Spring 1-1-2016

Mobile Random Video Chat: Understanding User Behavior and Misbehavior Detection

Lei Tian
University of Colorado at Boulder, tl2006cs@gmail.com

Follow this and additional works at: https://scholar.colorado.edu/csci_gradetds
Part of the Computer Sciences Commons

Recommended Citation
https://scholar.colorado.edu/csci_gradetds/122

This Dissertation is brought to you for free and open access by Computer Science at CU Scholar. It has been accepted for inclusion in Computer Science Graduate Theses & Dissertations by an authorized administrator of CU Scholar. For more information, please contact cuscholaradmin@colorado.edu.
Mobile Random Video Chat: Understanding User Behavior and Misbehavior Detection

by

Lei Tian

B.A., University of Science and Technology of China, 2010

M.S., University of Colorado at Boulder, 2012

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science
2016
This thesis entitled:
Mobile Random Video Chat: Understanding User Behavior and Misbehavior Detection
written by Lei Tian
has been approved for the Department of Computer Science

______________________________
Professor Qin Lv

______________________________
Professor Richard Han

______________________________
Professor Shivakant Mishra

______________________________
Professor Sangtae Ha

Page dimensions: 612.0x792.0

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.
Nowadays, the near-ubiquitous availability of smartphones and the significant improvement of cellular networks make the video chat applications become mainstream for mobile devices. Meanwhile, because of the capability to make friends in the virtual domain, online random video chat services such as Chatroulette and Omegle have become increasingly popular. Given these changes, we expect the mobile random video chat services will also gain the public attention and greatly increase in volume and frequency soon. In this thesis, I focus on analyzing the user behavior and seeking for possible improvements of user experience in such kind of mobile service. I build an Android-based Omegle compliant mobile random video chat application to collect data at scale. Using the collected data, we analyze user behavior patterns from multiple aspects and reveal some concerns regarding user experience in such service. We then conduct an in-depth meaningful user behavior analysis to understand the key characteristics of effectiveness for promoting long video chat sessions. Furthermore, motivated by the negative user experience caused by the existence of obscene content, I propose an accurate and efficient misbehavior classifier. The classifier leverages multi-modal sensors and temporal modality in each session to improve accuracy. It also applies a multi-level cascaded classification procedure to quantify the tradeoff between efficiency and accuracy. Finally, I briefly introduce the potential directions which could be further investigated to improve user experience of mobile random video chat services in the future.
Dedication

I would like to dedicate this thesis to my sweet and wonderful parents, Fanjuan Kong and Gang Tian, and to my beloved grandfather, Qingyan Kong. Their words of encouragement make me able to get such success and honor. I also want to dedicate this thesis to the loving memory of my grandmothers, Baoying Lu and Wenxian Guo, for their tireless love over the years. Without their support, this work would not have been possible.
Acknowledgements

Much of the research in this thesis was conducted in collaboration with current and former faculty members and students at the University of Colorado Boulder, including Professor Qin Lv, Professor Richard Han, Professor Shivakant Mishra, Shaosong Li, Rahat Ibn Rafiq, and Dr. David Chu from Microsoft Research. I would also like to thank Leif K-Brooks, the CEO of Omegle, for helping to enable our MVChat client to be compliant with Omegle’s server, and for providing data for our research. I would also like to thank Prof. Xinyu Xing, Dr. Hanqiang Chen and Dr. Yuli Liang for providing access to their state-of-art CR algorithm and Chatroulette for sharing the performance of CR algorithm on their system. I also sincerely thank Dr. Khaled Alanezi and Dr. Mike Gartrell for their comments and suggestions. Finally, I would like to acknowledge the support of the National Science Foundation (CSR-1162614 and CIC-1048298) and the University of Colorado Boulder for my PhD study.
Contents

Chapter

1 Introduction 1

   1.1 Understanding User Behavior in Mobile Random Video Chat Services 2
   1.2 MVChat: A Publicly Deployed Mobile Random Video Chat System 3
   1.3 Meaningful User Behavior Analysis 4
   1.4 Misbehavior Detection 5

2 MVChat: A Publicly Deployed Mobile Random Video Chat System 7

   2.1 System for Data Collection 8
   2.2 Mobile Client Application 11
   2.3 Real Time Media Flow Protocol and Adobe Cirrus 12
      2.3.1 Real Time Media Flow Protocol 12
      2.3.2 Adobe Cirrus 13
      2.3.3 Adobe Air 14
   2.4 Omegle Server 14
   2.5 Data Collection Server 15
   2.6 Data Collection Experiment 17

3 Understanding User Behavior in Mobile Random Video Chat 19

   3.1 Related Work 19
   3.2 Methodology 20
### 3.3 User Behavior Analysis

3.3.1 Overall Usage Pattern Analysis ................................................. 21
3.3.2 Mobile-Online Video Chat Comparison ......................................... 33
3.3.3 Taxonomy Correlation Analysis .................................................. 35
3.3.4 Acceleration Analysis ............................................................... 39

### 3.4 Conclusions ................................................................. 42

### 4 Meaningful User Behavior Analysis ........................................ 44

4.1 Related Work ............................................................................ 44
4.2 Session-based Behavior Consistency ............................................. 46
4.3 User-based Behavior Consistency ................................................. 48
4.4 Behavior Effectiveness Analysis .................................................. 50
4.5 Conclusions .............................................................................. 52

### 5 Flasher Detection ................................................................. 53

5.1 Related Work ............................................................................ 54
5.2 New Problem for Mobile Random Video Chat .............................. 56
5.3 Flasher Detection Algorithm ....................................................... 59
5.3.1 Enhanced Image-based Classification ......................................... 60
5.3.2 Sensor-based Classification ...................................................... 61
5.3.3 Audio-based Classification ........................................................ 62
5.3.4 Session-based Classification ...................................................... 63
5.3.5 Cascaded Fusion Classification .................................................. 64
5.4 Evaluation ................................................................................ 67
5.4.1 Dataset ................................................................................. 68
5.4.2 Image-based Classifier Performance ......................................... 69
5.4.3 Audio Category Classifier Performance ..................................... 70
5.4.4 Multi-sensor Fusion Classifier Performance ............................... 71
5.4.5 Session-based Classifier Performance ........................................ 72
5.4.6 Classifier Efficiency on Mobile Devices ........................................ 73
5.4.7 Cascaded Classifier Performance ............................................... 76
5.5 Conclusions .............................................................................. 83

