SAfeDJ: A Crowd-Cloud Co-design Approach to Situation-aware Music Delivery for Drivers

XIPING HU, The University of British Columbia
JUNQI DENG, The University of Hong Kong
JIDI ZHAO, East China Normal University
WENYAN HU, Nankai University
EDITH C.-H. NGAI, Uppsala University
RENFEI WANG, IBM Canada
JOHNNY SHEN, The University of British Columbia
MIN LIANG, IBM China
XITONG LI, HEC Paris
VICTOR C.M. LEUNG, The University of British Columbia
YU-KWONG KWOK, The University of Hong Kong

Driving is an integral part of our everyday lives, but it is also a time when people are uniquely vulnerable. Previous research has demonstrated that not only does listening to suitable music while driving not impair driving performance, but it could lead to an improved mood and a more relaxed body state, which could improve driving performance and promote safe driving significantly. In this paper, we propose SAfeDJ, a smartphone-based situation-aware music recommendation system, which is designed to turn driving into a safe and enjoyable experience. SAfeDJ aims at helping drivers to diminish fatigue and negative emotion. Its design is based on novel interactive methods, which enable in-car smartphones to orchestrate multiple sources of sensing data and the drivers' social context, in collaboration with cloud computing to form a seamless crowdsensing solution. This solution enables different smartphones to collaboratively recommend preferable music to drivers according to each driver's specific situations in an automated and intelligent manner. Practical experiments of SAfeDJ have proved its effectiveness in music-mood analysis, and mood-fatigue detections of drivers with reasonable computation and communication overheads on smartphones. Also, our user studies have demonstrated that SAfeDJ helps to decrease fatigue degree and negative mood degree of drivers by 49.09% and 36.35%, respectively, compared to traditional smartphone-based music player under similar driving situations.

Categories and Subject Descriptors: **C.2.4 [Computer Systems Organization]**: Distributed Systems

General Terms: Design, Algorithms, Experimentation

Additional Key Words and Phrases: Smartphones, crowdsensing, cloud, music mood, context, driving

ACM Reference Format:

Xiping Hu, Junqi Deng, Jidi Zhao, Wenyan Hu, Edith C.-H. Ngai, Renfei Wang, Johnny Shen, Min Liang, Xitong Li, Victor C.M. Leung, Yu-kwong Kwok, 2015. SAfeDJ: A Crowd-Cloud Co-design Approach to Situation-aware Music Delivery for Drivers. *ACM Transactions on Multimedia Computing, Communications and Applications.*

1. INTRODUCTION

According to the statistics published by the World Health Organization (WHO) [WHO 2014], 1.2 million people die and 50 million people are injured or disabled on roads every year. Statistics in Europe show that approximately 10-20% of all traffic accidents result from diminished vigilance of drivers, e.g., fatigue or negative emotion. Previous research has demonstrated that not only does listening to suitable

This work is supported in part by the Canadian Natural Sciences and Engineering Research Council through the NSERC DIVA Strategic Network, by TELUS and other industry partners, by the National Social Science Found of China under Grant 11&ZD174, Shanghai Pujiang Program and ECNU International Publication Program, and by the Vinnova GreenIoT project and STINT grant for international collaboration in Sweden.

Author's address: X. Hu (corresponding author), email: xipingh@ece.ubc.ca; J. Zhao (corresponding author), email: jdzhao@sem.ecnu.edu.cn

music while driving not impair driving performance, but it could lead to an improved mood and a more relaxed body state (e.g., decreasing the high breath rates of drivers when they are fatigued), which could improve driving performance and promote safe driving significantly [Zwaag et al. 2012].

In recent years, the capabilities of mobile devices such as smartphones have improved substantially. These capabilities include significant increases in computational resources (processing capability, local storage), multiple radios (second/third/fourth generation cellular, WiFi, Bluetooth, WiFi Direct, etc.), various sensing modules (cameras, accelerometer, gravity sensors, etc.), and high level programming languages (Java in Android, Object C in iOS), enabling mobile devices to form mobile cyber-physical systems (CPS) [Hu et al. 2013] and support many novel mobile applications in our daily lives [Hu² et al. 2013; Hu et al. 2015].

However, a comprehensive smartphone-based music platform specifically designed for vehicular users is not yet available. Some music recommendation systems available in the market like Last.fm [Last 2014], and the existing research work in mobile music recommendation like CAMRS [Wang et al. 2012], recommend music to users based only on their listening behavior, history, and/or locations [Baltrunas et al. 2012]; or only focus on distributed music resource (i.e., decoder) sharing [Ayaki et al. 2009]. Also, recent works like AmbiTune [Helmholz et al. 2013; Helmholz et al. 2014] only adapt the music to drivers based on the prediction of their route trajectories and driving speeds, but they do not include interactive methods that enable individual drivers to recommend suitable music in their new driving situations collaboratively. On the other hand, research works [Cassidy et al. 2009; Hunter et al. 2011] and our online survey [Driving 2014] have demonstrated that drivers' situations including their mood and fatigue status impact the choice of preferable music significantly while driving. Therefore, to achieve situation-aware music delivery for drivers via the smartphones they carry, two research challenges that have not been addressed by the existing research need to be explored:

- How to efficiently process and identify the attributes (e.g., tempo and tonal type) of each song and its corresponding music-moods to enable a smartphone to recommend music to a driver that fits his/her real-time mood;
- How a smartphone can effectively deal with the wide ranges of possible human moods and traffic conditions and recommend suitable music to a driver under driving situations that the driver may or may not have experienced before.

This paper addresses the two research challenges identified above by proposing SAfeDJ, a novel smartphone-based situation-aware music delivery system for drivers, which aims to help them relieve fatigue and ease negative emotion during driving. Our contributions in this paper are summarized as follows:

- To the best of our knowledge, SAfeDJ is the first smartphone-based interactive music recommendation system that recommends music not only based on drivers' listening behaviors, but also their real-time mood-fatigue levels and traffic conditions.
- We design novel interactive methods for smartphones, which orchestrate multiple sources of sensing data (e.g., On-Board Diagnostic (OBD) units, heart rate sensors, and front camera of smartphones) with drivers' social contexts, and leverages advantages of cloud computing to form a seamless crowdsensing solution. This solution enables smartphones to collaboratively and effectively recommend preferable music to drivers in real-time according to each driver's specific situations.

- We deploy and evaluate SAfeDJ through a set of real-world scenarios, which show that SAfeDJ helps to decrease fatigue and negative mood degrees of drivers by 49.09% and 36.35%, respectively, compared with traditional smartphone-based music player under similar driving situations.

The rest of the paper is organized as follows. Section 2 reviews the background of related techniques and concepts of SAfeDJ. Section 3 presents the models and key designs of SAfeDJ with implementation strategies, and discusses how SAfeDJ addresses the research challenges of situations-aware music delivery for drivers. Section 4 demonstrates the mobile application of SAfeDJ, and evaluates the performance of SAfeDJ through a set of practical experiments and user studies. Section 5 concludes the paper.

2. BACKGROUND

In general, the current mobile music recommendation systems mainly follow three approaches [Adomavicius et al. 2005]: (i) *Collaborative filtering (CF)*; (ii) *Content-based*; and (iii) *Hybrid*. *CF* is based on the assumption that if two users have similar music preferences, then the songs preferred by one user will also be recommended to the other. A problem with the current *CF* approach is that it could not recommend new songs to users if such songs have not previously received any comments from any user [Su et al. 2010]. In contrast, the *content-based approach* recommends songs to user by calculating the similarity between songs. For example, if a user likes a particular song, then other similar songs are recommended to the user. However, as discussed in Section 1, driving situations may also have significant impacts on the choice of preferable songs, e.g., a song may be preferable to a user when he is driving on an open road but not in when he is stuck in a traffic congestion. Also, both the current *CF* and *content-based*, and even the *hybrid approach* (which combines *CF* and *content-based*) would be difficult to predict and recommend songs to individual drivers when they encounter new driving situations.

As a special form of crowdsourcing [Howe 2006], mobile crowdsensing leverages human intelligence/experience from the general public to collect, process, and aggregate sensing data using individuals' mobile devices to solve specific problems collaboratively [Hu et al. 2014; Hu[2] 2015]. Mobile crowdsensing involves participatory sensing or opportunistic sensing at the two ends: participatory sensing requires the active involvement of individuals to contribute sensing data related to some large-scale phenomenon, and opportunistic sensing is more autonomous and involves minimal user involvement [Ganti et al. 2011]. In this paper, we propose a novel crowdsensing-based approach to achieve situation-aware music delivery for drivers. Similar to the *hybrid approach*, we involve participants from the general public to provide feedbacks on several representative songs via participatory sensing. Based on the feedbacks, we explore the relations between the attributes of such songs and their corresponding music-moods, and then set up the baseline for similarity calculation of any new songs. However, different from the *hybrid approach*, we further leverage the computing and sensing capacity (i.e., front camera) of smartphone and the sensors like OBD in cars to automatically analyze the impact of music to drivers in different situations, and recommend music to drivers considering both their listening behaviors and current driving situations. Furthermore, similar to opportunistic sensing, we enable automatic sharing of such data (i.e., the drivers' mood-fatigue changing history with information of recommended song and location) in an interactive manner. Also, we design novel data statistics and similarity

computing methods to orchestrate the interactive data in cloud, so as to achieve intelligent recommendations of preferable music to drivers in new driving situations.

In addition, cloud computing is emerging as a new trend that involves the provision of elastic services (i.e., video-on-demand service) over networks [Sardis et al. 2013]. Today, migrating traditional mobile services to the cloud is becoming increasingly important, as doing so can achieve seamless execution of resource-intensive multimedia applications on mobile devices through parallel and distributed computing techniques [Ahmed et al. 2015; Ahmed[2] et al. 2015]. In SAfeDJ, we adopt a cloud platform mainly for two purposes: (i) working with the smartphones to process and identify the attributes of new songs and their corresponding music-moods, to achieve higher efficiency of music recommendation and decrease the battery consumption of smartphones, and (ii) working as a central coordinating platform to aggregate and store the sensing data from diverse drivers, to facilitate the orchestration of such data from different participants, and hence improving the intelligence of the crowdsensing-based music recommendation for individual drivers.

