Final Report for Automatic a Daily Chronicle Creator for individual project

- Anbang Xu, JIMAHN OK, Taewoo Kim

1. Introduction

Our goal in this project is to develop an Automatic Diary Chronicle Creator for individual person (henceforth ADCC) based on various data that the person is providing to ADCC and other data gathered by our crawler component of ADCC. Examples of primary data from user's smart phone are GPS streams, call history, SMS or picture which the person creates during a specified day. Among these, GPS streams are generated automatically throughout a day based on the person's location, while the person should create other data explicitly to let our application utilizes those data. Also, personal data from Facebook as well as Google Calendar can be crawled and used by ADCC. With these data, ADCC can generate a KML(Keyhole Markup Language) file based on time, data, and place (GPS) and can display the file on a digital map such as Google Maps.

In our project, we have focused on how to analyze GPS streams and consolidate various data sources. Time plays crucial role here since it is used as Primary Key to associate various sources into one meaningful representation. Also, analyzing GPS streams correctly has been another important key factor for our project since time used for consolidating from various sources rely on clustered time duration which we will discuss in subsequent section. In addition, without precise location and time information, we cannot build the ADCC effectively. These two issues have been essential to our success. The following figure shows the abstract structure of the final version of the ADCC.

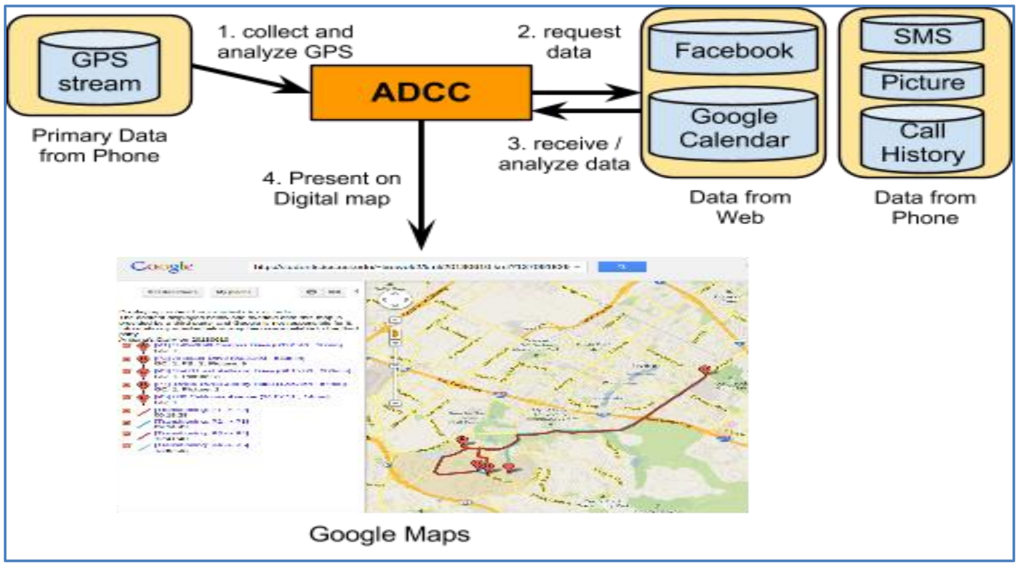


Figure 1. The component and workflow of ADCC

2. Related Work and Our Contribution, Approach

Currently, there are several algorithms to convert GPS streams into meaningful clusters. In [1], the author uses the DBSCAN clustering algorithm for extracting places from GPS data. DBSCAN uses spatial density of data to extract meaningful GPS groups, which are clusters. Another paper [2][3][4] mentions that significant places are the locations where a person spends some periods of time. From the GPS data, the algorithm looks for locations where the person stays for a some specified time and cluster these locations into group. Also, in [5], the author contends that GPS points can be sequentially connected into a track. The author further says that such track can be divided into trips if the staying time interval of the consecutive points exceeds a given threshold. A change point is a place where people change their transportation modes. [6] also mentions that a new setup of clustering is started when distance between the new location and the center of the current cluster is greater than certain threshold. Their algorithm compares each incoming coordinate with previous coordinates in the current cluster. By considering these researches, we have concluded that we need to introduce additional step to the current GPS analysis algorithms.

We will put our implementation detail about analyzing GPS coordinate in the next chapter. However, this is maybe a good place to put the summary about our algorithm. Basically, our algorithm works similarly when comparing to current algorithms except one more step. First, our algorithm checks the distance between neighbor coordinate **x** and **y** and if distance between **x** and **y** is less than certain threshold(***t1 : unit - meter***) [*dist(x, y) <= t1*], we will make **x** and **y** belong to same cluster. For next coordinate **z** followed by **y**, if distance between **x**(now, it became starting point of a cluster) and **z** is less than **t1** [*dist(x, z) <= t1*], z will be belong to the same cluster which x and y belong to. If distance between x and z is greater than t1, we will begin to find new cluster. This step looks similar to current algorithms. However, our algorithm has an additional step. After finding and creating number of clusters, our algorithm iterates through cluster's starting point(which is base point of that cluster) to calculate distance between starting point ***a*** of one cluster and another starting point ***b*** of next neighbor cluster. If distance between **a** and **b** is less than certain threshold(***t2 : unit - meter***) [dist(a, b) <= t2], our algorithm merge two clusters into one cluster which **a** belongs to. Note that usually threshold **t2** needs to be greater than **t1**. Our algorithm executes this additional step to minimize GPS error. For example, the person can stay in a same spot for one hours. Then, there could be a GPS error that the location of that person could be reported in the another spot whose distance is greater than **t1**. After that, the location of that person could be reported continuously at the error spot. Then, we will have two cluster. However, these two clusters are originally one spot and should be merged. Therefore, **t2** should be greater than **t1**. By applying this additional step, we could minimize the error. After these two merge steps, we finally count the duration time of the each cluster that the person spent for each cluster. If duration for each cluster is less than 10 minutes, we regard the status of the cluster as transitioning to another place and delete the cluster from the clusters group. These steps are summarized in the Figure 2 below.