6 Conclusions & Future Work .......................................................... 84

Bibliography .................................................................................... 86
Tables

Table

3.1 Top 10 Words in MVChat Text Message .................................................. 29
3.2 Top 5 Topics in MVChat Text Message .................................................. 29
3.3 Taxonomy Distribution of Meaningful Session Images ............................ 32
3.4 Taxonomy Distribution of Meaningful Sessions ........................................ 32
3.5 #Occurrences of Different Acceleration Orientation Classes .................. 41
3.6 Combining Acceleration Orientation and Camera Position to Detect Misbehaving Users . . 42
4.1 Image Similarities of MVChat Snapshots .............................................. 47
4.2 Behavior Consistency Between Meaningful and Non-Meaningful Sessions ........ 49
5.1 Classification Quality Comparison of Different Image Features ................ 69
5.2 Confusion Matrix of Audio Category Prediction ...................................... 70
5.3 Classification Quality Comparison of Multi-sensor Fusion .......................... 71
5.4 Muti-sensor fusion classifier performance for different fusion techniques ........ 72
5.5 Session-based Classifier Quality Comparison ........................................ 73
5.6 Acquisition Time Comparison of Different Features ............................... 76
5.7 Structure and Performance of Optimal Cascaded Classifier for BAST and BFA when $\rho = 0.75$ 80
5.8 Session-based Cascaded Classifier Performance ....................................... 82
Figures

Figure

2.1 System architecture of the MVChat system. ........................................ 9
2.2 A screenshot of the MVChat mobile client ........................................ 11
2.3 A screenshot of the Omegle web interface ........................................ 11
2.4 The architecture of the MVChat data collection server ....................... 16
2.5 (L) User country distribution and (R) installation device OS distribution. 18

3.1 CDF of session duration: 99.5% of the 1.7 million video chat sessions were shorter than 60 seconds. ........................................................................ 22
3.2 CDF of number of sessions per user: 42% of the 4,632 users had more than 100 video chat sessions. ............................................................ 23
3.3 Time of use: (L) day of week and (R) hour of day. ............................. 24
3.4 Distribution of App Usage ................................................................. 25
3.5 Retention Rate of MVChat over 3 weeks ......................................... 26
3.6 PDF of local stop probability: Mobile users are more likely to end video chat sessions locally. 27
3.7 (L) CDF of number of text message per session and (R) CDF of number of character per message. ............................................................ 28
3.8 Distribution of Gyroscope Absolute Rotation Degree Per Second ....... 30
3.9 Sample images collected by our app showing more diversity in mobile video chat than those in online video chat. ....................................... 33
3.10 Comparison of user gender distribution: (L) mobile video chat and (R) online video chat. 34
3.11 Comparison of time of use (day of week and hour of day): (L) mobile video chat and (R) online video chat. 35
3.12 Taxonomy correlation analysis: (L) image-based graph and (R) session-based graph. Highlighted in the graphs are strong positive correlations (both \textit{all} and \textit{cosine} are > 0.85, strong negative correlations (both \textit{all} and \textit{cosine} are < 0.1), some non-correlations (\(\chi^2\) is close to 0 and lift is close to 1), and association rules with confidence > 0.8. 37
3.13 Acceleration distribution of meaningful sessions. 40
3.14 Acceleration orientation classes defined based on the degree of rotation along the x axis (long side plane of smartphones). 41
4.1 Sample images for comparison with overall image similarity 48
4.2 (L) face existence distribution and (R) person existence distribution on different types of users. 50
4.3 (L) female distribution and (R) male distribution on different types of users. 51
4.4 Front camera usage distribution on different types of users. 51
5.1 Classification accuracy of the CR algorithm on mobile video chat data using different split thresholds. 57
5.2 Examples of mobile phone-captured images that contain objects or background but no human. 58
5.3 Examples of low-quality images captured by mobile phones. 59
5.4 Number of normal vs. misbehaving users in different audio categories. 62
5.5 K-stage cascaded classifier. 65
5.6 Audio category prediction performance 70
5.7 Average CPU utilization of our fusion classifier(L) on HTC One and (R) on Galaxy Nexus. 75
5.8 Bidirectional confidence distribution generated by Weka for binary classification. 78
5.9 Optimal cascaded classifier performance by the (L) BAST criterion and by the (R) BFA criterion. 79
5.10 Optimal cascaded classifier performance on different control balance and confidence threshold. 81
Chapter 1

Introduction

Nowadays, realtime, interactive video-based applications are fast becoming an integral part of the Internet user experience. Video chat services such as Skype, Microsoft Live Messenger and Google+ are now commonplace. Meanwhile, thanks to the near-ubiquitous availability of smartphones and the significant improvement of cellular networks, jump started by iPhone 4, video chat services like FaceTime and ooVoo are rapidly becoming mainstream for mobile devices such as phones and tablets as well. A common theme shared among these applications is that these video chat services enable users to strengthen existing friendships face to face in the virtual domain.

Random chat, especially random video chat services such as Chatroulette [19], Omegle[50] and MeetMe [43] recently started to gain in popularity. In random video chat, strangers are randomly paired together for video-based conversations. The appeal of such kind of services is a classic one: the capability to meet new people face to face, now shifted to the virtual domain. Nowadays, these services have become extremely popular over the last few years. For example, both Chatroulette and Omegle have tens of thousands of online users at any given time using their systems. In the mobile area, even though there are very limited applications supporting random video chat services, random chat with text messages has been widely used across multiple platforms on mobile devices including iOS and Android. For example, Fav Talk [25, 24] and Chatous[18, 17] allow users to chat with strangers who share similar interests. Chat for Singles [16] allows people to chat anonymously without any personal information required. And similarly, RanCat [54] provides a comfortable chat environment without the societal attribute and natural attribute for people sharing their happiness and sadness. And moreover, such kind of applications also gain great popularity that
some even appeared in the top 20 popular application list of social networking category in the Apple Store.

As video chat services have been widely extended onto the mobile platform and random chat services continue to gain in popularity, we expect mobile random video chat services to increase in volume and frequency greatly as well.