3. CROWD-CLOUD CO-DESIGN APPROACH

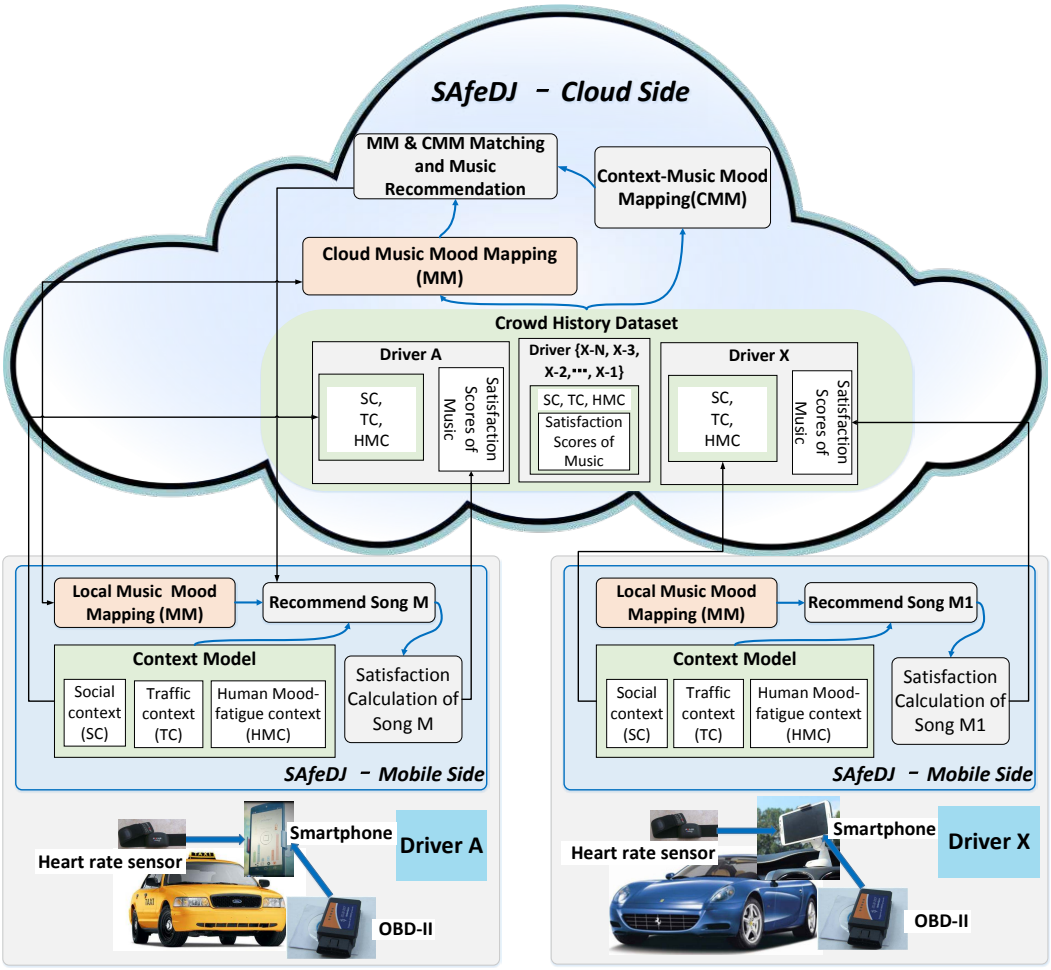

Fig. 1. System architecture of SAfeDJ

The system architecture of SAfeDJ is shown in Fig. 1, in which we can observe the structure and key components of SAfeDJ. Correspondingly, Fig. 2 shows the overall workflow of SAfeDJ. In this section, we first propose our ***music-mood mapping (MM)*** methods in Section 3.1, which could process and identify the attributes (e.g., tempo, and tonal type) of songs and their corresponding music-moods for smartphones in an effective manner. Then, in Section 3.2, we introduce a new ***context model*** designed for smartphones, which could automatically collect and analyze data from multiple sensors, and infer the real-time mood-fatigue degrees of drivers. Finally in Section 3.3, we present our ***context-music mood mapping (CMM)*** methods for music recommendation, and introduce a new method to automatically calculate and update the satisfaction level of the recommended songs.

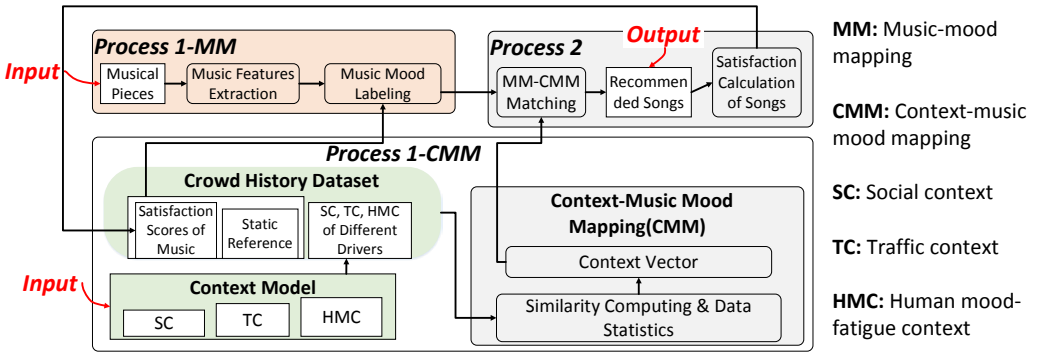

Fig. 2. Overall work flow of SAfeDJ

3.1 Music-mood Mapping (MM)

To recommend suitable music that fits the drivers' real-time moods, we first need to identify the corresponding music-moods that each song expresses. The main idea of ***MM*** is to process each musical piece inside a user's smartphone. For each piece of music, ***MM*** extracts a six-dimensional musical feature, classifies it to one of the existing clusters generated by some well selected representative pieces of music, and calculates a mood label based on the music representatives in that cluster. The final output of ***MM*** is a dictionary with song titles as keys and their mood labels as values. As shown in Fig. 2, ***MM*** consists of two processes: (i) Music feature extraction, and (ii) Music mood labeling.

3.1.1 Music feature extraction

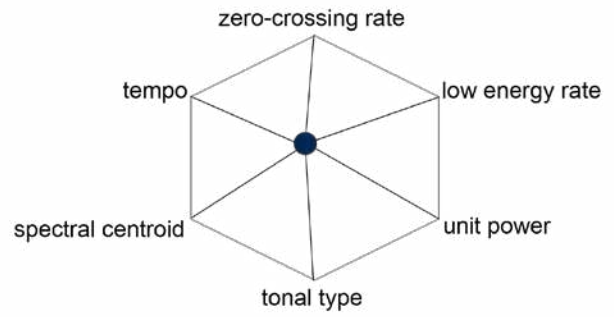

Fig. 3. Six-dimensional music representation

Here we define a six-dimensional music representation as shown in Fig. 3. It is based on music representations of well-known music genre classification works [Tzanetakis et al. 2002]. These six dimensions are ***zero-crossing rate***, ***unit power***, ***low energy***

rate, ***tempo***, ***spectral centroid***, and ***tonal type***. The first four are temporal features and the last two are spectral features. Accordingly, each piece of music can be represented as a point in the six-dimensional space.

The feature extraction starts from reading the digitized musical piece, which can be in any popular format (such as wav, aiff, aac, mp3, etc.), followed by combining the two channels into one, normalizing it and storing it into an array. With this monaural music array and the audio sampling rate, the ***zero crossing rate, unit power,*** *and* ***low energy rate*** could be calculated as follows [Tzanetakis et al. 2002]:

Let $x =$ music array,

$$\textbf{zero crossing rate} = \frac{1}{N}\sum_{n=1}^{N}|sign(x[n]) - sign(x[n-1])| \quad (1)$$

$$\textbf{\textit{unit power}} = \frac{1}{N}\sum_{n=1}^{N} x[n]^2 \quad (2)$$

$$\textbf{low energy rate} = \frac{\#\text{ of point k with x[k]}^2 < unit\ power}{length\ of\ x} \quad (3)$$

Tempo: Based on a well-established tempo tracking algorithm as described in [Tzanetakis et al. 2002], we design and implement a new music tempo module. The algorithm of this module goes in several steps. First we take a down-sampled copy of the absolute value of the music array with down-sample rate 32:

$$y = \text{downsample}(|x|, 32) \quad (4)$$

Then we compute the autocorrelation of the above output, and limit its minimum value to 0:

$$y'[k] = \max(\sum_n y[n]y[n-k], 0) \quad (5)$$

Then we take an approximately 4 seconds section of the above output:

$$y'' = y'[\frac{fs}{32\times4} : \frac{4\times fs}{32}] \quad (6)$$

where 32 is the down-sample rate. The above output is fed to a 50^{th}-order one dimensional median filter for smoothing. We then calculate a 5^{th} order polynomial fitting of the above output, and subtract the post-fit data from the pre-fit data, followed by a normalization process. The resulting data should contain clear peaks whose positions indicate the tempo of the original musical piece. Then we apply a peak picking algorithm to the data, with minimum peak height of 0.5 and minimum peak distance fs / (32×4) as constraints. With a scaling factor to transform the current scale back to the beats per minute (bpm) scale, the first peak position in the bpm scale is the tempo in question.

Spectral centroid*:* The original music array in time domain needs to be transformed into frequency domain before calculation. The transformation is performed by applying discrete Fourier transform to the array:

$$X_k = \sum_{n=0}^{N} x_n\, e^{-\frac{i2\pi kn}{N}}, k \in Z(integers), \quad (7)$$

$$where\ \ N = 2^t, t \in Z, such\ that\ 2^{t-1} < (length\ of\ x) \le 2^t.$$

Then we take the single side amplitude spectrum by taking the absolute value of the first half of X.

$$X' = |X\left[0:\frac{N}{2}\right]| \quad (8)$$

Also, the original linear spaced number scale is updated to linear frequency scale:

$$f_k = \frac{fs*k}{N}, k \in [0, \frac{N}{2}] \quad (9)$$

With these, the *spectral centroid* can be calculated as:

$$\textbf{spectral centroid} = \frac{\sum_k X'_k * f_k}{\sum_k X'_k}, k \in [0, \frac{N}{2}] \qquad (10)$$

Tonal type: The last feature tonal type indicates the overall color of a musical piece. Generally there are 3 tonal types, namely, major, minor, and none [Gómez 2006]. To calculate tonal type, we first transform the amplitude spectrum into a chromagram, which is a representation of the accumulated strength of every musical pitch classes within a given music piece signal [Pauws 2004]. To get a chromagram, the frequency bins of amplitude spectrum below 20Hz is cut, and then all the bins' frequency dimensions are mapped to the midi number dimension,

$$\mathrm{M_k} = round(12 * \log_2\left(\frac{f_k}{440}\right) + 69) \qquad (11)$$

where 69 is the midi number of the *A* above middle *C* (frequency = 440 Hz). This *A* note is provided as the standard reference for converting frequency number to midi number [Fujishima 1999]. The midi numbers are then mapped to midi pitch classes (MPCs) simply by taking modulo 12.

$$\mathrm{MPC_k} = mod(M_k, 12) \qquad (12)$$

The 12 bin chromagram is finalized by summing up values of all MPCs with the same pitch class. After we have a 12 bin chromagram, we proceed to calculate tonality by applying a tonal gravity idea, which is based on the idea describe in [Pauws 2004]. If the tonal gravity is at pitch x, and tonal type is major, then the sum of the chromagram at bins located at x major scale will have the maximum value among all scales. The same is also true for the tonal gravity of minor tonal type.

$$\mathrm{major[k] = chroma[k] + chroma[(k+2)\%12] + chroma[(k+4)\%12] + chroma[(k+5)\%12] + chroma[(k+7)\%12] + chroma[(k+9)\%12] + chroma[(k+11)\%12]} \qquad (13)$$

$$\mathrm{minor[k] = chroma[k] + chroma[(k+2)\%12] + chroma[(k+3)\%12] + chroma[(k+5)\%12] + chroma[(k+7)\%12] + chroma[(k+8)\%12] + chroma[(k+10)\%12]} \qquad (14)$$

Note that major [(k+3) % 12] = minor [k], where "%" is the modulo operator. We compare the chroma bin value maximum entry of both major and minor, take (b+3) % 12 and b for example, if chroma [(b+3) % 12] > chroma [b], the tonality will be a major at (b+3) % 12, if chroma [(b+3) % 12] < chroma [b], the tonality will be minor at b, otherwise, there will be no tonal gravity nor tonal type.

3.1.2 Music mood labeling

ALGORITHM 1. Music Mood Labeling

Input: Musical pieces.
Output: A [song title, mood label, music feature] dictionary of the musical pieces.
Initialize: static references = 21 clusters centered at 21 music representatives' feature points in 6D space, all of the 21 points are initialized with mood labels; distance threshold = DT,
for each musical piece in this music library: p = this musical piece; fe = music feature extract(p)

1. **if** distance of p and every references(static+dynamic) is larger than DT
   create a new cluster centered at p, and assign mood label of nearest point to p
2. **else** cluster p to the nearest cluster
   update the cluster centroid to be the mean of every points in this cluster
   assign mood label to p using the mean of mood labels in this cluster
3. **async task:** dynamic references = a new music point whose mood label is confirmed by crowd history

Algorithm 1 describes the process of music mood labeling.