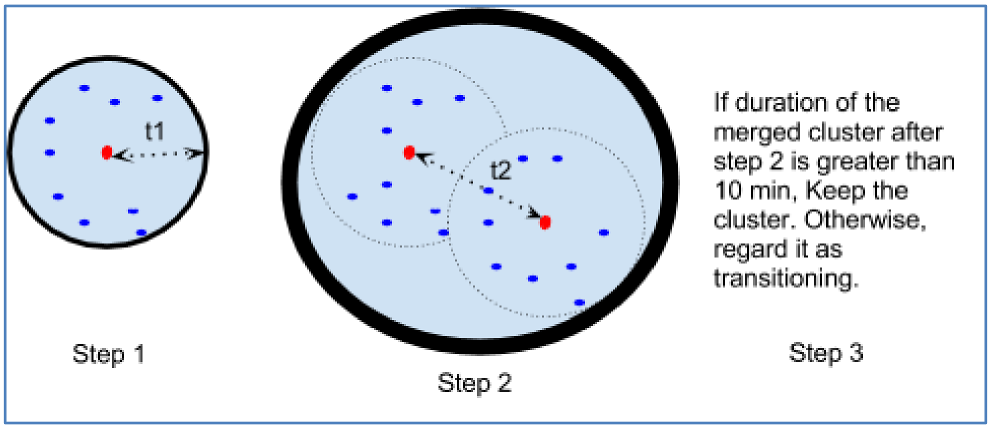
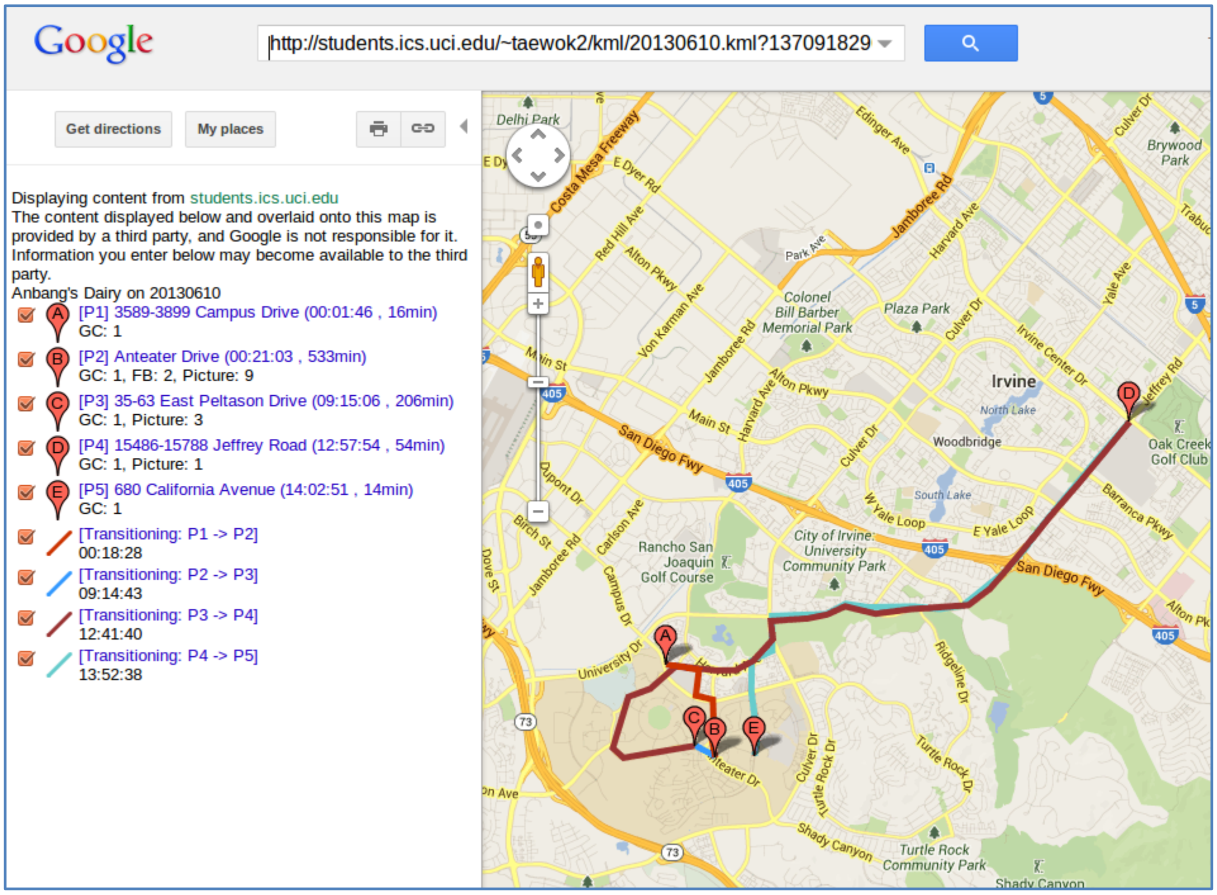