1.1 Understanding User Behavior in Mobile Random Video Chat Services

While standard and even random video chat services have been studied extensively (see e.g., [31, 3, 63, 20, 64, 38]), prior work has not shed any light on the behavior of users in mobile random video chat services. Introducing the new paradigm of mobile interaction into what had formerly been primarily a desktop-based interaction paradigm with webcam-driven online random video chat raises new questions: whether the design techniques to improve the scalability and quality of these services developed based on desktop user behaviors will remain applicable when a significant percentage of users start using mobile clients. In other words, how are mobile users using random video chat? Is physical mobility fundamentally altering the nature of interacting with a virtual acquaintance?

There are mainly two important motivations for understanding the behavior of mobile users in random video chat service. First, from a social science perspective, the study of user behavior can reveal insight about why users engage in these services which can aid in designs with better user experience. For example, matching users based on their preferences could lead to higher user satisfaction and presumably extended session durations. Therefore, an understanding of mobile user behavior can inform us about the salient characteristics that would lead to better matching beyond purely random pairings. Second, safeguarding regular users from misbehaving users is important. Standard (non-mobile) random video chat services are especially prone to objectionable content such as unauthorized advertisements and sexually explicit flashers who expose private parts of their bodies. Prior work has shown that intelligent filtering that leverages user behavior patterns can help to safeguard regular users from such content [63, 20, 64, 38]. An understanding of user behavior in the mobile setting can help us investigate the existence of misbehaving problem on mobile domain and if existed, further extend prior protections to the mobile domain.

However, studying mobile user behavior in random video chat services is challenging due to two
compounding factors. First, the very nature of mobility means that users have frequent, untethered and low-overhead opportunities to interact with the video chat services. Therefore, outside observation of active mobile users is a myopic undertaking. Second, as we will show in detail, users are very selective about with whom they interact. Small scale user studies are thus ill-suited to capturing the aggregate trends of random user-to-user interactions, and the formation of new social bonds. So in order to fully understand mobile user behavior patterns in the random video chat services, it is necessary to obtain data from a large scale study proceeded in the real-world scenario.

With the insights gained from analyzing the behavior of mobile user in the random video chat services, we can then further improve user experience on mobile random video chat services in multiple aspects such as: a) characterize key effectiveness of the “meaningful” users to guide users behave effectively in mobile random video chatting; b) safeguard normal users from obscene content and “flasher”.

1.2 MVChat: A Publicly Deployed Mobile Random Video Chat System

Even though random video chat services such as Chatroulette and Omegle have gained a great popularity among teenagers and current smartphones have mastered sufficient power to support fluent mobile video chat services, mobile random video chat is still a new-emerging realm that has very limited available applications. Due to the sensitive (obscene) content, Apple bans such kind of applications for video conversation from its store and clearly states that porn is prohibited for all its random text chat apps [25, 18, 16, 54]. Meanwhile, random chat services in the Google Play Store such as Fav Talk [24] and Chatous [17] also only support random chat with text messages. Because of these, there is no way for us to obtain the real-world user behavioral data at a large scale from existing services to conduct our studies. In order to understand user behavior at scale, we implement the MVChat system, our own publicly deployed Android based and Omegle compliant mobile random video chat system, to collect data at the scale of thousands of mobile video chat users from millions of video chat sessions. The system helps us collect hundreds of gigabytes of multi-dimensional user behavioral data including image snapshot, audio, text message and smartphone sensor data.
1.3 Meaningful User Behavior Analysis

Insights gained from our overall usage pattern analysis show that “pure” random pairing is not effective for a user as it generates lots of short sessions until “right” pairing shows up in a video chat session. Most people make great efforts going through many random pairing sessions in order to find someone interested to chat with for a longer duration. This issue negatively impacts the user experience and inspires us to better understand key characteristics of user behavior which could promote more effective sessions in random video chat services.

Plenty of existing works focus on designing effective pairing algorithms for random chat services or social media [23, 2, 29, 36] to improve user satisfaction and presumably extend session durations. However, building such effective pairing mechanism is challenging with our current system due to three aspects:

- **Privacy**: Pairing algorithms require the context information from both sides of a conversation. Due to privacy consideration, our current system is only allowed to collect data from local client side.

- **Anonymity**: Random video chat services such as Chatroulette and Omegle allow users to chat without any personal information, which makes it difficult to extract user preference and interests.

- **Dynamic**: Popular services like Chatroulette and Omegle have a large and dynamic user set, and hundreds of users could switch status within minutes at peak time.

All these prevent us from extracting user preference and interests to design an effective pairing algorithm for mobile random video chat services.

To address this challenge, we learn that there are also a few amount of mobile users acting more effectively than the others (the majority) in promoting longer video chat sessions as “Meaningful Users”. In this thesis, I aim to understand the key characteristics of effectiveness among these users. We believe the findings learnt from this meaningful user behavior analysis could guide users to behave more effectively in the mobile random video chat services.

In our study, we first demonstrate that users generally behave consistently within sessions by measuring the similarities of consecutive snapshots within each session. Then by comparing the similarities
of taxonomy proportions between every user’s “meaningful” sessions and “non-meaningful” sessions, we further prove that users also act consistently across sessions regardless of being meaningful or not. Finally, based on the effectiveness of participating in the “meaningful” sessions, we divide users into three categories. And to explore whether certain taxonomy could promote longer sessions, we compare the taxonomy proportion distributions of “meaningful” sessions among different types of users and draw the conclusions that female users are much more effective in promoting long video sessions and to some extent, showing face with front camera could also contribute to have longer sessions.

1.4 Misbehavior Detection

The existence of misbehaving, or “flashers” users who expose their private body parts, has been a serious problem in online video chat services which support random video chat among strangers. Based on our large scale user study, this problem is also propagating to our mobile random video chat application. Our general study about understanding user behavior on mobile random video chat application identifies some key characteristics for flasher detection by assessing a variety of correlative factors that occur between mobile sensor data and the two types of user behavior, namely normal and flashing user behavior. With these observations, we seek to further address this problem of detecting flashers in a systematic manner with high accuracy and efficiency.