Static reference: Our final goal is to assign a mood label to each musical piece going through this music mood mapping process, but a big question is where the "mood" comes from. Just as can be seen in a lot of online music channels, human tends to affiliate mood labels to music [Last 2014], such as "groovy", "sad", "happy", "lively", "energetic", etc. These are evidences showing that music has not only "musical" dimensions, but also "mood" dimensions. Also, there are underlying connections between these two sets of dimensions [Hu 2010]. In order to establish a reference point that could serve as the ground truth of mood representation of music, we first set up also a six-dimensional space representing music mood based on previous music mood researches [Hu 2010]. The representation is [lively, energetic, sentimental, groovy, noise, peaceful], or simply, [L,E,S,G,N,P], each with a value ranging from 0 to 1. Then we manually chose 21 representative songs, each based on our own humanized judgment on its mood being dominated by only one dimension or only two dimensions (totally 21 possibilities). After that, we involved 224 participants from the general public in our online survey [Music 2014], and they were asked to vote on each song the dominant moods, given the [L,E,S,G,N,P] multiple selections. After voting, a six-dimensional mood label is generated at each entry, showing the percentage of positive votes for the song's [L,E,S,G,N,P] label respectively. Note that we do not assume that the voting results will be as the same as what we initially think these songs might be, but our effort is only to try to make sure that these 21 songs can represent as many as 21 different musical classes. Thus these 21 songs serve as a static reference, which refers to 21 static music clusters, each of which contains one point in the six dimensional music feature space that is attached with a confirmed mood label in the six-dimensional mood feature space.

Music clustering: When a piece of music is input to this process, it is first reduced to a 6-dimensional music feature using the process described in Section 3.1.1, and then this feature is compared to the centroids of the existing clusters. The centroid with smallest Euclidean distance wins and this new input is grouped into that cluster.

Dynamic reference: Considering that the 21 songs used in the static reference may not represent all the possible styles of different music precisely, we also adopt the scheme of dynamic reference. From Fig. 2, we can observe that there is a process called *satisfaction calculation of songs* (details of which will be introduced in Section 3.3.2), which provides inputs to the dynamic references. For example, if there is a song title in crowd history with a high satisfaction score (e.g., >0.8), and it has already been recommended 10 times, it is considered as a dynamic reference point, which can be added to the reference pool together with its mood label in crowd history. If it is already in the reference pool, only the mood label is updated. Once this dynamic reference is added, it is also being clustered as described in the above subsection.

A New cluster: Similar to dynamic reference, a new cluster is created with center at this point and the mood label of its nearest point is assigned to it, when a new input is too far away from (e.g., distance of overall music features more than the smallest distance between the static references) all the static references. Also, this point is counted as one of the static reference points.

Mood calculation: The mood label of a new input is assigned as the mean of all the reference points' mood labels of the cluster it belongs to. Simultaneously, all the mood labels in that cluster will be refreshed whenever a new reference point is added to an old cluster.

Music mood dictionary – final output

After all the above steps, a musical piece is mapped to a mood label. Thus the final output of this process applying to a smartphone running SAfeDJ is a music mood dictionary (MMD) with song titles as keys and their mood labels - S_{songi} as values as shown in Table 1.

Table I. Music mood dictionary (MMD)

user_id	Song Title	S_{songi} - music_mood [L,E,S,G,N,P]	music_features
0	American Idiot -Green Day	[0.434783, 0.913043, 0.086957, 0.347826, 0.173913, 0]	[0.152975,0.110374,0.698956,0.575,0.165831,1]
0	Better That We Break-Maroon	[0.304348, 0, 0.565217, 0.043478, 0, 0.347826]	[0.157671,0.093225,0.715329,0.49375,0.201159,1]
0	Bring Me to Life-Evanescence	[0.086957, 0.304348, 0.608696, 0.173913, 0.304348, 0.173913]	[0.143611,0.09379,0.711815,0.78125,0.167227,-1]
0	Dancing Queen-ABBA	[0.695652, 0.521739, 0.130435, 0.521739, 0.086957, 0.130435]	[0.163954,0.064364,0.703567,0.61875,0.163235,1]
0	Don't Know Why-Norah Jones	[0.086957, 0, 0.521739, 0.086957, 0, 0.782609]	[0.075943,0.021805,0.790892,0.36875,0.109837,1]
0	…	…	…

Furthermore, considering the intensive computing overhead of ***MM*** on smartphones, we deploy the ***MM*** both on smartphone and SAfeDJ cloud. Then the cloud could assist smartphones to run ***MM*** and directly return the results (i.e., music dictionary shown in Table I) to the smartphones, so as to improve the efficiency and decrease battery consumption of smartphones when running SAfeDJ. Some experimental results about this will be presented later in Section 4.1 and Section 4.4.

3.2 Context Model

The ***context model*** is designed for smartphones to automatically collect and analyze data about drivers from multiple sensors in real-time. This model is shown by the green boxes in Fig. 1 as input data for each car (or driver). It includes three major categories: *social context (SC), traffic context (TC),* and *human mood-fatigue context (HMC)*. The above contexts can be obtained from different kinds of sensing data, and then used as key contexts to support the predictions of drivers' music preferences while driving as presented in Section 3.3.

***SC*:** From our online surveys [Driving 2014], we observe that the probabilities of two drivers favor the similar style of music in the same driving context are higher when their *SC* (i.e., gender) are closer. Thus, in our current design of SAfeDJ, the data about *SC* is obtained from the user profiles of Facebook, which includes the *cultural background*, *gender*, *age*, and *driving experience* of a driver. Also, it includes the interaction history of a driver with friends in his social circle. In addition, we plan to integrate the data from the professional social networking music websites like SoundCloud and Deezer, so as to further analyze the users' music interest pattern in the future evolution of SAfeDJ.

TC: It expresses information about the traffic condition that a driver is situated in. This includes the *location*, *weather*, *road surface condition*, *traffic congestion*, etc. The location data can be obtained automatically by a smartphone using its onboard Global Positioning System (GPS) receiver, while the weather and traffic related information can be obtained from the Internet accordingly. The data about *TC* can also be acquired by aggregating traffic condition data reported by individual drivers.

HMC: We design a mood-fatigue detector in *HMC*, which can collect and analyze multiple sensing data on a smartphone to infer the real-time mood-fatigue degrees of the driver. As shown in Fig. 4, the mood-fatigue detector mainly considers the data: (i) *engine rpm (RPM)*, *throttle input (Tht)* and *driving speed (Sp)* from OBD-II; (ii) *heart rate (Hr) and in-vehicle temperature (T)* from wearable sensors; (iii) *facial expression* and *blinking frequency of the driver's eye-lid (EI)* from the front camera of

smartphones; and (iv) *road congestion situation* from online traffic service (e.g., Google map). Based on the above sensing data, we develop a process to analyze the mood-fatigue degrees in the detector. In this process, the status of a driver's mood are classified into six dimensions: *aggressiveness, patience, neutral, happiness, sadness,* and *anger*, and then the degrees of the six dimensional moods and fatigue are inferred and quantified as float data (from 0 to 1.0).

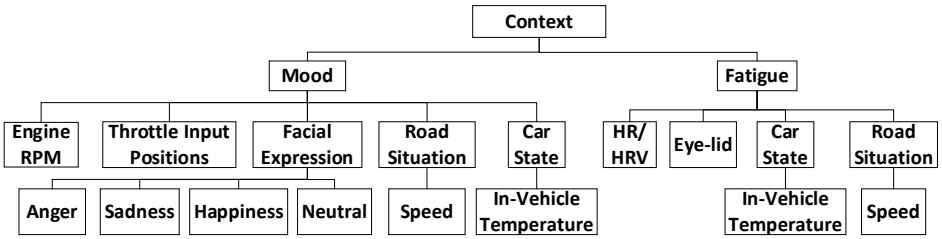

Fig. 4. Structure of the data used in the driver's mood-fatigue detector

Table II. User history table (UHT) in the crowd history dataset

Hist._id	user_id	fatigue_deg.	aggressive_deg.	neutral_deg.	happy_deg.	sad_deg.	angary_deg.	patience-deg.	longitude	latitude	music scores-S_x[L,E,S,G,N,P]	timestamp
1	23673	0.8	0.4	1	0	0	0	0.6	122.085	37.422	[0.71,0,0.1,0.9,0,0.65]	16:31:30
2	14655	0.2	0.8	0.8	0	0	0.2	0.2	120.666	38.523	[1,1,0,1,0,0]	16:32:31
3	17887	0.6	0.6	1	0	0	0	0.4	123.785	37.123	[0,0,1,0,0,0]	16:33:33
4	23673	0.4	0.3	0.1	0.9	0	0	0.7	122.133	37.333	[1,0.2,0,1, -0.2,0.87]	16:35:35
...	...	...	...	...	...	...	...	...	...	...	...	...

Finally, after all the data about *SC, TC* and *HMC* have been collected and contextualized, we design a mechanism that enables the smartphone of each driver to automatically synchronize such contexts to the crowd history dataset (see Fig. 1) in SAfeDJ cloud 5s before the end of each song being played. An example about the user history table (UHT) of such dataset is shown in Table II.

In addition, preserving the privacy of the users, e.g., in terms of their geographical locations and personal music preference, is always a concern. Different approaches have been proposed to provide privacy-preservation by transforming the data, e.g., adding noise [H. Ahmadi et al. 2010]. In the context of location privacy, it ensures that the user's location is indistinguishable from at least K-1 other users. To achieve K-anonymous location privacy, one common approach is to incorporate a trust server that is responsible for removing the user's ID and selecting an anonymizing spatial region containing the user and at least K-1 other users [K. Vu et al. 2012]. Even though the current implementation of SAfeDJ does not incorporate any privacy preserving mechanism yet, these cloaking and anonymization techniques can be applied to provide privacy preservation on the geographical location and music interests of users in our system in the future.

3.3 Context-music Mood Mapping (CMM) and Music Recommendation

As discussed at the beginning of this paper, there are many different possible types of human moods and traffic conditions in a driver's daily life. Therefore it would be difficult for individual smartphones to predict and recommend suitable music to a driver in new driving situations that have not been encountered before. For instance, there may be records that some songs were preferable to a driver driving in mountains with a happy mood, while there is no music record corresponding to situations when this driver is driving in similar mountains but with a sad mood, or

driving on bumpy roads for the first time but with a similar happy mood. In Section 3.2, we have presented the ***context model*** designed for smartphones, which enables an in-car smartphone to automatically collect and analyze the data of an individual driver's *SC*, *TC* and *HMC* with satisfaction scores of songs, and synchronize the processed data to SAfeDJ cloud in real-time. In this part, we further propose the ***CMM*** and music recommendation methods. These methods could effectively analyze and orchestrate the data from different parties in the SAfeDJ cloud, so as to enable different in-car smartphones to help each other and recommend preferable music to individual drivers in their new driving situations.