t1
t2
If duration of the merged cluster after step 2 is greater than 10 min, Keep the cluster. Otherwise, regard it as transitioning.
Step 1
Step 2
Step 3

Figure 2. Our GPS Analysis Algorithm

Even after correctly analyzing user's movement activities through GPS streams, there are some important issues left. GPS data should be consolidated with other data sources in order to form a unified journal for user's activities. For generating GPS stream and transferring to Server, we have uses the GPS Logger application in the Android Phone[7]. This app runs in the background in the phone so that we can take a long walk, hiking, a flight, a ride, a photo session or even go buy some groceries and have this application running as long as possible. The user can create text file which includes raw data of GPS coordinates based on the specific day. After we collect GPS streams in Android Phone with GPS Logger in Text file format, we have applied our GPS algorithm to analyze GPS coordinates. Then, we have converted geographic coordinates of each cluster into a human-readable address. For this purpose, we have applied reverse geo-coding using Google Geo-coding API[8] (The term geo-coding generally refers to translating a human-readable address into GPS data. The process of doing the converse, translating a location data into a human-readable address, is known as reverse geo-coding).

Other than GPS, we collect data from the various sources such as Google Calendar, Facebook in the Web and also collect Call history, SMS, Picture from the user's smartphone. For Google Calendar schedule data, we have utilized Google Calendar API[9]. Google Calendar allows client applications to view and update calendar events in the form of Google Data API feeds. Our application queries for events that had happened during the specified day. For example, we can crawl schedules from the Google Calendar on Jun 13, 2013. Here, you can see that time plays important role because time is the primary key that will associates various data sources. Another collecting method is using Facebook API to extract personal data from the Facebook. The Facebook Graph API[10] is the primary way to get data in and out of Facebook's social graph. It's a low-level HTTP-based API that we can use to query data so that we use this API to crawl post or events during specific day. In addition, we will collect call history, SMS, and picture from user's smartphone.

After collecting data sources, we consolidate the data when user specifically requests based on the specified day. For every data source, we store time as primary key so that we can associate any data sources based on the time period of each GPS cluster. Finally, we create KML file from consolidates sources and display the file on Google Maps as follows.


Google
http://students.ics.uci.edu/~taewok2/kml/20130610.kml?137091829
Get directions
My places

Displaying content from students.ics.uci.edu
The content displayed below and overlaid onto this map is provided by a third party, and Google is not responsible for it. Information you enter below may become available to the third party.
Anbang's Dairy on 20130610

- A [P1] 3589-3899 Campus Drive (00:01:46 , 16min)
  GC: 1
- B [P2] Anteater Drive (00:21:03 , 533min)
  GC: 1, FB: 2, Picture: 9
- C [P3] 35-63 East Peltason Drive (09:15:06 , 206min)
  GC: 1, Picture: 3
- D [P4] 15486-15788 Jeffrey Road (12:57:54 , 54min)
  GC: 1, Picture: 1
- E [P5] 680 California Avenue (14:02:51 , 14min)
  GC: 1
- [Transitioning: P1 -> P2]
  00:18:28
- [Transitioning: P2 -> P3]
  09:14:43
- [Transitioning: P3 -> P4]
  12:41:40
- [Transitioning: P4 -> P5]
  13:52:38

Figure 3. The KML file is displayed on the Google Maps

3. Our Implementation

1) Work Flow

The workflow of our application is as follows. 1) The ADCC collects and analyze GPS stream based on the time and GPS coordinate. 2) The ADCC collects data from Google Calendar, Facebook, Call history, Picture, and SMS based on the given time duration as Primary Key which was generated in the analysis step 1. 3) The ADCC creates KML file based on the result of step 2. 4) Finally, the ADCC will present KML file on the Google Maps.

2) Step 1 : Collect and Analyze GPS Stream

As stated in earlier chapter, the format of GPS raw data file is comma delimited text file. The file's format is as follows :

```
2013-06-06T07:08:03Z,33.643818,-117.841371, ...
```

The first column (2013-06-06T07:08:03Z) stands for the time that GPS coordinates were recorded. The last character(Z) in the first column means that time zone used for recording is GMT. Since PDT time zone is -7 hours from GMT, w(e need to subtract -7 hours from the raw data. Second and third column is the latitude (33.643818) and longitude(-117.841371) of GPS coordinates, respectively. We use these three column to collect the time which will be used as Primary Key for consolidating other sources and location from GPS raw data.