In our work, we follow the common definition shared in the previous researches [31, 3, 20, 64, 38] and define misbehaviors (or flashers) as users who expose their private bodies and show obscene content to others such as males/females showing their lower body part or females exposing their chests. Prior researches have described some successful image-only classifications for online webcam-video chat users [31, 3, 20, 64, 38] based on the correlations between presence of facial feature indicators and user behavior that online users who show certain facial features are found to be highly correlated with normal behavior, whereas users who do not display them are found to be associated with flashing behavior. However, our first investigation about performance of these classifiers onto our mobile client indicates these relationships are broken down by multiple factors and these face-centric classifications achieve a poor accuracy when operated over mobile video chat data among the various techniques explored in our study.
Because today’s smartphones provide a wealth of contextual data from their mobile sensors such as three-axis acceleration and gyroscope, light intensity, audio, front/back camera state and image snapshot, our main focus is to investigate whether additional sensing modalities can be leveraged to improve mobile flasher classification performance. We characterize which of these mobile sensors could help to meaningfully improve the accuracy of classification and show that by fusing together classifiers based on image as well as key mobile sensing modalities like acceleration, camera state and audio, the resulting fused classifier is substantially more accurate than those basic image-only face-centric classifiers. Moreover, we demonstrate that by integrating the temporal modality within a session, our session-based classifier’s accuracy can be further improved.

Besides accuracy, another key focus of our research is to improve the efficiency of classification, namely reducing the running time and resources usage of the fused classifier. Fused classification can result in executing classifiers that are especially compute-intensive, such as image-based and audio-based classifiers. The efficiency optimization is particularly helpful for mobile devices that are comparatively resource-constrained. Our final multi-level cascaded classifier could greatly improve the efficiency and meanwhile largely preserve the accuracy by carefully limiting execution of high-complexity individual classifiers and further imaged-based predictions within a session.

In the remainder of this thesis, I will first describe the architecture and each of its components in detail of our MVChat system in Chapter 2. Then in Chapter 3, I will discuss our study of understanding general mobile user behavior in our random video chat service and highlight the major characteristics learnt from it. Next, an in-depth meaningful user behavior analysis is conducted to investigate the key characteristics of user behavior to promote longer sessions in random video chat services. And then I will introduce our proposed misbehavior classifier for mobile random video chat service and demonstrate the overall classification accuracy improves substantially by incorporating mobile sensing modality. Finally, I summarize our thesis work and briefly discuss the potential directions which could be further investigated in the mobile random video chat area.
Chapter 2

MVChat: A Publicly Deployed Mobile Random Video Chat System

Thanks to the significant improvement of mobile hardware capability and cellular network connectivity in the past few years, since FaceTime was supported on iPhone 4, more and more video chat applications such as ooVoo and Qik have emerged on mobile platform and quickly gained in popularity. Prior work by Jana et al. [34] did a comprehensive study and discussed the issues and challenges in designing successful mobile video chat applications in detail. And Cahill et al. [14] describes a generally computer-implemented architecture of providing live video chats in a network.

Meanwhile, random video chat services have been widely supported among desktops and PCs by means of web service. Popular services like Chatroulette, Omegle and MeetMe have tens of thousands of online users at any given time during a day. Furthermore, applications such as Fan Talk and Chatous which provide pure random text chat services have also appeared on different mobile platforms and become increasingly popular recently.

However, even though from these changes we can expect mobile random video chat services will also increase significantly in volume and frequency in the future, currently there are still very limited random video chat applications available on the mobile platforms. The largest mobile platform, Apple bans such kind of service from its store for video conversation because of the presence of obscene content. Meanwhile, in another great mobile platform, the Google Play Store, popular random chat applications only support random chat services with text messages.

To conduct a first-ever detailed study of understanding mobile user behavior in a random video chat service at scale, we developed an Android-based mobile client named MVChat for random video chat ser-
vices through which we can collect user data. This client allows mobile users to connect with online Omegle users for random video chat. In addition, MVChat logs on our server multi-dimensional, user-related sensor data from each participating mobile client. The data includes image snapshot, audio, camera position (front or back), accelerometer and gyroscope data, text message, and disconnection behavior. And our MVChat mobile application is publicly deployed onto Google Play Store.

2.1 System for Data Collection

In order to understand user behavior at scale, we design our MVChat system to collect data on the scale of thousands of mobile video chat users, millions of video chat sessions, and hundreds of gigabytes of mobile sensing data including image snapshot, audio, acceleration and etc. Building a mobile application that is compatible with an existing popular online video chat service allows us to quickly scale our study to a large number of users. We choose to make MVChat compatible with Omegle, which is a popular random video chat service that has tens of thousands of randomly paired users online at any given time. Another great advantage of the Omegle service is, unlike Chatroulette, which has frequent changes in its web client code, Omegle provides a relatively stable code base for its client, so that we can easily build a mobile client that is compatible with the Omegle web service via a stable application protocol. Besides, different from Chatroulette, Omegle’s user population is unfiltered, which allows MVChat to gain insights about misbehaving users and thereby capture a more representative sample of the true proportions and behaviors of mobile random video chat users. Meanwhile, we choose the Android platform because the application approval process for iPhone essentially precludes any applications such as random video chat which has some indecent content or misbehaving users. Accordingly, the MVChat system has been designed to let mobile users of Android-based smartphones connect to the Omegle server and (video) chat with other online Omegle users. In addition, this system is designed to collect mobile user data including image snapshot, local text message, camera position (front or back) as well as sensor data such as accelerometer, audio and gyroscope. Our MVChat system also logs disconnection behavior which could tell us which side shuts down the session.

Figure 2.1 illustrates the overall architecture of our MVChat system. The system consists of four
major components:

- **Android mobile client**: provides mobile a random video chat service compliant to Omegle service.

- **Adobe Cirrus (Stratus) server**: helps to build peer-to-peer communications between clients.

- **Omegle server**: handles random pairing mechanism and assists to transmit text messages between the paired devices.

- **Data collection server**: stores multi-dimensional user behavioral data collected during the (video) chat.

---

**Figure 2.1**: System architecture of the MVChat system.
Meanwhile, compatibility of our MVChat clients with the Omegle system requires careful engineering to mimic the behavior of online web clients. As it is shown in Figure 2.1, our MVChat client has mainly four types of communications for establishing a peer-to-peer video conversation:

1) **Stratus Registration:** A typical MVChat client first contacts the Adobe Cirrus server to acquire a unique peer ID. This peer ID is later used to assist building peer-to-peer communications.

2) **App Behavior Control:** Once a client gets the peer ID from the Adobe Cirrus server, the next thing it does is to establish a TCP connection with the Omegle server and provide its peer ID for registration. The TCP connection between the Omegle server and the MVChat client is used to guide the client behavior such as establishing/disconnecting random video chat sessions and transmitting text messages.