3.3.1 Context-music mood mapping (CMM)

The main idea of ***CMM*** is to use statistical methods to explore each type (*SC, TC, HMC*) of context's impact on music choice in driving situations, and then use similarity computing methods to calculate the context similarities between multiple drivers to achieve situation-aware music delivery. For example, as shown in Fig. 5, to recommend a song to *Driver A* while he is driving, we can calculate the context similarity between *Driver A* and *Driver X* (from crowd history dataset shown in Fig. 1) in terms of their *SC*, *TC* and *HMC*. If the overall similarity between them is high, it means the probability that *Driver A* and *Driver X* prefer the same style of music in the similar driving situations is high. Then the satisfaction score of music recorded with the specific driving situations from *Driver X* could be used as a reference. Likewise, the similarities between other drivers and *Driver A* could be calculated, and then their satisfaction scores of music could be used together with *Driver X*'s score as overall references for music recommendation to *Driver A* in his new driving situations.

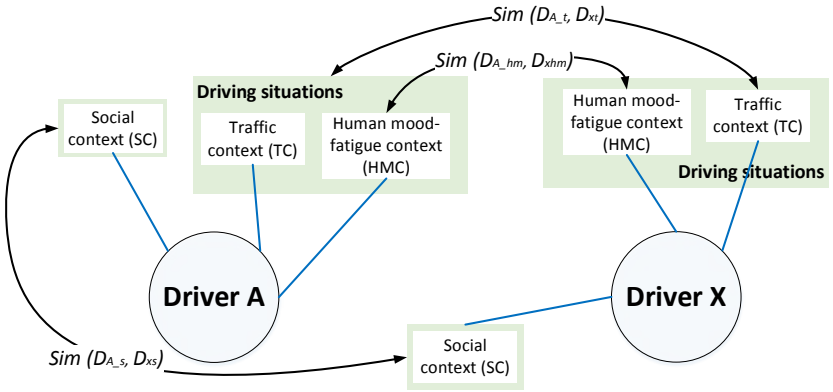

Fig. 5. Example of similarity computing in CMM

A. Data statistics and similarity computing

In ***CMM***, the context similarity between two drivers like *Driver A* and *Driver X* is initially defined as $sim\_{ini}(D_A, D_X)$ in equation (15), where $Sim(D_{A\_S}, D_{XS})$ is the *SC* similarity, $Sim(D_{A\_t}, D_{Xt})$ is the *TC* similarity, and $Sim(D_{A\_hm}, D_{Xhm})$ is the *HMC* similarity, and W_s, W_t, W_{hm} are the weights for *ST, TC*, and *HMC*, respectively. N_{DA_Sinput}, N_{DA_tinput}, $N_{DA\_hminput}$ are the number of items about *SC*, *TC*, and *HMC* from *Driver A*, respectively, and N_{DX_Sinput}, N_{DX_tinput}, $N_{DX\_hminput}$ are the number of items about *SC*, *TC*, and *HMC*, respectively, which could be provided in the historic record of *Driver X*. Finally, the overall context similarity $sim(D_A, D_X)$ between these two drivers is calculated as the weighted average similarity shown in equation (19).

$$sim\_{ini}(D_A, D_X) = \sum_{j=1}^{N1} Sim(D_{A\_s}, D_{X_s})W_s + \sum_{j=1}^{N2} Sim(D_{A\_t}, D_{X_t})W_t + \sum_{j=1}^{N3} Sim(D_{A\_hm}, D_{X_{hm}})W_{hm} \quad (15)$$

$$N_1 = min(N_{D_{A\_s_{input}}}, N_{D_{Xs_{input}}}) \quad (16)$$

$$N_2 = min(N_{D_{A\_t_{input}}}, N_{D_{Xt_{input}}}) \quad (17)$$

$$N_3 = min(N_{D_{A\_hm_{input}}}, N_{D_{Xhm_{input}}}) \quad (18)$$

$$sim(D_A, D_X) = \frac{sim\_ini(D_A, D_X)}{N_1 * W_S + N_2 * W_t + N_3 * W_{hm}} \quad (19)$$

To define the values of W_s, W_t, W_{hm}, we need to explore and quantify the impact of *SC, TC*, and *HMC* on music choice while driving. Thus, we set up an online survey [Driving 2014], in which 505 participants with driver licenses were involved globally. Based on the data collected from this survey, we adopt the professional data statistics software *Stata 13* [Stata 2014] to analyze the data. In addition, we also use these data as the initial dataset of the crowd history dataset in SAfeDJ cloud, so as to avoid the potential cold-start problem (i.e., no user records of SAfeDJ at the very beginning) [Ness et al. 2009]. The experimental setting and outputs of *Stata 13* are shown in Table III. From this table, we can observe that with the stated 95% confidence intervals, the average coefficients of *SC, TC* and *HMC* are 0.347, 0.254, and 0.956, respectively. The higher coefficient value means the higher impact of the corresponding type of context on music choice. Thus, from these results, and $W_s+W_t+W_{hm}$=1.0, 0.374+0.254+0.956=1.584, we can define: W_s=0.374/1.584=0.223, W_t=0.254/1.584=0.163, W_{hm}=0.956/1.584=0.614. Note, as we have discussed in Section 3.2 that *TC* may impact the mood-fatigue of drivers and hence indirectly impact their preference of music, while here we only define the direct impact of *TC* to drivers' music choice.

Table III. Statistics about the impact of context to music choice in driving situations

Experimental settings: Statistics software: Stata 13; Logistic regression; Number of obs = 505; LR chi2(4) = 9.27; Prob > chi2= 0.0547; Log likelihood = -167.08999; Pseudo R2 = 0.0270			
Items	**Coefficient**	**Standard Error**	**[95% Confidence Interval]**
Social context (SC)	0.347	0.266	-0.703, 0.314
Traffic context (TC)	0.254	0.314	-0.672, 0.871
Human mood-fatigue context (HMC)	0.956	0.312	-1.726, -0.188

During the development of SAfeDJ, we observe that the data about *SC* (i.e., gender, cultural background) and *TC* (i.e., road condition) are usually string data. While as introduced in Section 3.2, the data about *HMC* have been normalized as decimals. Thus, we design two similarity computing methods: **1).** Ontology-based similarity computing to calculate the similarities of *SC - Sim*(D_{A_S}, D_{XS}), and *TC - Sim*(D_{A_t}, D_{Xt}), respectively; and **2).** Euclidean similarity computing to calculate the similarity of *HMC - Sim*(D_{A_hm}, D_{Xhm}).

1) Ontology-based similarity computing

Normally, the data about *SC* and *TC* are string data and they usually have implicit contexts. It would be difficult for the traditional keyword based matching (e.g., location based in [Baltrunas et al. 2012]) to explicitly compare the similarity between two contexts of *SC* and *TC*. For example, regarding the *SC*, directly comparing the two words *"Canada"* and *"UK"* as two drivers' cultural backgrounds would return a similarity of 0. Similarly, for *TC*, directly comparing the two places *"Hana Road"* and

"Hamakua Coast", which are totally different words and locations, would also return a similarity of 0, while their views and road conditions are actually similar. Thus, we adopt the ontology-based similarity computing to calculate the similarities of *SC* and *TC* between drivers. In the research community, there already exist several popular ontologies, e.g., *WordNet* [Pedersen et al. 2004] with a large lexical database of English, *FOAF* [Brickley et al. 2012] that descripts the social relations between people, and *geo.owl* [Battle et al. 2012] that defines geographic concepts. Considering the high efficiency requirements in the driving situations, we first adopt labeled links to establish relationships between a context and its semantics, so as to enable multiple collected data of *SC* and *TC* to be used to compute the context similarity. Then, we design a lightweight and scalable similarity computing method in ***CMM***, which can make use of the above and other existing ontologies to calculate the similarities between two context items of *SC* and *TC* in an effective and efficient manner.

In a conceptual graph (i.e., the common ontology), three aspects impact the semantic similarity of two concepts (items): the distance (the length of the path) between two items, the depth of two items and the depth of their most specific common parent in the common ontology, and whether the direction of the path between the two items is changed. The semantic similarity is calculated as:

$$Sim(I_1, I_2) = 2C^{(k+p+d)} depth_{common\_ancestor} / (depth_{I_1} + depth_{I_2}) \qquad (20)$$

where *common_ancestor* is the most specific common parent item; *depth* represents the depth of an item in the ontology; k defines the length difference of the two paths between the most specific common parent item and the two items; p defines the path length between two items; d defines the changes of the directions over the path; and c is a constant between 0 and 1. After the similarity of each item about *SC* and *TC* has been calculated, we can get the $\sum_{j=1}^{N1} Sim(D_{A\_s}, D_{X_s})$ and $\sum_{j=1}^{N2} Sim(D_{A\_t}, D_{X_t})$ based on equations (15), (16) and (17).

2) Euclidean similarity computing

In Section 3.2, the contexts of *HMC* have been defined in seven dimensions (one dimension of fatigue and six dimensions of moods): *fatigue-degree, aggressive-degree, neutral-degree, happy-degree, sad-degree, angary-degree, patience-degree*; and each dimension is normalized as decimals between 0 and 1. Thus, we just need to calculate the Euclidean distance of each dimension between two drivers' *HMC* shown in equation (21), and then sum up the results in equation (22) to get the $\sum_{j=1}^{N3} Sim(D_{A\_hm}, D_{X_{hm}})$.

$$Sim(D_{A\_hm}, D_{X_{hm}}) = 1 - \sqrt{(D_{A_{hm}} - D_{X_{hm}})^2} \qquad (21)$$

$$\sum_{j=1}^{N3} Sim(D_{A\_hm}, D_{X_{hm}}) = \sum_{j=1}^{7} (1 - \sqrt{(D_{A_{hm\_j}} - D_{X_{hmj}})^2})/7 \qquad (22)$$

B. Context vector

After calculating the similarities between *Driver A* and all the other drivers, we could select the most suitable drivers with specific driving situations recorded in the crowd history dataset in SAfeDJ cloud as context representatives for music recommendation to *Driver A*. To avoid the possible random and subjective factors of single driver that may impact the choice of music, we need to select more than one representative. Also, to make sure the music could be delivered in time in highly dynamic driving scenarios, we need to limit the number of representatives. Thus, in our design, we select the most similar driving situation of *Driver A* himself (i.e.,

recorded a few minutes ago in time *t*) and marked as D_{At} plus the records of the other ten drivers as representatives.

Furthermore, to make sure that the summed up contexts of the other ten representatives could be most similar to *Driver A*'s current context, we define a context vector:

$$\overrightarrow{sim(D_A, D_{sum})} = [\sum_{x=1}^{10} \overrightarrow{Sim(D_{A\_s}, D_{X_s})} W_s, \sum_{x=1}^{10} \overrightarrow{Sim(D_{A\_t}, D_{X_t})} W_t, \sum_{x=1}^{10} \overrightarrow{Sim(D_{A\_hm}, D_{X_{hm}})} W_{hm}] \quad (23)$$

in which the $\left|\overrightarrow{sim(D_A, D_{sum})}\right|$ means the similarity between *Driver A* and the summed up context of the other ten representatives, and it should be as large as possible. Also, $|\sum_{x=1}^{10} \overrightarrow{Sim(D_{A\_s}, D_{X_s})}| \leqslant 1.0$, $|\sum_{x=1}^{10} \overrightarrow{Sim(D_{A\_t}, D_{X_t})}| \leqslant 1.0$, $|\sum_{x=1}^{10} \overrightarrow{Sim(D_{A\_hm}, D_{X_{hm}})}| \leqslant 1.0$, since the highest similarity is 1.0 as defined in equation (19). The process to choose the ten representatives for music recommendation is shown in Algorithm 2, in which it computes the similarities of all context items and compares the constitutes of them, so as to select the maximum $\left|\overrightarrow{sim(D_A, D_{sum})}\right|$.