After collecting data from the Phone application and storing the raw data on the server, we applied our GPS analyzation algorithm stated in earlier chapter. To calculate distance between two GPS coordinates, we have applied the Haversine formula[11]. Following codes show the important part of the GPS analyzation algorithm.

```
// 1 - Read and Store raw data into Java Object
Scanner scanner = new Scanner(file);
String line = scanner.nextLine();
String lineGPS[] = line.split(",");
recordDate.add(t1);
latitude.add(Double.parseDouble(lineGPS[1]));
longitude.add(Double.parseDouble(lineGPS[2]));

// 2 - Create clusters based on threshold t1(distance between base point and new point)
distanceMfromBasePoint.add(HaversineInM(latitude.get(basePoint), longitude.get(basePoint),
latitude.get(i), longitude.get(i)));
if (distanceMfromBasePoint.get(i) <= cluster_threshold)  {
  clusterLatitude.add(clusterLatitude.get(i-1));
  clusterLongitude.add(clusterLongitude.get(i-1));  }
```

// 3 - Merge clusters

```
double distanceBetweenBasePoints = HaversineInM(latitude.get(currentBasePoint), longitude.get(currentBasePoint), latitude.get(nextBasePoint), longitude.get(nextBasePoint));

if (distanceBetweenBasePoints <= interCluster_threshold) {
   clusterLatitude.set(j, clusterLatitude.get(currentBasePoint));
   clusterLongitude.set(j, clusterLongitude.get(currentBasePoint)); }
```

// 4 - Print Final base Point(Starting point of a cluster)

```
for (int i=0;i<basePoints.size();i++)        {
  if (duration.get(i) >= 5)
              System.out.println(basePoints.get(i) + " - duration : " + duration.get(i) + "min. " + ….   }
```

3) Step 2 : Collect and Analyze data sources

A. Google Calendar

We use Google Calendar API[9] to crawl user’s Google Calendar data. First of all, we need to pass username and password to authenticate and retrieve the calendar lists which we need. And then we store the corresponding data we need in java object. In our project, we crawl the calendar data from a specified date. Three important classes are CalendarService, CalendarFeed and CalendarEntry. Code sample is as follows :

// Create a CalenderService and authenticate

```
CalendarService myService = new CalendarService("exampleCo-exampleApp-1");
myService.setUserCredentials("jo@gmail.com", "mypassword");
```

// Send the request and print the response

```
URL feedUrl = new URL("https://www.google.com/calendar/feeds/default/owncalendars/full");
CalendarFeed resultFeed = myService.getFeed(feedUrl, CalendarFeed.class);
for (int i = 0; i<resultFeed.getEntries().size(); i++)
```

```
  CalendarEntry entry = resultFeed.getEntries().get(i);

//Query calendar in a specific range between startTime and endTime
calendar = dateRangeQuery(myService, DateTime.parseDate(startTime), DateTime
        .parseDate(endTime));

//Store imported data in java object fields
GoogleCalendar calendar = new GoogleCalendar();
calendar.setStartTime(startList);
calendar.setEndTime(endList);
calendar.setTitle(titleList);
calendar.setLocation(locationList);
calendar.setDescription(descriptionList);
calendar.setActionLink(actionLink);
```

B. Facebook

We use the Facebook API[10] to crawl Facebook. Firstly, we need to provide access key of facebook and do the query in the specific date. And then we store the data we need into java object. Code sample is as follows :

```
//Initialization
FacebookClient facebookClient = new DefaultFacebookClient(MY_ACCESS_TOKEN);

//Fetching User information
User user = facebookClient.fetchObject("me", User.class);

//Fetching Connection information
Connection<User> myFriends = facebookClient.fetchConnection("me/friends", User.class);
Connection<Post> myFeed = facebookClient.fetchConnection("me/feed", Post.class);
Connection<Photo> myPhoto = facebookClient.fetchConnection("me/feed", Photo.class);

//Filter Connection information
Connection<?> filteredFeed = facebookCrawl.filterConnection(startDate, endDate);
```

```
//Store imported data in the Java object
FB fb = new FB();
fb.setPostTime(postTime);
fb.setType(type);
fb.setTitle(title);
fb.setLink(link);
fb.setdescription(description);
fb.setActionLink(actionLink);
```

C. Call history and SMS

We use Android Application(Call Log Tools and SMS Tools) to collect call history and SMS as CSV(Comma Separated Values) format file, and then use OpenCSV library[12] to parse the CSV file and eventually store the data we need into java object. The important part of the code is as follows :

```
//Read CSV file
CSVReader csvReader = new CSVReader(new FileReader(csvFilename));

//read line by line
while((row = csvReader.readNext()) != null)

//Store import fields into java object
Call call = new Call();
call.setCallTime(callTime);
call.setName(name);
call.setNumber(number);
call.setType(type);

SMS sms = new SMS();
sms.setTextTime(textTime);
sms.setNumber(number);
sms.setContent(content);
```

D. Images in the Phone

We copy the Image directory from phone to PC and use metadata-extractor[13] to analyse the metadata such as EXIF of all the images under that directory.

```
//Initialization
Metadata metadata = ImageMetadataReader.readMetadata(fs);

// query the tag's value
Date date = directory.getDate(ExifSubIFDDirectory.TAG_DATETIME_ORIGINAL);

//Store data into JAVA object
Photo photo = new Photo();
photo.setCreateTime(createTime);
photo.setType(type);
photo.setTitle(title);
photo.setLink(link);
```

4) Step 3 : Consolidates Data sources and Presents these information on the Google Maps

A. KML file

We firstly need to talk about KML (Keyhole Markup Language) file format. We decided to generate KML file as intermediate output of ADCC to present our final results on the Web (in our case, on the Google Maps) rather than making one web page only for our application. The reasons to make this decision are as follows:

(1) We can create more flexible multimedia document as our output.