3) **Video Session Establishment:** Once the two sides of a random pair know each other’s peer ID from the Omegle server, they could establish a peer-to-peer video session by the Real Time Media Flow Protocol (RTMFP).

4) **Data Collection:** Once a video session is established, MVChat mobile client will periodically posts user behavior data to our data collection server until the session is closed.
2.2 Mobile Client Application

Figure 2.2: A screenshot of the MVChat mobile client

Figure 2.3: A screenshot of the Omegle web interface

Figure 2.2 shows a screenshot of our MVChat mobile client which is patterned to mimic the Omegle user interface shown in Figure 2.3, because since we get the permission to claim as a compliant client for Omegle, we assume our first group of users should be familiar with Omegle web application. Following their design pattern will make people easy to use our application. However, due to restrictions such as the
limited screen space and the specificity of mobile platform software, we have to make some changes to adapt them for emerging on the mobile platform.

First, unlike the Omegle web client, where text messaging dominates the screen and the videos of the sender and receiver are shown in less than half the screen, our MVChat mobile client emphasizes video, as there is limited screen real estate and we feel most mobile users would interact easily with video and audio. After invocation and connection with a remote user, the mobile client displays the remote client’s video (captured via remote user’s device camera) in the larger window on the right since people care more about others’ actions than their own. The local camera view that is being sent to the remote user is shown in the lower left window. However, to maintain compatibility with the user experience of Omegle chatters, we still include text messaging. Text messages are composed in the upper left dialog box and the most recent messages are shown in the middle left box.

Also unlike web clients, mobile devices have both front and back cameras. So we add a pull-down menu that allows mobile users to switch the cameras for the video chat. The other pull-down menu allows mobile users to specify the microphone that they wish to use, should there be more than one supported.

Besides, to maintain compatibility with Omegle’s session flow, which one click ends a session and a second click requests a new session, we add a Disconnect button that is displayed at the top when a video chat session is in progress. A user could press this button to end the current video chat session, which changes the button to a Next button for him to request another new session.

2.3  **Real Time Media Flow Protocol and Adobe Cirrus**

To easily build a peer-to-peer communication and guarantee smooth video stream transmission, we choose to use Real Time Media Flow Protocol(RTMFP) on the Adobe Air platform for our MVChat system.

2.3.1 **Real Time Media Flow Protocol**

The Real-Time Media Flow Protocol (RTMFP) is a protocol developed by Adobe Systems. The protocol provides encrypted, scalable and efficient multimedia delivery between end users for peer com-
munication on both client-server and peer-to-peer models over the Internet. By using RTMFP, applications that rely on live, real-time communications, such as social networking services and multiuser games are able to deliver higher quality communication solutions. And the RTMFP enables end users to connect and communicate directly with each other using their computers’ microphones and webcams. RTMFP has the following benefits:

- **Bandwidth cost reduction:** RTMFP can help reduce the bandwidth costs for direct, live, real-time communication solutions, such as audio and video chat and multiplayer games. Because RTMFP only need to send data directly between the end-user clients and not through the server, solutions are less expensive to scale.

- **Low latency delivery:** RTMFP increases the quality of delivery through the use of the User Datagram Protocol (UDP). UDP is a more efficient way to send video and audio data over the Internet that helps ensure connections are not interrupted if variations occur within the network.

These benefits together with the end-to-end peer capability and scalability make RTMFP especially well suited for developing real-time collaboration applications by not only providing superior user experience but also reducing cost for operators.

**2.3.2 Adobe Cirrus**

Adobe Cirrus (previously codename Stratus) is a hosted rendezvous service that enables peer assisted network communication using the Real Time Media Flow Protocol (RTMFP) on the Adobe Flash Platform. Adobe Cirrus service consists of a RTMFP-capable server. In order to use RTMFP, each endpoint need to first connect to it and request for a peer ID. This ID helps Cirrus to identify the instance of a web client using Adobe Flash or a mobile device using Adobe AIR, and assist it to build a peer connection with others. And because of the demand for Adobe Cirrus and RTMFP, we choose to use Adobe Air programming on the Android devices.
2.3.3 Adobe Air

Adobe Integrated Runtime, also known as Adobe AIR, is a cross-platform runtime system developed by Adobe Systems for building desktop or mobile applications programmed by Adobe Flash, ActionScript and optionally Apache Flex. The runtime system supports installable applications on Windows, OS X and mobile operating systems like Android, iOS and BlackBerry OS.

The Adobe AIR is a runtime environment that allows Adobe Flash, ActionScript, HTML5 and JavaScript code to construct applications that run outside a web browser and behave like a native application on supported platforms. Internally, Adobe AIR uses Adobe Flash as the rendering engine, and ActionScript 3 as the primary programming language. And applications developed by Adobe Air allow users to install the application from an installer file (Windows and OS X) or through the appropriate App Store (iOS and Android). Besides, AIR applications have unrestricted access to local storage and file systems, while browser-based applications only have access to the files selected by users.

For mobile platforms, AIR supports many mobile hardware features, including hardware-accelerated graphics rendering, touch-screen gestures, camera and microphone, accelerometer and networking with HTTP, TCP and UDP protocols. At the same time, Adobe AIR applications are easily published as native phone applications onto certain mobile operating systems such as Android and iOS.

2.4 Omegle Server

Since we want our application to quickly scale our study to a large number of users, we choose to hook our application to an existing popular web-based random video chat service, the Omegle Service. Omegle is known as one of several heated websites providing random video chat service for tens of thousands of users at any given time during a day. Meanwhile, it also suffers from the problem of obscene content. Making our application compliant with Omegle will help us to conduct a scalable study and at the same time, capture a more representative sample of the true proportions and behaviors of mobile random video chat users.

To connect with Omegle service, every client needs to communicate with the Omegle server by maintaining a TCP connection with it. The Omegle server uses this connection to control client behavior such
as handling random pairing mechanism and transmitting text messages between the paired devices. Once a client gets its peer ID from the Adobe Cirrus server, it immediately builds a TCP connection with the Omegle server and sends this ID to the Omegle server for registration. Then, anytime the client wants a video session, it sends a request to the Omegle server through the TCP connection for random pairing. The Omegle server will randomly select another peer ID from all the idle devices and send it back as a response. Meanwhile, it will also notify the selected device with the ID belonging to the requesting device via its TCP connection. Then the paired devices know each other and can build a peer-to-peer video chat session by RTMFP. Once any side of a video session wants to terminate the session, it needs to first send a termination message to the Omegle server, then stop transmitting video stream and release resources. Next, the Omegle server will notify the other side of the session to disconnect the video conversation. Additionally, text messages are retransmitted by the Omegle server between paired devices and the Omegle server also implements the reCAPTCHA [27] to protect its service from spam and abuse by the TCP connections. Last but not least, once built, the TCP connection of a client will be held until our application is shut down on the device.