ALGORITHM 2. Finding the Ten Most Suitable Context Representatives

Input: Similarity items of each context representative candidates.
Output: Ten context representatives
1. **For** each x, compute all the similarity terms of driver x with driver A, and sum up all similarity items as sim(x)
2. **Sort** sim(x) in descending order
3. **Mark** sim(0)
4. **From** i = 1 on, observe individual terms in sim(x), **if** the constitution of sim(x) is similar to sim(x-1), then throw away sim(x), **else**, mark sim(x)
5. **Repeat** 3 until there are 10 sim(*) are marked

Once the D_{At} and the other ten context representatives are selected, since each of them has the corresponding history satisfaction scores of music - S_x (has six dimensions [L,E,S,G,N,P]) along with the driver's specific driving situations (see Table II in Section 3.2), then we can get the overall music recommendation score - S_{AR} for *Driver A* through:

$$S_{AR} = (S_{At} * mSim(D_A, D_{A_t}) + \sum_{x=1}^{10} S_x * Sim(D_A, D_x))/(mSim(D_A, D_{A_t}) + \sum_{x=1}^{10} Sim(D_A, D_x)) \quad (24)$$

In equation (24), we can observe that the higher overall similarity $Sim(D_A,D_X)$ between the *representative X* and *Driver A*, then the higher weight of S_x in S_{AR}. Also, usually the history of *Driver A* himself has more reference value than others, thus we put a constant value *m* in this equation and set its default value as 2, and we plan to let the users of SAfeDJ configure *m* to suit their needs. In addition, if there is a disconnection between the smartphone and the SAfeDJ cloud while driving, the context representative will consist of only D_{At}, then S_{AR}=S_{At}.

3.3.2 Music recommendation and satisfaction calculation of songs

As discussed in Section 3.1, each song belonging to a driver using SAfeDJ could be processed and calculated in the six dimensions to get the music-mood - S_{songi} ([L,E,S,G,N,P]). Thus, we just need to compare S_{AR} and each S_{songi} in the MMD (referring Table I in Section 3.1.2) of *Driver A*, then the song with the minimum $|S_{songi} - S_{AR}|$ will be selected and recommended to *Driver A*.

Furthermore, as discussed in Section 3.2, we could use the smartphones to detect and record the real-time mood-fatigue data of drivers and store such data in the UHT (Table II in Section 3.2). As each user of SAfeDJ has a MMD, we could adopt

Algorithm 3 for the smartphones to automatically calculate the satisfaction scores of each song the driver is listening (to obtain the changing trend of a driver's moods during the time while he is listening to the song), and synchronize such scores to SAfeDJ cloud to generate/update the historic satisfaction score of the music - S_x in real-time, i.e., data shown in the right side of Table II.

ALGORITHM 3. Satisfaction Calculation of Songs and Music Scores Update

Input: start time of the song, end time of the song, S_{songi} ([L,E,S,G,N,P]) in the MMD, UHT-mood-fatigue history data within the time range of the song, and the history satisfaction scores of music - S_x [L,E,S,G,N,P]
Output: a score [-1,1] indicating the level of satisfaction the user towards the song, and updated S_x -[L,E,S,G,N,P]
Positiveness definition: happy +2, neutral +1, patient +1, not aggressive +1, sad -1, angry -2, not patient –1, aggressive -1 (with '+' indicates positive and '-' indicates negative)
1. range = [start of song : end of song]>[0:n-1]; **for** k in [0:n-1]
p_k= happy*2 + neutral*1+ patient*1+not aggressive*1+sad*(-1)+angry*(-2)+not patient*(–1)+aggressive*(-1) (all of point k)
for k in [0:n-1]
Compute d = $(p_{(k+1)} - p_k)$ - $(p_k - p_{(k-1)})$
If d⩾0, satisfaction += 1, **else**, satisfaction -=1; normalized satisfaction = satisfaction / n
2. Initialize: threshold T = x (x default value is 0.4)
If satisfaction⩾0
satisfaction = satisfaction*T
updated S_x = satisfaction*S_{songi} + S_x
Else updated S_x = satisfaction*S_{songi} + S_x
Limit updated S_x to [-1,1]

4. EXPERIMENTS AND USER STUDIES

In this section, we evaluate SAfeDJ in terms of four aspects: (i). Accuracy and efficiency of ***MM***; (ii). Effectiveness of the mood-fatigue detector in the ***context model***; (iii). Impact of the music recommended by SAfeDJ to drivers; and (iv). Overhead of SAfeDJ in popular smartphones.

4.1 Accuracy and Efficiency of MM

Similar to our online survey method presented in Section 3.1.2, we randomly picked 10 songs which cover different styles of music and involved 183 volunteers globally to listen to and vote on each song with respect to the dominant moods, and then calculate the voting for the percentage of each song's [L,E,S,G,N,P]. Simultaneously, we used our ***MM*** methods to calculate the [L,E,S,G,N,P] of each song and compared the results with the respective results from the online survey. The closer (smaller distance) are the results from these two methods, the higher is the accuracy of our ***MM*** methods in the identification of music-mood in real-world. In addition, each song's music-mood dimension has a maximum distance value 1.0, and the corresponding maximum value of the overall distance of each song is 2.45.

The experimental results are summarized in Table IV. It shows that regarding each song's music-mood dimension, the distance is generally around 0.1 between our ***MM*** methods and the results from online voting, and the maximum distance is 0.437 in the ***N*** dimension of the song *FuzzUniverse*. The overall distance is normally around 0.247~0.577 for each song, and 0.419 in average for the 10 songs. Therefore, the results demonstrate that the music-mood dimensions calculated by our ***MM*** methods are close to the results from people's real feeling about these songs in the real-world. Furthermore, as discussed in Section 3.1.2, since our ***MM*** methods

support self-update of the music clustering with music moods, the accuracy will be improved when there are more and more users of SAfeDJ in the future.

Table IV. Case study about the accuracy of MM

Name of Song	Lively(L) |MM-Vote| =Distance	Energetic(E) |MM-Vote| =Distance	Sad(S) |MM-Vote| =Distance	Groovy(G) |MM-Vote| =Distance	Noisy(N) |MM-Vote| =Distance	Peaceful(P) |MM-Vote| =Distance	Overall Distance (Max:2.45)
1984	|0.348-0.439| =0.091	|0-0.041| =0.041	|0.391-0.306| =0.085	|0.044-0.055| =0.011	|0-0| =0	|0.522-0.918| =0.396	0.418
Haoting	|0.696-0.776| =0.080	|0.522-0.306| =0.216	|0.130-0.041| =0.089	|0.522-0.384| =0.138	|0.087-0| =0.087	|0.130-0.459| =0.329	0.443
Manhattan	|0.696-0.449| =0.247	|0.522-0.225| =0.297	|0.130-0.031| =0.099	|0.522-0.500| =0.022	|0.087-0.020| =0.067	|0.130-0.541| =0.411	0.577
WannaBe	|0.261-0.459| =0.198	|0.870-0.776| =0.094	|0.435-0.286| =0.149	|0.435-0.455| =0.020	|0.044-0.286| =0.242	|0.044-0.225| =0.181	0.403
Nizaiganma	|0.609-0.532| =0.077	|0.043-0.165| =0.122	|0.304-0.190| =0.114	|0.478-0.354| =0.124	|0-0.101| =0.101	|0.261-0.139| =0.122	0.273
Huahuo	|0.609-0.190| =0.419	|0.043-0.139| =0.096	|0.304-0.494| =0.190	|0.478-0.203| =0.275	|0-0.013| =0.013	|0.261-0.418| =0.157	0.567
AndIloveher	|0.348-0.076| =0.272	|0-0.051| =0.051	|0.391-0.544| =0.153	|0.043-0.089| =0.046	|0-0.013| =0.013	|0.522-0.671| =0.149	0.353
FuzzUniverse	|0.217-0.430| =0.213	|0.435-0.722| =0.287	|0-0.013| =0.013	|0.304-0.228| =0.076	|0.215-0.652| =0.437	|0-0| =0	0.570
Neverwakeup	|0.217-0.190| =0.027	|0.435-0.316| =0.119	|0-0.063| =0.063	|0.304-0.076| =0.228	|0.652-0.747| =0.095	|0-0| =0	0.247
Waltzfordebby	|0-0.089| =0.089	|0-0.013| =0.013	|0.696-0.392| =0.304	|0.043-0.101| =0.058	|0-0.038| =0.038	|0.783-0.671| =0.112	0.343

Moreover, we conducted experiments about the time efficiency of ***MM*** to analyze music in smartphones. We used 180 songs in mp3 format in our tests, with each song having two bit rates (128kbps and 320kbps). Based on the type of song's bit rate, we divided the tests into two cases, and recorded the average time for music analysis (overall time divides 180) in each case. Also, there are three situations in each case: (i) Operating ***MM*** in SAfeDJ cloud to analyze the music under LTE network under normal driving situations (e.g., around 40~60km/h driving speed in urban areas); (ii) Operating ***MM*** in SAfeDJ cloud to analyze the music under stable WiFi network in non-driving situations; and (iii) Only operating ***MM*** in smartphones to analyze the music. The experimental environment of SAfeDJ cloud was as follows: ***Hardware*** - Amazon EC2 M3 2xlarge Instance; 30 GB memory; 8v EC2 Compute Unit; 2*80 GB SSD storage; Moderate I/O Performance; EBS-Optimized Available: No. ***Software*** - operating system: Windows Server 2008. ***Experimental device*** - Samsung Galaxy S4. Vehicle - 2005 Toyota Sienna.

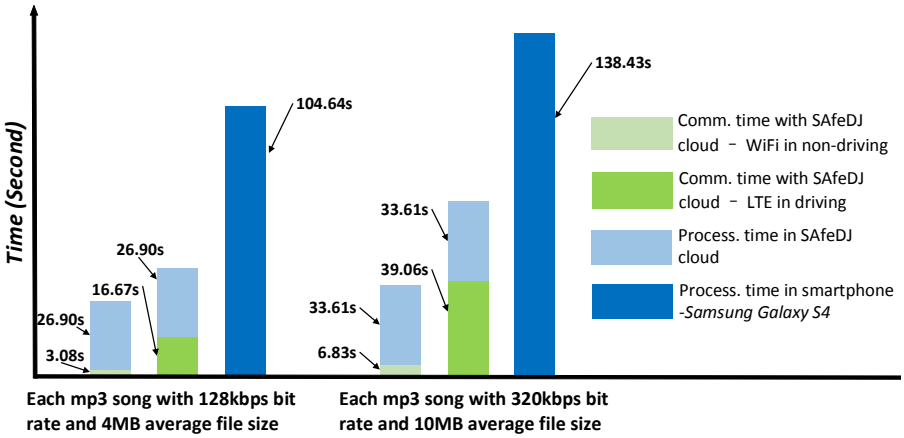

Fig. 6. Time efficiency of MM

The results about the time efficiency of ***MM*** are shown in Fig. 6. We find that the average processing time in SAfeDJ cloud is about 26.9s on each mp3 song with bit

rate of 128kbps and about 4MB file size, while 33.61s when the mp3 song has a bit rate of 320kbps and about 10MB file size. Since the communication time is mainly made up of the time for uploading the mp3 file from Galaxy S4 to SAfeDJ cloud for processing, this time mostly depends on the size of the mp3 file and network condition (WiFi or LTE). Also, we observe that the uploading speed of the mp3 file may decrease occasionally due to adverse channel conditions and latency of handovers encountered during our driving tests. In most cases, however, the uploading speed of SAfeDJ over an LTE network is very stable in normal driving situations, and the maximum delay is 39.06s in our experiments. This is because LTE is the most recent generation of cellular networking technology with a high performance and support for terminal mobility at a speed up to 350km/h [G. Araniti et al. 2013]. In addition, a lot of research works have addressed vertical handovers in heterogeneous wireless networks that support seamless connectivity for smartphones and other terminals in vehicular networks [S. Fernandes et al. 2012], and different mechanisms can be integrated in SAfeDJ to better support vertical handovers in the future. On the other hand, Fig. 6 shows that the average processing time of ***MM*** in Galaxy S4 are 104.64s and 138.43s, respectively, for the two bit rates. Since a song normally plays for more than 3 minutes, thus the probability that ***MM*** can process a new mp3 song in time while a driver is driving and using SAfeDJ should be high.