(2) It is safe to choose KML file as our output because KML has already become an international standard of the Open Geospatial Consortium in 2008[14][15].

(3) As we can notice from its name, KML is an XML grammar and file format for modeling and storing geographic features such as points, lines. That is, KML is designed to describe and carry data, so if users want, they can use KML to share places and information with other users.

As we've already seen in chapter 2., we applied our KML file on Google Maps. Not only Google Map, but also any other applications like Marble[16] which can import KML files could be the presentation tier of our application. As an example, here is a captured screenshot when we apply our KML file to Google Earth:

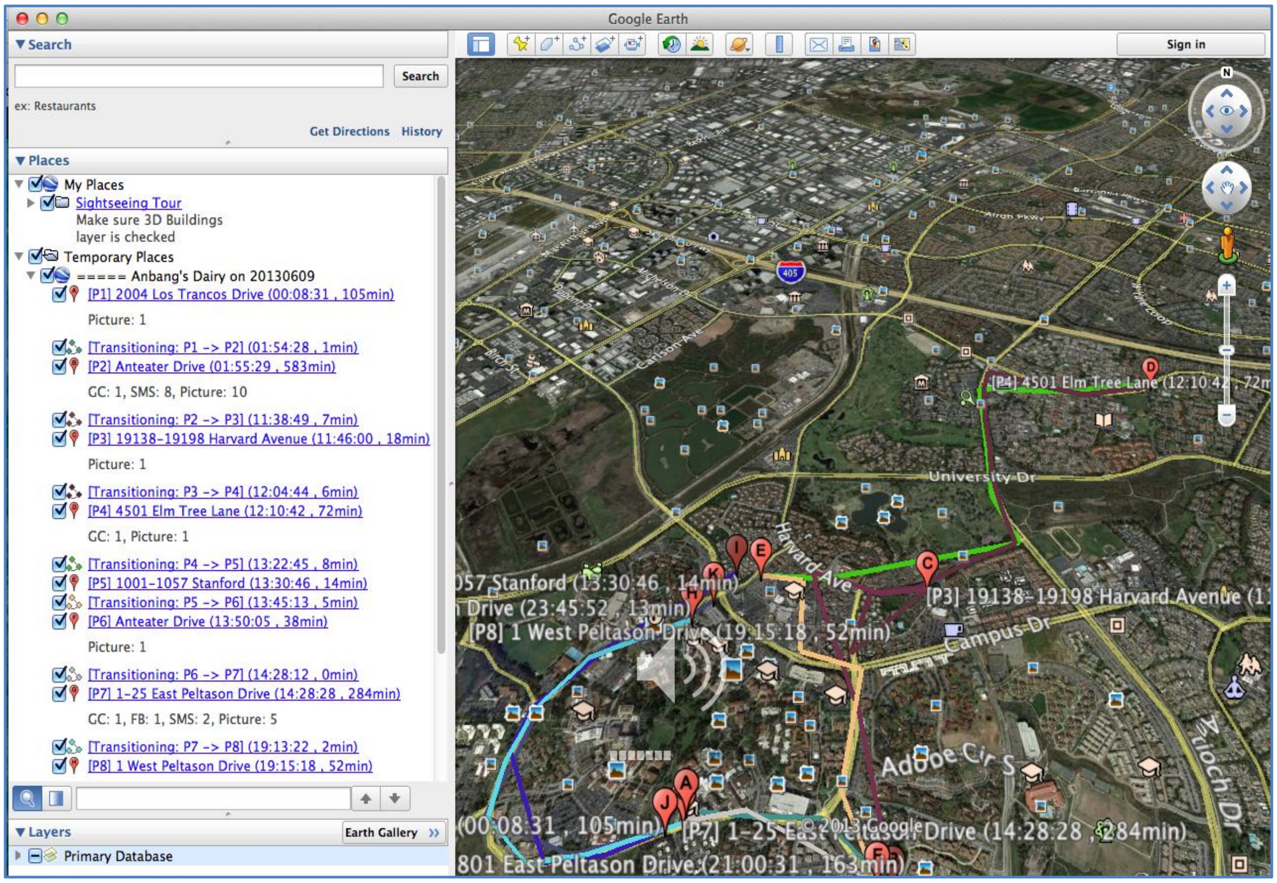

My Places
Sightseeing Tour
Make sure 3D Buildings layer is checked
Temporary Places
===== Anbang's Dairy on 20130609
[P1] 2004 Los Trancos Drive (00:08:31 , 105min)
Picture: 1
[Transitioning: P1 -> P2] (01:54:28 , 1min)
[P2] Anteater Drive (01:55:29 , 583min)
GC: 1, SMS: 8, Picture: 10
[Transitioning: P2 -> P3] (11:38:49 , 7min)
[P3] 19138-19198 Harvard Avenue (11:46:00 , 18min)
Picture: 1
[Transitioning: P3 -> P4] (12:04:44 , 6min)
[P4] 4501 Elm Tree Lane (12:10:42 , 72min)
GC: 1, Picture: 1
[Transitioning: P4 -> P5] (13:22:45 , 8min)
[P5] 1001-1057 Stanford (13:30:46 , 14min)
[Transitioning: P5 -> P6] (13:45:13 , 5min)
[P6] Anteater Drive (13:50:05 , 38min)
Picture: 1
[Transitioning: P6 -> P7] (14:28:12 , 0min)
[P7] 1-25 East Peltason Drive (14:28:28 , 284min)
GC: 1, FB: 1, SMS: 2, Picture: 5
[Transitioning: P7 -> P8] (19:13:22 , 2min)
[P8] 1 West Peltason Drive (19:15:18 , 52min)
Google Earth
Search
ex: Restaurants
Get Directions
History
