. In hindsight, however, this would have been an area that could have benefited from more research because in a forensic investigation the step cadence over a time period can be used to determine if the suspect wearing the smartwatch was walking at ease or in a hurry. Similarly, the step cadence could be used in correlation with speed over a time interval to determine whether the suspect was moving not by walking but by using other methods involving leg movement such as cycling, for example, and all these could be important forensic artefacts to be used during a forensic investigation.

The Google Fit database analysis revealed that it contains 15 tables and following further investigation it was observed that only 6 tables out of these 15 are involved in storing information related to geo-location data or the GPS sensor. That said, the other tables also contain important forensic artefacts such as the start and end time of fitness sessions, times and dates of when the Google Fit performed a synchronisation operation with the paired devices or even the name and unique identifier of the smartwatch. All these could be very useful artefacts in a forensic investigation trying to establish for example when was the last time that the user had in possession the paired phone and the smartwatch together.

It has also been found that geo-location data was stored as individual data points in the DataPointRows table and each data point had a unique UNIX timestamp associated with it. The analysis of multiple datasets containing geo-location data generated during the tests conducted as part of the experimental stages has found that the data points containing GPS coordinates are recorded in the database on average every 4-5 seconds apart. This is an important artefact because in a forensic context this time interval can be used to calculate the speed and the heading of the user wearing the smartwatch. Finally, one of the most important findings resulted from the research undertaken was that the GPS coordinates were stored as 8-bytes HEX values of floats in little-endian format inside the *data_point* field of the DataPointRows table at specific byte-offsets presented in Table 1. Furthermore, it has been found that the conversion of the latitude, longitude, altitude and accuracy from their 64-bit hexadecimal representation to a more human-readable decimal float-point value needs to be done using the IEEE Standard for Binary Floating-Point Arithmetic. This is important because knowing how the coordinates are encoded can help develop a method of automating the conversion process.

A very important outcome of the research has been the development of a Python script that can be used in a forensic investigation to automatically convert the GPS coordinates together with their respective timestamps into a KML file. Once generated by the Python script, the KML file can be further opened in Google Earth or other similar mapping browsers. This offers the forensic practitioner a wide range of very useful tools, such as displaying a timeline based on the timestamps contained in the KML file or even generating an elevation profile based on the coordinates and a series of aggregate functions applied to these coordinates in order to derive tertiary information not stored in the database, such as speed. The information provided by the elevation profile is very important in a forensic investigation since the speed can indicate to the forensic practitioner whether the suspect was running, walking, driving a motor vehicle or was stationary between certain times or between certain geographical locations. Furthermore, this is presented in a comprehensive visual interface which can dramatically improve the workflow and the decision making process in a forensic investigation, specifically because it places hard-to-visualise decimal numbers representing GPS coordinates onto a detailed map, and that makes it much easier to assess the meaning and further forensic implications of those coordinates once they are placed into such a wider context.

However, due to project time constraints this script has been tested only on the version of Google Fit application examined in this project. This could potentially have an impact on the functionality of the script in the future when, for example, future versions of Google Fit might introduce changes to either the database structure or to the structure and format of the geo-location data. The consequences of such changes might potentially result in the script not functioning as designed or returning unexpected results. Since the script developed for this objective is in fact a tool that can be used in a forensic investigation, great care has been taken to design and develop the script so that it checks whether the Google Fit database file is read-only or not and if it is writable, then it creates an exact copy of the file and it works on that copy of the database file, instead of the original. This ensures that the script complies with the ACPO guidelines, especially with Principle 1 regarding the preservation of digital evidence that could be later relied upon in court (ACPO, 2012). Another very important feature of the script is that it generates a log file containing textual information about the outcome of each step of the conversion process executed by the script, together with the date and time when each event occurred. This is very important forensically because it complies with Principle 3 of the ACPO guidelines regarding the necessity of creating a record of all processes applied to digital evidence.

7. Conclusions and Future Work

With the arrival of Android Wear only a few years ago, smartwatches became very popular and their adoption by the wider population is on a fast growing trend. The ubiquity of smartwatches and the fact that are weared by the user most of the time makes it particularly important to a forensic investigator especially since these devices are capable of recording geo-location data using a variety of built-in sensors. This project aimed to determine what geo-location data recorded by Google Fit could be used in a forensic investigation, how this data was structured and whether a process of automating the extraction decoding and presenting of this geo-location data could be developed.

Using a research methodology focused on combining analysis of the Google Fit API and database together with practical experiments conducted in a controlled environment, and following the findings of the research, it can be concluded that important geo-location forensic artefacts can be recovered and visualised from an Android Wear smartwatch that uses the Google Fit application.
Although this project aimed to advance research into an undocumented area of wearable mobile forensics, there are some areas that were not covered by this research, which are discussed below.

Devices
Even though a series of Android Wear smartwatches have been considered, this research only covers the Google Fit geo-location data generated by and recorded on the Sony SWR50 smartwatch. It would be worthwhile to conduct further research in which other makes and models of Android smartwatches are being used. It would be especially significant to find out whether the findings of this research could be confirmed in similar experiments being carried out on smartwatch devices that do not feature a stand-alone GPS sensor but are capable instead of running the Google Fit application.