2.5 Data Collection Server

In order to fully understand user behavior in a mobile random video chat, our experiment tries to maintain a good coverage of mobile sensor collection from all aspects. Our data collection server collects multi-dimensional sensor information including periodical snapshots and audio clips, continuous acceleration, gyroscope and camera state information, every text message sent from local mobile client and the disconnecting information from each session.

As shown in Figure 2.4, our data collection server is comprised of three components:

- **Apache server**: runs Python CGI scripts to store image snapshots directly on the server’s file system and at the same time cooperates with the MySQL database to store the mobile sensor data.

- **MySQL database**: creates separate tables for different types of sensor data. Current database manages four tables logging acceleration, gyroscope, text message and the session disconnecting
information respectively.

- **Flash Interactive Media Server**: is responsible for storing the audio clips because it is easy to interact with our mobile client via the RTMFP protocol.

![Figure 2.4: The architecture of the MVChat data collection server](image)

For each mobile device, a random and unique device ID (not the Adobe Cirrus peer ID) is generated, and for each new video session, the device also creates a unique session ID. Every sample of data posted from our MVChat client to our data collection server is associated with timestamp, current used camera position as well as its session ID and device ID for easy segmentation in our future analysis.

The default sampling frequency for accelerometer and gyroscope is 5Hz. Snapshots are captured every 30 seconds and audio is captured in the first 10 seconds of every 40-second interval. On average, each image is about 35 KB (120*160), each audio file (10 second duration) is around 110 KB and the amount of the rest sensor data (acceleration and gyroscope) is less than 1KB each second. So the total amount of data transmitted from mobile client to the data collection server is less than 5 KB per second per user. This
aggregate sampling rate in theory allows us to scale to thousands of concurrent users, though in practice we
have thus far seen a maximum of 80 concurrent users.

Moreover, in order to accommodate unforeseen workloads and help the system scale, we design our
data collection server to adjust flexibly or even turn off the sensor reporting streams from the mobile clients.
For each mobile client, the server can vary the sampling frequencies, or even stop data collection altogether.
This ability allows us to throttle clients if the overall load on the data collection server is too high, and also
to suspend data collection while allowing the clients to continue to video chat normally. To achieve this
mechanism, when a client starts, it first contacts the data collection server, which responds with one of three
options:

- Disable the client completely so that it will not access the Omegle service at all
- Disable just the posting of sensor data to the collection server, whereupon the client can still access
  the Omegle service without any remote logging
- Enable the posting of sensor data as the user accesses the Omegle service. In this case, the server
  also (optionally) specifies the sampling frequencies for each sensor modality

With the help of this adaptively sampling control mechanism, we could even apply different sampling modes
to different subsets of users based on different requirements to balance load of our system for scalability.
Besides, even if our data collection server is not up or if we purposely take down the server when the client
tries to contact it, mobile client will continue to function as normal, because this is just an HTTP POST and
the exception can be ignored.

2.6 Data Collection Experiment

We publicly deployed our MVChat mobile application onto the Google Play store. The latest version
was released in late Jan of 2013 and got more than thirty thousands of downloads until May of 2013. In this
thesis, I analyze in detail about three weeks of user behavior data spanning from January 25th to February
14th, 2013. In total, 4,632 distinct users of our MVChat application were identified in this period of time,
generating 1,703,837 pairing sessions. To protect remote users’ privacy, data was only collected locally from our mobile users. The total amount of data used for analysis was about 170 GB, comprised of 70 GB of image snapshots, 90 GB of audio snippets, 8.5 GB of mobile sensor, texting, and clicking data.

Institutional Review Board (IRB) approval was obtained for this deployment and users of MVChat need to give their informed consent that they fully understood that a variety of their data would be collected for our research project. We achieved this by hardcoding our consent form into our mobile application. For the first time using our app, users must first promise they are over 18 and agree with our consent form shown in the application before further usage.

![Pie chart showing user country distribution.](image1)

![Pie chart showing installation device OS distribution.](image2)

Figure 2.5: (L) User country distribution and (R) installation device OS distribution.

Figure 2.5(L) shows the distribution of users by country of origin who have downloaded and installed our application. We see that the vast majority are from the United States and that all other countries comprise at most single digit percentages of the downloads. At the same time, Figure 2.5(R) demonstrates the software and hardware capabilities of our users that many users of our application still rely on older versions of Android 2.3, but that a fairly large fraction have Android 4.0 or higher. We also found that about three-fourths of our users’ smartphones were equipped with both front and back cameras, but that a fourth still lacked the front camera.
Chapter 3

Understanding User Behavior in Mobile Random Video Chat

3.1 Related Work

For previous works related to understanding user behavior in video chat applications, Scholl et al. [57] designed and built an online video chat application, reporting results from 53 users in a social setting, focusing on bandwidth issues and view navigation. VideoPal explored the use of video to facilitate asynchronous communication between six children and their close friends [31]. Other works focused on understanding video chat system usage between teenagers [13, 59] or in the context of families, especially the facilitation of communication between grandparents and grandchildren [3, 53]. And similar study [44] explored the motivations of Canadian citizens over the ages of 55 for using video communication, unveiling challenges and benefits they experience. Our work differs from these prior works in one or more of the following respects: its larger scale; its publicly available deployment; its focus on the mobile context; and the study of random video chat service.

In mobile video research, Schoeffmann et al. [56] integrated literature from multiple video interaction aspects, classified them by the underlying interaction method and depicted a set of new problems in the video interaction community. MoVi explored the use of collaborative sensing on mobile phones to trigger the video recording of a social event by one of the participants’ camera phone, as well as the generation of video highlights of the event [7]. MicroCast sought to share video streaming amongst a local group of smartphones, who also shared their partial results with one another [35]. Besides, mobile video encoding for wireless links has recently been introduced by cross-layer encoding (SoftCast) [33] and reliable coding (ChitChat) [61] techniques. Finally, a study measuring the energy and bandwidth costs of streaming video
from popular websites such as YouTube to six different smartphones has been conducted in 2012 [30].