Places
Layers
Earth Gallery
Primary Database
Sign in
University Dr
Harvard Ave
Campus Dr
Adobe Cir S
[P4] 4501 Elm Tree Lane (12:10:42 , 72m
057 Stanford (13:30:46 , 14min)
Drive (23:45:52 , 13min)
[P8] 1 West Peltason Drive (19:15:18 , 52min)
[P3] 19138-19198 Harvard Avenue (1
(00:08:31 , 105min)
[P7] 1-25 East Peltason Drive (14:28:28 , 284min)
801 East Peltason Drive (21:00:31 , 163min)
© 2013 Google

Figure 4. Our KML file on the Google Earth

B. Consolidating Data Sources

To consolidate data, firstly we made time frames by categorising our GPS tracking base point data into two categories, ‘place’ and ‘transitioning’. If duration time in a base point is less than 10 min, we consider those points as ‘transitioning’, otherwise as ‘place’. For example, if base point data are as follows:

0 - duration : **16min**. 33.651974 -117.841756 Mon Jun 10 00:01:46 PDT 2013

44 - duration : **0min**. 33.651359 -117.837291 Mon Jun 10 00:18:28 PDT 2013

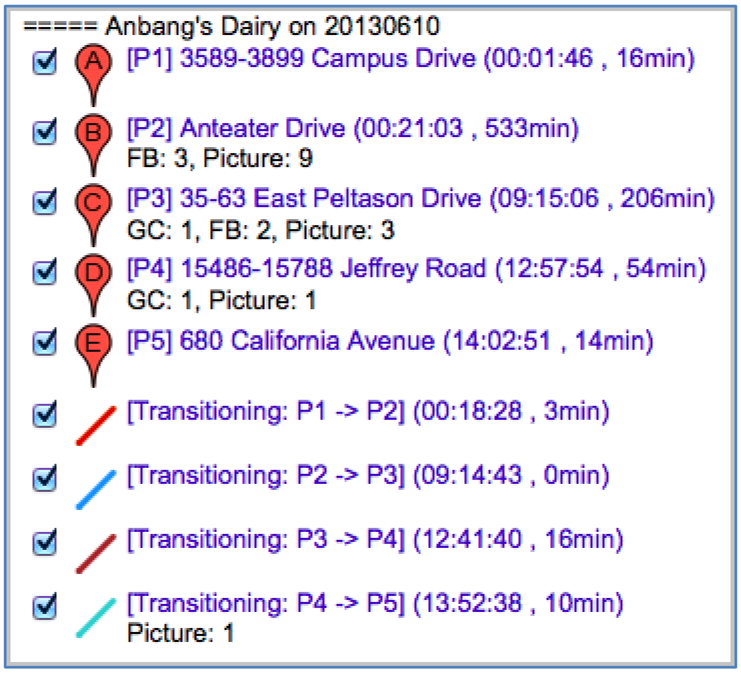
46 - duration : **0min**. 33.648466 -117.83773 Mon Jun 10 00:19:06 PDT 2013

51 - duration : **0min**. 33.643667 -117.835579 Mon Jun 10 00:20:18 PDT 2013

53 - duration : **533min**. 33.641475 -117.835081 Mon Jun 10 00:21:03 PDT 2013 we assumed that a user was transitioning from point ‘0’ to point ‘53’ when the user is at point ‘44’, ‘46’, and ‘51. So, in this example, we have two ‘place’ (point ‘0’ and point ‘53’) and one ‘transitioning’ (from point ‘44’ to point ‘51’). After categorizing GPS data, we consolidated all the data from various data sources such as GPS, SMS, Calls, Pictures, Google Calendar, and Facebook into the each time frame according to the time data of each data source. That is, we use ‘TIME’ as our major synchronizer.