Wi-Fi and Google Cloud
The Sony SWR50 smartwatch is also capable of connecting directly to a Wi-Fi network. However, this project did not study the implications of having the smartwatch using this feature and only the Bluetooth connection between the paired phone and the smartwatch has been used in the experimental stages. This would be a worthwhile area of further research, especially in light of the results from the experiment set to examine the consequences of deleting all Google Fit history. It would be particularly interesting to determine if after the deletion process would end and both paired devices would be factory reset and configured again with the same Google user account and then only the smartwatch would be allowed to synchronise with the Google Fit Store via Wi-Fi, whether the deleted geo-location data could still be found on the smartwatch.

Future versions
Finally, as previously discussed, the Python script developed as part of this research covers only databases from the version of Google Fit used in this project. This means that future versions of the Google Fit application might affect the functionality of the script if the database structure changes or the data points will be encoded differently.

References

ACPO (2012) *ACPO Good Practice Guide for Digital Evidence, version 5.0*, [online] Available from: http://www.digital-detective.net/digital-forensics-documents/ACPO_Good_Practice_Guide_for_Digital_Evidence_v5.pdf

Cuartielles-Ruiz, D. and Goransson, A. (2015) *Professional Android Wearables*, 1st ed. Hellman, E. (ed.), Indianapolis, John Wiley & Sons, Inc., [online] Available from: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118986873.html

Google (2015a) ‘Data types for instantaneous readings’, *Google Fit API Documentation*, [online] Available from: https://developers.google.com/fit/android/data-types#data_types_for_instantaneous_readings

Google (2015b) ‘Fit REST API - Data Sources’, *Google Fit API*, [online] Available from: https://developers.google.com/fit/rest/v1/reference/users/dataSources

Google (2015c) ‘Google Fit data model’, *Google APIs for Android*, [online] Available from: https://developers.google.com/android/reference/com/google/android/gms/fitness/data/package-summary

IEEE Computer Society (2008) ‘754-2008 - IEEE Standard for Binary Floating-Point Arithmetic’, C/MSC - Microprocessor Standards Committee, [online] Available from: http://standards.ieee.org/findstds/standard/754-2008.html

Kramer, J. A. (2013) ‘DroidSpotter: A Forensic Tool for Android Location Data Collection and Analysis’, Iowa State University, [online] Available from: http://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=4414&context=etd

Last, D. (2009) ‘Expert Advice: GPS Forensics, Crime, and Jamming’, *GPS World*, [online] Available from: http://gpsworld.com/defensesecurity-surveillanceexpert-advice-gps-forensics-crime-and-jamming-8986/

LeMere, B. (2011) ‘Enhancing Investigations with GPS Evidence’, *Forensic Magazine*, [online] Available from: http://www.forensicmag.com/article/2011/04/enhancing-investigations-gps-evidence

Lombardi, M. A., Nelson, L. M., Novick, A. N. and Zhang, V. S. (2001) ‘Time and Frequency Measurements Using the Global Positioning System’, *The International Journal of Metrology*, 8(3), pp. 26–33, [online] Available from: http://www.glb.nist.gov/calibrations/upload/1424.pdf

Parikh, S., Chavda, D., Chakraborty, S., Rughani, P. H. and Dahiya, M. S. . (2015) ‘Analysis of Android Smart Watch Artifacts’, *International Journal of Scientific and Engineering Research*, 6(8), pp. 920–930, [online] Available from: http://www.ijser.org/researchpaper%5CAnalysis-of-Android-Smart-Watch-Artifacts.pdf

Pichai, S. (2014) ‘Sharing what’s up our sleeve: Android coming to wearables’, *Official Google Blog*, [online] Available from: https://googleblog.blogspot.co.uk/2014/03/sharing-whats-up-our-sleeve-android.html

Roeloffs, M., Le-Khac, N.-A. and Kechadi, M.-T. (2014) ‘Forensic Investigation of Tomtom Application on Android Mobile devices’, School of Computer Science and Informatics, University College Dublin, [online] Available from: https://www.insight-centre.org/sites/default/files/publications/ifif-wg-11-9-2014.pdf

Spreitzenbarth, M., Schmitt, S. and Freiling, F. (2012) ‘Comparing Sources of Location Data from Android Smartphones’, In Springer Berlin Heidelberg, pp. 143–157, [online] Available from: http://link.springer.com/10.1007/978-3-642-33962-2_10

Sun, Y. (2012) ‘Geo-Location Forensics on Mobile Devices’, In *International Conference on Digital Forensics and Investigation*, Beijing, Chinese People’s Public Security University, [online] Available from: http://secmeeting.ihep.ac.cn/paper/Paper_Yi_Sun_ICDFI2012.pdf.

Wilson, C. (2015) ‘Sony Smartwatch Forensics – Discovering What the Watch Watched’, *Data Forensics Blog*, [online] Available from: http://www.dataforensics.org/smartwatch-forensics/