Besides, there have been a number of studies examining user behavior of mobile applications and usage of mobile devices. In prior art of works about mobile device and application usage, MyExperience [26] captured both objective in situ data like device usage, sensor reading and subjective user feedback to keep track of more than 140 types of events. Church and Smyth [21] investigated mobile information needs, understood the contexts on the needs and highlighted the importance of temporal and location dependencies. And Verkasalo [60] studied the mobile service usages in different contexts. And there are several other researches exploring the web usage on the mobile device [22, 32]. Böhmer et al. [9] presented a large-scale study analyzing Android application usage in basic descriptive and contextual descriptive statistics. And other works like [41, 28] focused on investigating usage patterns of social networks such as Facebook and Sina Weibo on mobile.

3.2 Methodology

In seeking to understand user behavior of mobile random video chat at scale, we are interested in answering a variety of questions, starting with overall usage pattern questions. What is the length of a typical random video chat session? How frequent do users use our application and how often do they seek for a new session while using the application? What (day/night/hour) are the most popular times for random video chat? How often do mobile users terminate their sessions compared to the other side terminating the session? Compared to audio, what role does text message play during a typical mobile video chat session and what are the hot topics discussed via text message? What fraction of mobile users behave in an unsafe manner, e.g., flash or reveal themselves? And what are the key differences in user behaviors between mobile users and other online video chat users?

Besides, behavioral context questions are also of great interest. We conduct both image-based and session-based taxonomy analysis on our data. Once we have the basic taxonomic classifications for images and their corresponding sessions, we explore whether there are certain strong correlations (positive or negative) between various characteristics, such as presence/absence of a person, presence/absence of a face, gender, front/back camera usage, normal behavior/misbehavior and audio silence/voice/music? And
how these correlations change between different granularities (image and session)? And finally, is there any correlation between sensor data and user behavior? Or in other words, from mobile sensor contextual standpoint, to what extent can mobile sensor data help us understand the overall environment, e.g., mobile orientation and motion, background sound, etc for solving specific mobile video chat problems?

3.3 User Behavior Analysis

We conducted a comprehensive analysis of our data to understand and identify key behavioral characteristics of mobile video chat users at scale. Our data analysis consists of mainly four specific components: (1) Overall usage pattern analysis studies the overall statistical distributions and identifies common user behaviors. (2) Mobile-online video chat comparison highlights the differences between mobile and online video chat, and new features associated with mobile video chat. (3) Taxonomy correlation analysis identifies user behavioral attributes that have strong positive or negative correlations and directional association rules with strong confidence. (4) Accelerometer data analysis studies how accelerometer sensor data can help identify certain user behavioral characteristics in mobile video chat.

3.3.1 Overall Usage Pattern Analysis

We first conduct an overall usage pattern analysis to understand general user behavior patterns in mobile random video chat, including session duration, time of use, application usage, local stop behavior, usage of text message, GPS data, gyroscope data and a taxonomy distribution.

Session duration refers to the length of users’ video chat sessions. Figure 3.1 shows the cumulative distribution function (CDF) for session duration. Among the 1.7 million video chat sessions we have collected, 80% of the sessions are less than 5 seconds; only 1% of the sessions are longer than 30 seconds; and only 0.5% of the sessions are longer than 60 seconds. Figure 3.2 shows the CDF of the number of sessions for each user. Among the 4,632 users we have observed, 80% of the users participate in at least 10 sessions, 42% of the users participate in more than 100 video chat sessions; and 6% of the users participate in over 1,000 video chat sessions during our 3-week data collection period. A hypothesis consistent with these findings is that video chat users spend a lot of effort going through many random pairing sessions in
order to find someone interested to chat with for a longer duration. These numbers are significant and somewhat surprising given the high popularity of video chat services. Since strangers in video chat services are randomly paired and each user can click “Next” to end the current session and start the next pairing, users are constantly “seeking” the right person to talk to. Or, to be precise, users are “waiting” for the right person to show up in their video chat screen. Such passive pairing, while introducing some element of surprise, is ineffective in connecting strangers who are interested in meeting each other. While some video chat services have recently introduced interest-based pairing, it is hard to extract user interests and build an effective matching metric for our current system. To improve this problem, we conduct an in-depth meaningful user behavior analysis later in this thesis.

Figure 3.1: CDF of session duration: 99.5% of the 1.7 million video chat sessions were shorter than 60 seconds.
Figure 3.2: CDF of number of sessions per user: 42% of the 4,632 users had more than 100 video chat sessions.

**Time of use** refers to the time when users participate in video chat sessions. We consider both day of week and hour of day using the local time reported by users’ smartphones. As shown in Figure 3.3(L), the average number of users is similar throughout a week except for Fridays and Saturdays. This is interesting as we expect more users because people have free time on Fridays and Saturdays. One possible explanation is that, because most users are free on Friday and Saturday nights, they go out and participate in other social events (i.e., “party time”) instead of using our mobile video chat application.

Meanwhile, Figure 3.3(R) shows the number of users during different hours of the day. We can see that early morning (5am) has the fewest number of users, and the average number of users increases steadily throughout the day, with quick jumps at 3pm and 10pm, and reaches the highest number around midnight. The fact that we observe more users during the late evening hours is potentially related to the private nature of mobile video chat, i.e., when users are by themselves and would like to meet strangers in the virtual
App Usage measures the frequency users use our application. In our study, if two consecutive sessions from a user have a time gap more than 12 hours, we count them as two separate app usages. Based on this definition, Figure 3.4 shows the CDF distribution of our application usage. It indicates that nearly 66% of users use our application only once, meanwhile more than 5% of users use it more than 5 times. Besides, after going through the image snapshots from each user identified by the unique device ID, we observe that some users have multiple device IDs in our dataset. It is important to note that in our application, device ID is created when the first time users start our client and it is only stored locally in the device. Uninstalling application will also erase the device ID from the smartphone. In other words, our client generates a new device ID for each time of installation. So this observation tells us that mobile users might uninstall our application when they finish video chat for privacy concern and re-install it later when they need it.
Moreover, we measure the daily retention rate of our application. The retention rate tells the ratio of retained user to an application over a period of time. It monitors application performance in retaining users and measures the stability of application usage. The daily retention rate is defined as follow:

\[
\text{Retention Rate}_i = \frac{N_{i+1} \land N_i}{N_i} \\
= \frac{N_{i+1} - X_{i+1}}{N_i}
\]  