C. Presenting information on the Google Maps

We used ‘Point’ element and ‘LineString’ element of KML format to present ‘place’ and ‘transitioning’, respectively. Each place is marked with an icon in alphabetical order and each transitioning is represented with line in different color. We put address information with time and duration data as name of each place and also provide summarized information to present how many data are included from each data source in that place. Following is an example (GC: Google Calendar, FB: Facebook):

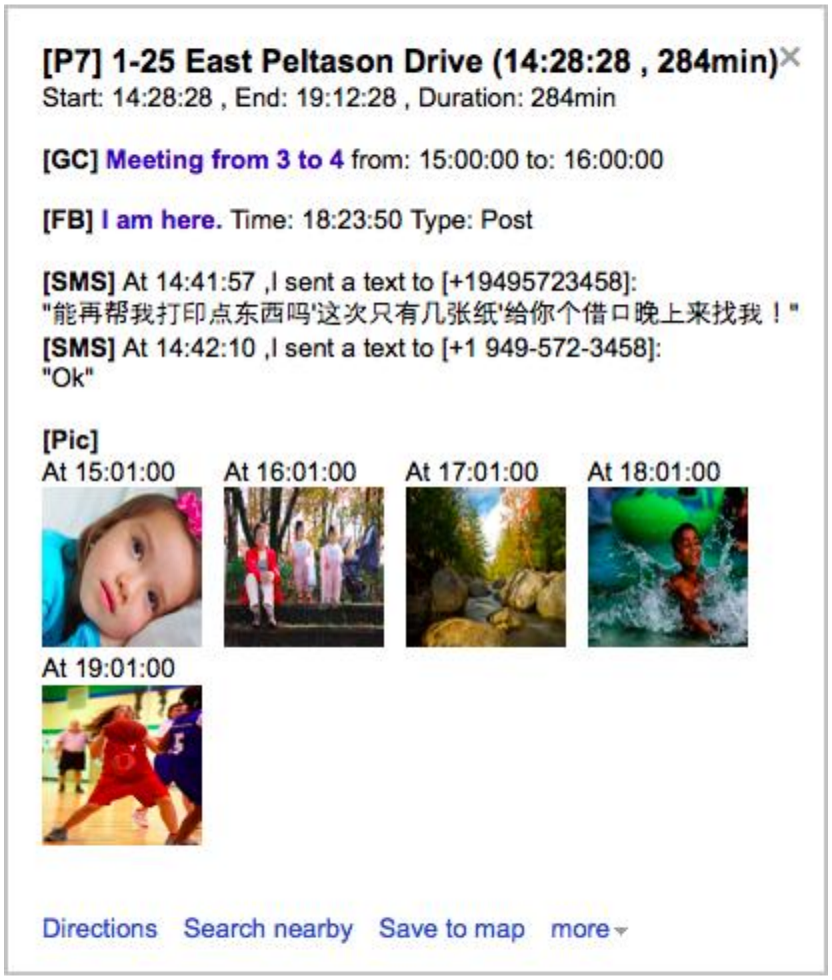


===== Anbang's Dairy on 20130610

- ✓ A [P1] 3589-3899 Campus Drive (00:01:46 , 16min)
- ✓ B [P2] Anteater Drive (00:21:03 , 533min)
  FB: 3, Picture: 9
- ✓ C [P3] 35-63 East Peltason Drive (09:15:06 , 206min)
  GC: 1, FB: 2, Picture: 3
- ✓ D [P4] 15486-15788 Jeffrey Road (12:57:54 , 54min)
  GC: 1, Picture: 1
- ✓ E [P5] 680 California Avenue (14:02:51 , 14min)
- ✓ [Transitioning: P1 -> P2] (00:18:28 , 3min)
- ✓ [Transitioning: P2 -> P3] (09:14:43 , 0min)
- ✓ [Transitioning: P3 -> P4] (12:41:40 , 16min)
- ✓ [Transitioning: P4 -> P5] (13:52:38 , 10min)
  Picture: 1

Figure 5. Summarized information about data

Google Maps supports to write descriptions about a certain place with links using pop-up windows. Users can see the pop-up window by clicking names of ‘place’ or ‘transitioning’. We displayed all detailed data gathered from every data source into the descriptions with links to the original data. By doing this, users can see what schedules were there in Google Calendar, what text messages were sent or received, and what pictures they took at that place using their phone. Here is an example of that pop-up window:



[P7] 1-25 East Peltason Drive (14:28:28 , 284min)
Start: 14:28:28 , End: 19:12:28 , Duration: 284min

[GC] **Meeting from 3 to 4** from: 15:00:00 to: 16:00:00

[FB] **I am here.** Time: 18:23:50 Type: Post

[SMS] At 14:41:57 ,I sent a text to [+19495723458]:
"能再帮我打印点东西吗'这次只有几张纸'给你个借口晚上来找我！"
[SMS] At 14:42:10 ,I sent a text to [+1 949-572-3458]:
"Ok"

[Pic]
At 15:01:00 At 16:01:00 At 17:01:00 At 18:01:00
At 19:01:00

Figure 6. An example of Pop-Up window showing detailed information

4. What we have applied to our project based on the professor's Feedback

For our research report, the professor provided important feedback. First, we need to define role of time in the organizing and combining data more clearly. Second, we need to develop visualization of data from various sources. Therefore, the professor finally suggested that just select a few sources(photos, phone calls/messages and 1 or 2 more) and decide how to use GPS to make all these data interesting. Maybe, summarization of data can be also applied.