4.2 Effectiveness of Mood-fatigue Detector in Context Model

To evaluate the effectiveness of the mood-fatigue detector presented in Section 3.2, we involved ten volunteers (five males and five females) on campus to conduct a set of tests. At the same location under normal light, each subject was seated in front of a smartphone (Google Nexus 5) running SAfeDJ with a five-degree angle to simulate the normal driving position. Then the subject was requested to do five tests, each lasting for 300s, to evaluate the effectiveness of detections for: **fatigue** and four moods **- neutral, sad, happy,** and **angry**, respectively.

To test the accuracy of fatigue detection, each subject was asked to pretend that he/she was tired, and express his or her common reactions under such status, like closed or squint eyes frequently. Similarly, in the tests for detections of the four moods, each subject was asked to express the four types of moods in terms of how they may appear as their daily facial expressions. As mentioned in Section 3.2, each item of the mood-fatigue could be quantified as a degree which value ranges from 0 to 1.0. For example, in every ten-second period, if the fatigue status of a subject is detected 7 times and the non-fatigue status is detected 7 times, then fatigue degree of him/her has a value of 7/(7+7)=0.5 in this period. This means that the higher value of the fatigue degree detected in our test, the higher is the effectiveness of our mood-fatigue detector. Also, as the results may be impacted by the sequences of the five tests, e.g., people's faces may get tired or stiffed after expressing a certain mood for a long time, we assigned a random sequence of these tests to each subject to avoid such impacts. Finally, after all the data about the ten people are recorded in each test, the average values of them were calculated.

The experimental results of the five tests are shown in Fig. 7. We can observe that the average detected value of the fatigue degree is around 0.8, which means most of the fatigue expressions (i.e., tired eyes) from the people could be detected by our mood-fatigue detector. Regarding the detections of the four moods, the average value of the neutral mood is around 0.9 indicating a high effectiveness of detection, since this is the most common mood status of people. However, for the detections of happy and sad moods, the detected average values of them are both around 0.6, as the facial

expressions of such moods are similar to neutral expression for some people. As the most diversified expression, the detected average value of angry is around 0.5 in the corresponding test. Moreover, we make a simple comparison of our mood-fatigue detector with a similar approach found in the literature [Suk 2014]. Under similar situations, the effectiveness of their approach using smartphones to detect the moods of drivers is around 0.9 for neutral and happy, 0.3 for sad, and 0.4 for angry, respectively. Even though our approach and theirs are running on different Android devices, a key reason for the better detection of negative moods in our approach is that we lower the threshold in the determination of drivers' negative moods in our algorithm.

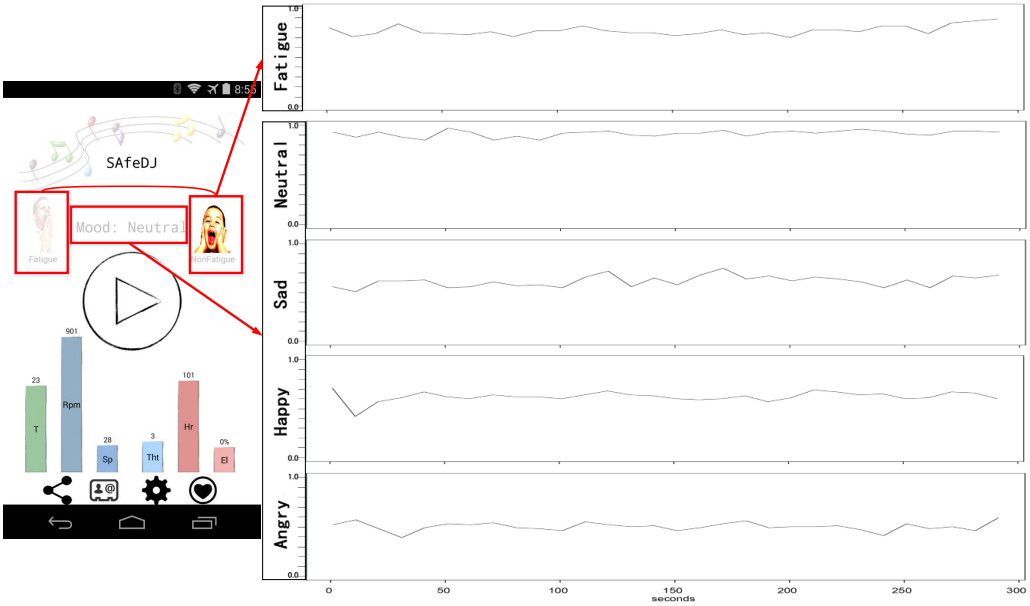

Fig. 7. Experimental results about the detections of mood-fatigue degree

Furthermore, it is worth noting that as presented in Section 3.2, beyond using the front camera of the smartphone to detect the mood-fatigue of drivers like the above tests, our solution also supports using the data from OBD and heart rate sensors simultaneously to facilitate detection of the mood-fatigue of drivers while driving. For instance, the OBD could be used to detect hard accelerations of drivers and infer the aggressiveness and mood-fatigue degrees of them. Also, as demonstrated in the related work [Lee 2012], the effectiveness of prediction of drivers' mood-fatigue degree can even increase to 94% when multiple wearable sensors are used simultaneously with smartphones. Due to the experimental constraints of these tests, e.g., it would be unsafe to require the volunteers to do multiple hard accelerations while driving, we have not evaluated these approaches in practice, but only used front cameras of smartphones for mood-fatigue detections in our tests. It means that in the real-world scenarios during driving, our solution for the mood-fatigue detection could have a higher effectiveness when data from the OBD and heart rate sensors are also incorporated in the detection process.

4.3 Impact of Music Recommended by SAfeDJ to Drivers

Since it would be difficult to involve multiple drivers to use SAfeDJ and drive in the same road and traffic conditions in experimental tests, thus similar to former research in driving behavior analysis [Cassidy et al. 2009; Zwaag et al. 2012], we

adopted a driving simulator as shown in Fig. 8(a) to evaluate the impact of music recommended by SAfeDJ to drivers. In our user studies, we involved 48 volunteers (32 males and 16 females) with driver licenses. Currently, people usually listen to music via smartphones in two ways: (a) Collecting and storing the preferable songs in personal smartphones, and playing such songs on the smartphones directly; or (b) Playing all the songs on the smartphones by accessing popular online music streaming services. Thus, we randomly divided the 48 volunteers into two groups, *group a* and *group b*, each consisting of 24 people. Each driver in *group a* picked 40 songs that were most preferable to him/her and used them in all his/her tests. Drivers in *group b* all accessed the same music library consisting of 180 mp3 songs with different styles in all the tests. In both groups, each driver was required to finish three kinds of tests in the same road and traffic conditions: (i) Not listening music; (ii) Listening to music via the smartphone's (Huawei P7) default music player; (iii) Listening to music recommended by SAfeDJ. Each test lasted for 20 minutes, and the time interval between two tests was about 30 minutes for every driver, so as to ensure that the drivers could drive in the same fatigue and mood status as far as possible in each test. Also, in case the degree of familiarity to the roads may impact a driver's mood-fatigue status, the sequence of the three tests for each driver was randomized.

Fig. 8. Experimental driving scenarios

As shown in Fig. 8(b), when a driver started using SAfeDJ, he could use SAfeDJ to log into his Facebook account, which enables SAfeDJ to get his *SC* like gender, age and cultural background for the initial prediction of his music preference. During the tests, we used the mood-fatigue detector of SAfeDJ to record the mood-fatigue history of every driver every 30 seconds, and configured the detector to synchronize the results with the SAfeDJ cloud automatically. After all the tests were finished, we calculated and compared the average fatigue degree and negative mood degree (sad plus angry degrees) of each group's 24 drivers in the three different kinds of tests.

The comparison results about drivers' fatigue degrees in *group a* and *group b* are summarized in Fig. 9(a) and Fig. 9(b), respectively. We observe that in both groups, the drivers' fatigue degree is slowly increasing in their driving when they do not listen to music or listen to music via the smartphone's default music player. Also, it is interesting that in *group b*, the drivers' recorded fatigue degree is even higher in most of the time periods when they listen to music through the default music player (average value is 0.4435), comparing to not listen to music while driving (average value is 0.3606). The average value of fatigue degree is 0.3166 when using default music player and 0.3607 when non-listening music in *group a*. In contrast, the drivers' fatigue degree is the lowest when they listen to music recommended by

SAfeDJ during driving, and the related average value is only 0.1898 in *group a* and 0.1972 in *group b*, which means it is 49.09% lower than using default music player in overall.

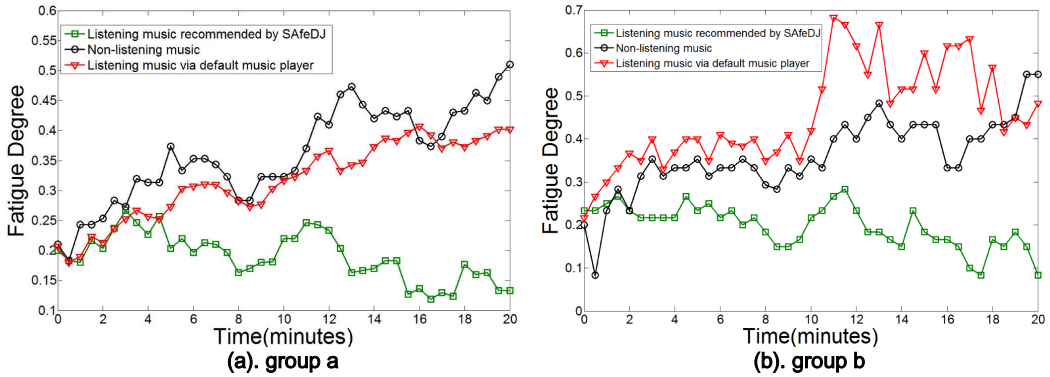

Fig. 9. Comparisons of drivers' fatigue degree

On the other hand, from Fig. 10, we find that the drivers' recorded negative mood degree is relatively complicated. In *group a*, the drivers' negative mood degree is close between using default music player and SAfeDJ. On the other hand, in *group b*, the drivers' negative mood degree may become the highest when they listen to music via the default music player in some time periods, but lower than that from listening to music recommended by SAfeDJ in a few time periods. Nevertheless, in general the recorded negative mood degree of drivers is still the lowest when they listen to music provided by SAfeDJ with an average value 0.0712 in *group a* and 0.0752 in *group b*. In contrast, the average value is 0.0961 in *group a* and 0.1339 in *group b* when using the default music player. Thus, it means that SAfeDJ decreases drivers' negative mood degree by 36.35% compared with the smartphone's default music player in overall. In addition, the average value is 0.2183 in *group a* and 0.2104 in *group b* when the drivers are not listening to music while driving.