After receiving the feedback from the professor, we have defined the role of time more clearly. First of all, we calculate time duration(starting time, ending time) of each GPS cluster(place) that the user stayed which satisfies our GPS algorithm condition. This duration plays as Primary Key to collect data from other sources. In fact, we applied duration of each cluster to gather relevant data which were generated at that duration. For example, if duration of one cluster is 10:03 ~ 12:05 on Jun 13, our program gathers the data from various sources on that duration. For developing visualization, we have decided that for given amount of time, it is better to construct the KML file and present that file on the Google Maps application because the KML file structure has many features that we would like to use so that we could utilize many aspects of KML files. In addition, the KML file can be displayed well on the Google Maps. For data sources, we have decided that we include photos, phone calls, SMS messages, Google Calendar, and Facebook data to focus. For summarization part, we show number of data in every cluster display before user checks in detail.

Overall, the professor's feedback provided more clear way of refining our project and we really appreciate the feedback.

5. Limitations of our work and Future Work

First of all, we have problems analyzing GPS streams because of application error. The program omit some GPS coordinates during random time. For example, It did not record GPS coordinates raw data between 2PM~3PM on a random day. Therefore, analyzing GPS streams was sometime harder than normal cases. Next, we could not fully automate our work. For storing GPS, Picture, SMS, Call history data on the location which the ADCC resides, we need to move the files from the phone to the server manually. This is because we used several outside applications to collect GPS data, SMS, or Call history of the phone. On the contrary, the ADCC can collect Google Calendar and Facebook data automatically. For presenting KML data on the Google maps, because the KML file should be located in the public Web-site, we could not fully automate the process. We need to manually upload the KML file on our Web server due to security reasons.

For future work, we can automate collecting process with help of the scheduler. If we find the outside program which periodically generates GPS, SMS, Call history data on a specified format on a specific directory on the phone's storage, we can create a program to crawl these data and send them to the Server. Or, we can create our own program to create GPS, SMS, Call history data on our format. For KML file presentation, we could think of creating customized upload module that will automatically transfer our generated KML file to our Web server.

6. References

[1] Nishino, M., Nakamura, Y., Yagi, T., Muto, S., Abe, M. : A Location Predictor based on Dependencies Between Multiple Lifelog Data, 12p-13p, 2010.

[2] Liao, L., Patterson, J. D., Fox, D., Kautz, H. : Building Personal Maps from GPS Data, 252p. 2006

[3] Zha, H., Ding, C., Gu, M., He, X., Simon, H.D. : Spectral Relaxation for K-means Clustering. Neural Information Processing Systems 14, 1057p–1064p, 2001.

[4] Ashbrook, D., Starner, T .: Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users. Personal and Ubiquitous Computing, 275p–286p, 2003.

[5] Zheng, Y., Liu, L. Wang, L. Xie, X. : Learning Transportation Mode from Raw GPS Data for Geographic Applications on the Web, 2008.

[6] Kang, J., Welbourne, W., Stewart, B., Borriello, G.: Extracting Places from Traces of Locations. In: Proc. WMASH, 110p–118p. ACM Press, New York, 2004

[7] Play.google.com. n.d.. GPS Logger for Android - Android Apps on Google Play. [online] Available at: https://play.google.com/store/apps/details?id=com.mendhak.GPSlogger&hl=en [Accessed: 1 Jun 2013].

[8] Developers.google.com. n.d.. The Google Geocoding API - Google Maps API Web Services — Google Developers. [online] Available at: https://developers.google.com/maps/documentation/geocoding/ [Accessed: 1 Jun 2013].

[9] Developers.google.com. n.d.. Google Calendar API v2 Developer's Guide: Java - Google Apps Platform — Google Developers. [online] Available at: https://developers.google.com/google-apps/calendar/v2/developers_guide_java [Accessed: 1 Jun 2013].

[10] Facebook Developers. 2013. Getting Started: The Graph API. [online] Available at: https://developers.facebook.com/docs/getting-started/graphapi/ [Accessed: 1 Jun 2013].

[11] En.wikipedia.org. n.d.. Haversine formula - Wikipedia, the free encyclopedia. [online] Available at: http://en.wikipedia.org/wiki/Haversine_formula [Accessed: 1 Jun 2013].

[12] Sullis, B. 2011. opencsv - Browse /opencsv at SourceForge.net. [online] Available at: http://sourceforge.net/projects/opencsv/files/opencsv/ [Accessed: 1 Jun 2013].

[13] Code.google.com. n.d.. metadata-extractor - Extracts Exif, IPTC, XMP, ICC and other metadata from image files - Google Project Hosting. [online] Available at: http://code.google.com/p/metadata-extractor/ [Accessed: 1 Jun 2013].

[14] Opengeospatial.org. 2008. OGC® Approves KML as Open Standard | OGC(R). [online] Available at: http://www.opengeospatial.org/pressroom/pressreleases/857 [Accessed: 11 Jun 2013].

[15] Opengeospatial.org. 2000. KML | OGC(R). [online] Available at: http://www.opengeospatial.org/standards/KML [Accessed: 1 Jun 2013].

[16] En.wikipedia.org. 2006. Marble (software) - Wikipedia, the free encyclopedia. [online] Available at: http://en.wikipedia.org/wiki/Marble_%28KDE%29 [Accessed: 1 Jun 2013].