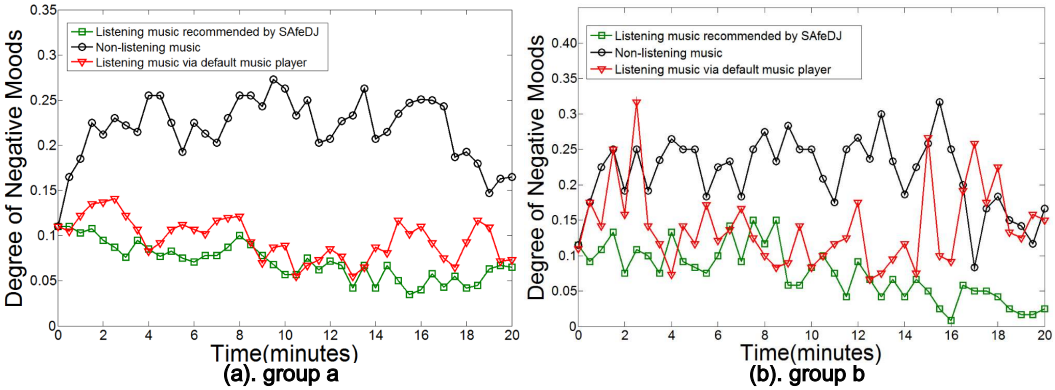

Fig. 10. Comparisons of drivers' negative mood degree

4.4 Overhead

We evaluated the basic overhead of SAfeDJ running in popular smartphones in three aspects: (i) CPU usage; (ii) RAM usage; (iii) Basic data consumption; and (iv) Battery consumption. We used a Google Nexus 5 to do the four tests in a 2005 Toyota Sienna vehicle. Also, each test lasted for 60 minutes and was repeated ten times, and then the average was calculated. The results of the four tests are summarized in Fig. 11.

The CPU usage is around 19% and RAM usage is approximate 175MB when running SAfeDJ in Nexus 5 without uploading ***MM*** for local music analysis. The CPU and RAM usages increase to around 45% (56% in peak period) and 240MB (320MB in peak period), respectively, when running SAfeDJ and uploading ***MM*** for local music analysis. Considering that the hardware capacity of smartphones is increasing and drivers can hardly operate other mobile applications while they are driving, thus the overhead of SAfeDJ should be acceptable for many popular smartphones.

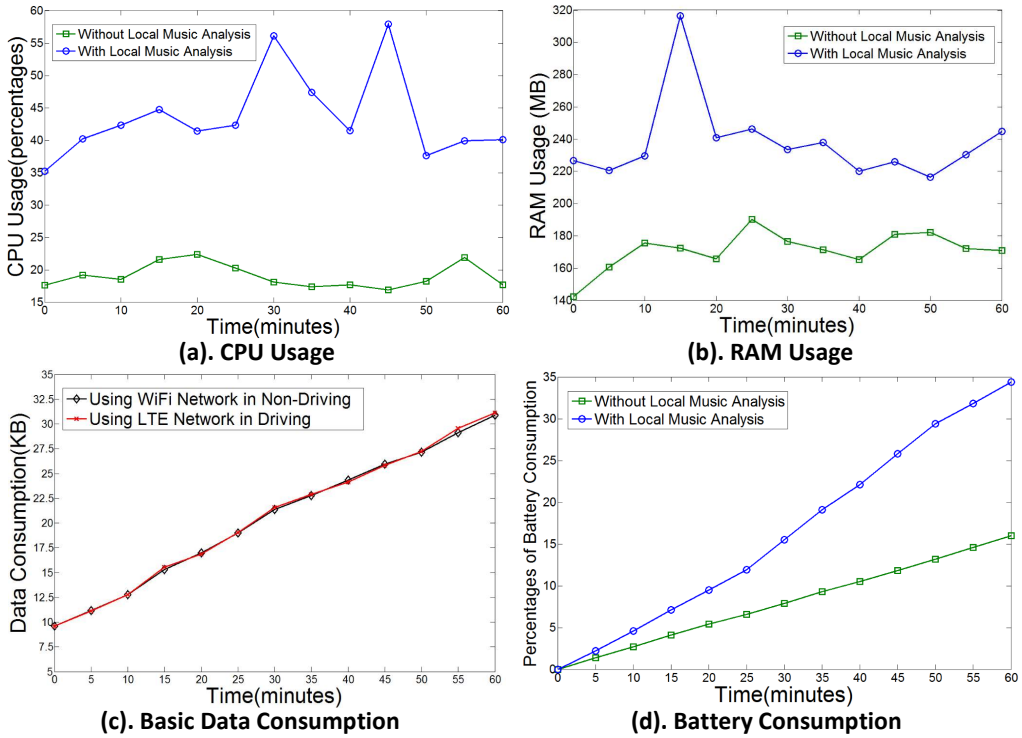

Fig. 11. Overhead of SAfeDJ in smartphones

Without uploading ***MM*** for music analysis, the data consumption of SAfeDJ is mostly spent in the transmission of music files; hence we only test the basic data consumption of SAfeDJ when finishing local music analysis in two situations. In the first situation, we recorded the data consumption over a stable WiFi network in non-driving scenarios. In the second situation, we recorded the data consumption when using LTE network and driving the 2005 Toyota Sienna in downtown Vancouver at normal speed. Fig. 11(c) shows that the basic data consumption of SAfeDJ is very low in both situations, only 9.59KB at the beginning for initial account synchronization (i.e., *SC* of a driver) between SAfeDJ cloud and Nexus 5. Then, the data consumption increases to around 31KB after running SAfeDJ in a Nexus 5 for 60 minutes, which is mainly spent in the synchronization of *HMC*, *TC* (i.e., location) and satisfaction score of music, and receiving the music recommendation results from the SAfeDJ cloud. Also, we can observe that the results in these two situations are quite close, mainly because SAfeDJ only synchronizes the context with and receives the recommendation results from the SAfeDJ cloud 5s before the end of each song as mentioned in Section 3.2. Thus, the likelihood of additional data consumption resulted by communication signaling is quite low in most common driving situations.

In addition, the battery consumption in the Nexus 5 due to running SAfeDJ is about 16% per hour without uploading ***MM*** or 34% when uploading ***MM***. Since the

smartphones could usually be charged onboard vehicles, the battery consumption should not be a serious concern for running SAfeDJ onboard vehicles. Furthermore, we observe that the ***CMM*** in SAfeDJ cloud can finish the similarity computing for music recommendation and return the results to the Nexus 5 within 3s in all our test cases, which is fast enough for music delivery in most driving scenarios.

5. CONCLUSIONS

In this paper, we have presented SAfeDJ, a novel smartphone-based situation-aware music recommendation system for drivers. SAfeDJ leverages sensing and computing capacities of smartphones to orchestrate multiple sources of sensing data with drivers' social contexts, and adopts cloud platform to form a seamless interactive system. This system enables different smartphones onboard vehicles to collaboratively recommend preferable music to drivers according to their specific driving situations intelligently in real-time. Our practical experiments have demonstrated that SAfeDJ enables the smartphones to process and identify the music-mood of each song efficiently and effectively, and detect the real-time mood-fatigue status of drivers in an effective manner. Furthermore, under the same driving situations, our user studies have verified that SAfeDJ can deliver suitable music to drivers and help them to relieve fatigue and negative emotions compared to not listening to music, or listening to music recommended by the traditional music player. To the best of our knowledge, SAfeDJ is the first smartphone-based interactive music recommendation system that recommends music to drivers not only based on their listening behaviors, but also their current states and driving situations.

REFERENCES

G. Adomavicius and A. Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the State-of-the-Art and possible extensions. *Knowledge and Data Engineering*. IEEE Trans., 17, 6 (June 2005), 734-749.

H. Ahmadi, N. Pham, R. Ganti, T. Abdelzaher, S. Nath, and J. Han. 2010. Privacy-aware regression modeling of participatory sensing data. *In Proceedings of SenSys*. ACM, New York, NY, USA, 99–112.

E. Ahmed, A. Gani, M. Sookhak, S. H. A. Hamid, and F. Xia. 2015. Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges. *Journal of Network and Computer Applications*. Elsevier, 52, 52-68.

E. Ahmed[2], A. Gani, M. K. Khan, R. Buyya, and S. U. Khan. 2015. Seamless application execution in mobile cloud computing: Motivation, taxonomy, and open challenges. *Journal of Network and Computer Applications*. Elsevier, 52, 154-172.

G. Araniti, C. Campolo, M. Condoluci, A. Iera, and A. Molinaro. 2013. LTE for Vehicular Networking: A Survey, *Communications Magazine*. IEEE, 51, 5, 148-157.

R. Ayaki, K. Kadowaki, T. Koita, and K. Sato. 2009. A proposal of distributed music delivery system with Adaptive Jini. *In Proceedings of ISADS*. IEEE, 1-5.

L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. 2012. Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing,* Springer-Verlag, London, UK. 16, 5 (June 2012), 507-526.

R. Battle and D. Kolas. 2012. Enabling the geospatial semantic web with Parliament and GeoSPARQL. *Semantic Web*. IOS Press, 3, 4 (2012), 355-370.

D. Brickley and L. Miller. 2012. FOAF vocabulary specification 0.98. Namespace Document, 9.

G.G. Cassidy and R.A.R. MacDonald. 2009. The effects of music choice on task performance: a study of the impact of self-selected and experimenter-selected music on driving game performance and experience. *Musicae Scientiae*. 13, 2 (September 21), 357–386.

Driving context-music choice survey. 2014. Retrieved from http://contextmusictest.tangkk.net

S. Fernandes, and A. Karmouch. 2012. Vertical mobility management architectures in wireless networks: A comprehensive survey and future directions. *Communications Surveys & Tutorials*. IEEE, 14, 1, 45-63.

T.Fujishima. 1999. Realtime chord recognition of musical sound: A system using common lisp music. *In Proceedings of ICMC*. 1999, 464-467.

R. K. Ganti, F. Ye, and H. Lei. 2011. Mobile crowdsensing: current state and future challenges. *Communications Magazine*. IEEE, 49, 11 (November 2011), 32-39.

E. Gómez. 2006. Tonal description of music audio signals. Doctoral dissertation, PhD thesis, UPF Barcelona.

P. Helmholz, E. Ziesmann, and S. Bissantz. 2013 Context-Awareness in the Car: Prediction, Evaluation and Usage of Route Trajectories. *Design Science at the Intersection of Physical and Virtual Design*. 7939, Springer Berlin Heidelberg. 412-419.

P. Helmholz, S. Vetter, and S. Bissantz. 2014. AmbiTune: Bringing Context-Awareness to Music Playlists while Driving. *Advancing the Impact of Design Science: Moving from Theory to Practice*. 8463, Springer International Publishing. 393-397.

X. Hu. 2010. Music and mood: where theory and reality meet. Retrieved from http://hdl.handle.net/2142/14956

X. Hu, T.H.S. Chu, H.C.B. Chan, and V.C.M. Leung. 2013. Vita: A Crowdsensing-oriented Mobile Cyber Physical System. *Emerging Topics in Computing*. IEEE Trans., 1, 1, 148-165.

X. Hu², V.C.M. Leung, K. Li, E. Kong, H. Zhang, N. Surendrakumar, and P. TalebiFard. 2013. Social Drive: A Crowdsourcing-based Vehicular Social Networking System for Green Transportation. *In Proceedings of MSWIM-DIVANet'13*. ACM, New York, NY. 85-92.

X. Hu, X. Li, E.C.-H. Ngai, V.C.M. Leung, and P. Kruchten. 2014. Multi-dimensional context-aware social network architecture for mobile crowdsensing. *Communications Magazine*. IEEE, 52, 6, 78-87.

X. Hu, T.H.S. Chu, V.C.M. Leung, E. C. -H. Ngai, P. Kruchten, and H.C.B. Chan. 2015. A Survey on Mobile Social Networks: Applications, Platforms, System Architectures, and Future Research Directions. *Communications Surveys & Tutorials*. IEEE.

X. Hu². 2015. Context-aware mobile crowdsensing in mobile social networks. Retrieved from http://mobilesoa.appspot.com/

J. Howe. 2006. The Rise of Crowdsourcing. *Wired Magazine*. 14, 6 (June 2006), 1-4. Retrieved from http://www.wired.com/wired/archive/14.06/crowds.html.

P. G. Hunter, E. G. Schellenberg, and A. T. Griffith. 2011. Misery loves company: Mood-congruent emotional responding to music. *Emotion*. 11, 5 (October 2011), 1068–1072.

Last.fm. 2014. Retrieved from http://cn.last.fm/

B. Lee, and W. Chung. 2012. A smartphone-based driver safety monitoring system using data fusion. *Sensors*, 12, 12, 17536-17552.

Music mood mapping survey. 2014. Retrieved from http://musicmoodtest.tangkk.net

S R. Ness, A. Theocharis, G. Tzanetakis, and L.G. Martins. 2009. Improving automatic music tag annotation using stacked generalization of probabilistic svm outputs. *In Proceedings of the 17th ACM international conference on Multimedia (MM'09)*. ACM, New York, NY, 705-708.

S. Pauws. 2004. Musical key extraction from audio. *In ISMIR*.

T. Pedersen, S. Patwardhan, and J. Michelizzi, J. 2004. WordNet: Similarity: measuring the relatedness of concepts. *In Proceedings of HLT-NAACL--Demonstrations '04 Demonstration Papers at HLT-NAACL 2004*. Association for Computational Linguistics, Stroudsburg, PA, 38-41.

F. Sardis, G. Mapp, J. Loo, M. Aiash, and A. Vinel. 2013. On the investigation of cloud-based mobile media environments with service-populating and QoS-aware mechanisms. *Multimedia*. IEEE Trans., 15, 4, 769-777.

Stata 13. 2014. Retrieved from http://www.stata.com/stata13/

M. Suk, and B. Prabhakaran. 2014. Real-Time Mobile Facial Expression Recognition System - A Case Study. *In Proceedings of CVPRW*. IEEE. 132-137.

J. H. Su, H. H. Yeh, P. S. Yu, and V. S. Tseng. 2010. Music recommendation using content and context information mining. *Intelligent Systems*. IEEE, 25, 1(March 2010), 16-26.

G. Tzanetakis, and P. Cook. 2002. Musical genre classification of audio signals. *Speech and Audio Processing*. IEEE Trans., 10, 5 (November 2002), 293-302.

K. Vu, R. Zheng, and J. Gao. 2012. Efficient algorithms for K-anonymous location privacy in participatory sensing. *In Proceedings of INFOCOM*. IEEE, 2399-2407.

X. Wang, D. Rosenblum, and Y. Wang. 2012. Context-aware mobile music recommendation for daily activities. *In Proceedings of the 20th ACM international conference on Multimedia (MM'12)*. ACM, New York, NY. 99-108.

World Health Organization (WHO). 2014. World report on road traffic injury prevention. Retrieved from http://www.who.int/violence_injury_prevention/publications/road_traffic/world_report/en/

M. Zwaag, C. Dijksterhuis, D. Waard, B. L. J. M. Mulder, J. H. D. M. Westerink, and K. A. Brookhuis. 2012. The influence of music on mood and performance while driving. *Ergonomics*. 55, 1 (2012), 12-22.

Online Appendix to: SAfeDJ: A Crowd-Cloud Co-design Approach to Situation-aware Music Delivery for Drivers

XIPING HU, The University of British Columbia
JUNQI DENG, The University of Hong Kong
JIDI ZHAO, East China Normal University
WENYAN HU, Nankai University
EDITH C.-H. NGAI, Uppsala University
RENFEI WANG, IBM Canada
JOHNNY SHEN, The University of British Columbia
MIN LIANG, IBM China
XITONG LI, HEC Paris
VICTOR C.M. LEUNG, The University of British Columbia
YU-KWONG KWOK, The University of Hong Kong

A. FOR SECTION 3.1.2

The diagrams about the music mood labeling presented in Section 3.1.2 are shown in Fig. A.

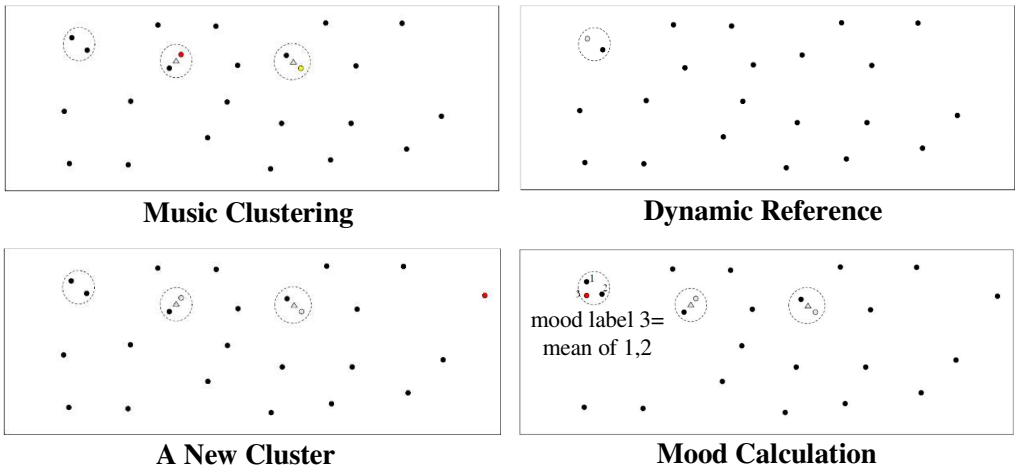

Fig. A. Diagrams of the music mood labeling

B. FOR SECTION 3.2

The working flow of the mood-fatigue detector presented in Section 3.2 is shown in Fig. B-1. Firstly, to determine the *aggressiveness* and *patience* of a driver, an OBD-II scanner is used to obtain driving data from the car, e.g., the *RPM, Tht,* and *Sp*; and these data are complimented by the data of *Hr* from a heart rate sensor. For example, driving at a high speed and braking hard may indicate aggressive driving behavior. Also, we can infer that a driver is likely aggressive in driving if the average *RPM* of vehicle is more than 4000 and the throttle input degree of the pedal is 100 percent. Similarly, the history of heart rate and driving behavior could be used to infer the driver's calmness or degree of being annoyed.

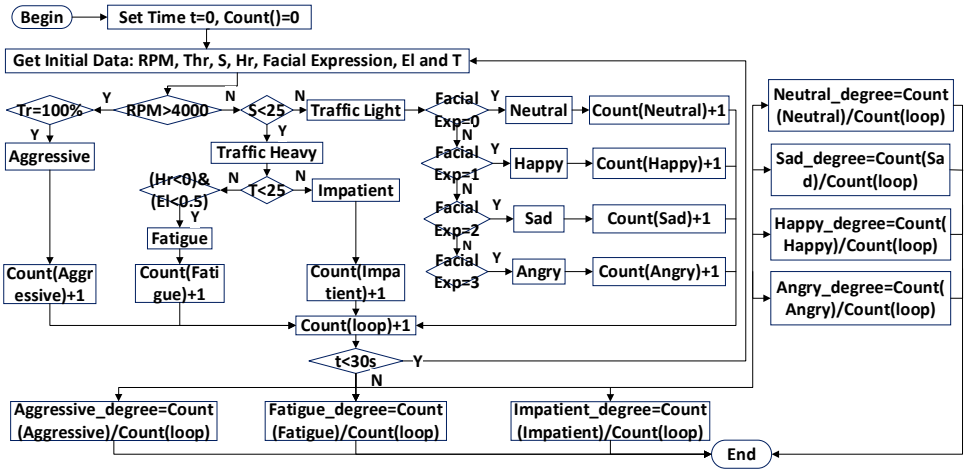

Fig. B-1. Working flow of the mood-fatigue detector

While to detect the remaining four dimensions (*neutral, happiness, sadness, anger*) of a driver's mood in a precise manner, we adopt *facial expression* to detect the driver's emotion in real-time, which is achieved by utilizing the front camera of a smartphone. Then as shown in Fig. B-2, we apply the cascade classifier to locate the face of the driver in the pictures and then use the face recognizer in the OpenCV-2.4.9 library [OpenCV 2014] to identify the mood of the driver. The recognizer is trained by the *eigenface algorithm* [Zhang et al. 2005] to increase the accuracy on mood detection.

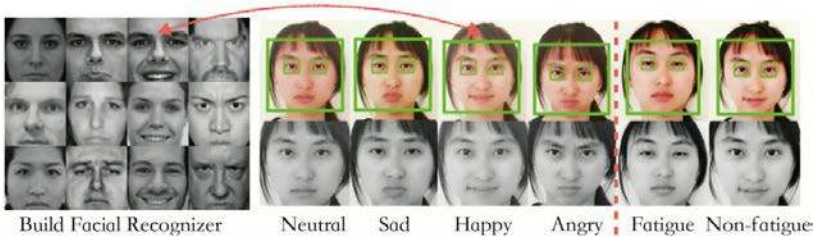

Fig. B-2. Facial expressions and eye-lid detector

On the other hand, we also invoke the front camera of smartphones and integrate the *EI* detection function to analyze the fatigue degrees of drivers. We use the *EI* distance of the driver's eyes to analyze the degree of drowsiness. First, we locate the eyes in the picture and measure the distance between the upper and lower eye-lids. We observe the driver's average eye-lid distance under neutral conditions, and use it as a reference to calculate the relative percentage on how wide the driver opens his eyes. From our initial testing, the drivers are often tired if the relative percentage is less than 50 percent. In addition, the data about *in-vehicle temperature (T)* and *road congestion situation* could also be used as complements to infer the fatigue degrees of drivers in a more precise manner.

C. FOR SECTION 3.3.1-A-1)

The method to find the most specific parent presented in Section 3.3.1-A-1) is shown in Algorithm A. The algorithm mainly contains two loops: Lines 1~3 set the quit standards for the first loop and retrieve a list stores the path information for an item. Line 4~14

include an embedded loop which traverses the path information for the second item, check the common item and return related information.

ALGORITHM A. Pseudo Code of Finding the Most Specific Parent

```
public long FindLCA(HierarchicalWordData[] words, out distance, out lcaDepth, out depth1, out depth2)
{ long LCA = -1; lcaDepth = -1; depth1 = -1; depth2 = -1; distance = MaxValue; i=-1;
1. while (++i < 1 && LCA == -1){
2.    IDictionaryEnumerator trackEnum = words[1 - i].Distance.GetEnumerator();
3.    if (trackEnum == null) return -1;
4.    while (trackEnum.MoveNext()){
5.     commonAncestor = trackEnum.Key;
6.     if (words[i].Distance.ContainsKey(commonAncestor))
7.      {dis_1 = words[i].GetDistance (commonAncestor);
8.     dis_2 = words[1 - i].GetDistance(commonAncestor);
9.      len = dis_1 + dis_2 - 1;
10.     if (distance > len)
11.      { lcaDepth_1 = words[i].GetDepth(commonAncestor);
12.      lcaDepth_2 = words[1 - i].GetDepth(commonAncestor);
```

REFERENCES

OpenCV. 2014. Retrieved from http://opencv.org

D. Zhang, S. Chen, Z. Zhou. 2005. A new face recognition method based on SVD perturbation for single example image per person. *Applied Mathematics and computation.* Elsevier, 163, 2 (April 2005), 895-907.