Where \( N_i \) is number of users in the \( i \)th day and \( X_{i+1} \) is compared to the \( i \)th day, the number of new users gained in the \( i + 1 \)th day. Figure 3.5 shows the retention rate of our application over our 3-week user study. It indicates our application has a relative stable retention rate, that the average over this period of time is about 34.3%. This result is consistent with the percentage of users using our client for more than one time.
Local stop probability refers to the probability of a user ending his/her session locally instead of the remote party ending the session. Each user can click the “Disconnect” and “Next” buttons at any time to end the current video chat session and start a new pairing session. It is important to note that, since our MVChat client connects mobile users to chat with Omegle online users, and given the fact that the number of our mobile users is only about one percentage of the number of Omegle users, the remote side of a mobile video chat session is mostly an Omegle user. So if we assume that for each video chat session, the two parties (local and remote) have equal probability of ending the session, then across all users and their sessions, we would expect to see a distribution for local stop probability with a mean value of 0.5. However, the distribution shown in Figure 3.6 is different, with a mean value close to 0.6. The higher local stop probability of our application means that mobile users are more likely to end a video chat session than the Omegle online users. We think such deviation is possibly due to the difference in network connections between mobile users and online users. Video chat generally requires good network connections. Mobile users are typically connected through 3G or WiFi networks which could cause very poor video quality, while the online users
usually have better network connection with higher bandwidth and more stability. Also, compared to the PCs, mobile devices are resource-restricted which could also cause video chat not running as smoothly as it does on PCs. So because of the poor video quality caused by the unstable network connections or limited mobile resources, users might have less tolerance when using our client or even think video sessions fail to be built as to restart a new session. As a result, mobile users are more likely to end a session if there is a delay in starting the session or the video quality is low. In fact, when we tested our application, we did find sometimes it takes 2-3 seconds to build a chat session and display video stream which rarely happens on the web application. Finally, it is easier to interact through mobile touchscreen, which could also make our mobile users more selective.

Figure 3.6: PDF of local stop probability: Mobile users are more likely to end video chat sessions locally.

**Text messages** can also be used in video chat sessions. Our preliminary study reveals that although 58.3% of the users have used text messages during video chat, only 2.8% of all video chat sessions contain text messages. We think this is due to the fact that most of the sessions are short sessions, when users are quickly clicking through many random pairing sessions in order to find the right person to talk to, they rarely type any message. So short sessions generally contain no text. Moreover, we learn that among those
sessions which have text messages, the number of text message per session is mostly small. As shown in Figure 3.7(L), the median of the number of text message per session is only 1 and the mean of it is 2.03.

Beside, we also observe that people are reluctant to chat via text message in mobile video chat applications because the keyboard sizes on smartphones are very limited and hard to interact compared to PCs. This is indicated in Figure 3.7(R) that the median value of the number of character per message is 5 and the average number of character in each message is 9.83, including all whitespace. Mobile users only text simple messages between each other during the chatting. As shown in Table 3.1, the most frequently used words in the mobile text message are either abbreviations or greeting words, which are sent at the beginning of video sessions.

Finally, we apply topic modeling to our text dataset. Topic modeling is a type of statistical model for discovering the abstract “topics” that occur in a collection of documents, based upon the idea that documents are mixtures of topics. And a topic is a probability distribution over words. Nowadays, topic modeling has been widely applied to multiple fields including natural language processing, bioinformatics and even social network [8, 58].

In our study, we use the topic modeling library available in Mallet toolkit [42], and all the messages from one session are grouped together into a single document. The stop words are manually selected by us, combining the default stop words from Mallet library with the words listed in Table 3.1, to remove those
Table 3.1: Top 10 Words in MVChat Text Message

<table>
<thead>
<tr>
<th>Ranking</th>
<th>Word</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>hi</td>
<td>0.187</td>
</tr>
<tr>
<td>2</td>
<td>u</td>
<td>0.084</td>
</tr>
<tr>
<td>3</td>
<td>me</td>
<td>0.070</td>
</tr>
<tr>
<td>4</td>
<td>i</td>
<td>0.064</td>
</tr>
<tr>
<td>5</td>
<td>see</td>
<td>0.051</td>
</tr>
<tr>
<td>6</td>
<td>hey</td>
<td>0.050</td>
</tr>
<tr>
<td>7</td>
<td>wanna</td>
<td>0.050</td>
</tr>
<tr>
<td>8</td>
<td>you</td>
<td>0.045</td>
</tr>
<tr>
<td>9</td>
<td>show</td>
<td>0.042</td>
</tr>
<tr>
<td>10</td>
<td>cum</td>
<td>0.041</td>
</tr>
</tbody>
</table>

frequently used abbreviations in online and mobile chatting. Table 3.2 shows the top 5 topics extracted from our text dataset. Interestingly, we find that most of the topics contain some offensive and erotic words, which indicates that the presence of obscene content problem has also propagated to the mobile random video chat services.

Table 3.2: Top 5 Topics in MVChat Text Message

<table>
<thead>
<tr>
<th>Ranking</th>
<th>Weight</th>
<th>5 Representative Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.09519</td>
<td>show, dick, tits, boobs, cock</td>
</tr>
<tr>
<td>2</td>
<td>0.05916</td>
<td>lol, good, cool, nice, cute</td>
</tr>
<tr>
<td>3</td>
<td>0.05535</td>
<td>asl, cam, female, sex, gay</td>
</tr>
<tr>
<td>4</td>
<td>0.03184</td>
<td>cum, watch, fast, hard, damn</td>
</tr>
<tr>
<td>5</td>
<td>0.01799</td>
<td>see, cum, girl, horny, penis</td>
</tr>
</tbody>
</table>

GPS data collection is a function incorporated in our application. However, the data we collected contain almost no GPS data. This, together with our examination of the snapshot images, indicates that almost all video chat sessions occurred indoors. This is reasonable as video chatting with strangers is considered as a private activity and people prefer to participate in such activity in private indoor environments. Moreover, based on our observation of the snapshot images, many of our mobile video chat users are young people and tend to use the application in their homes or dorms. The indoor locations vary from living room to bedroom and even bathroom. Also, mobile user posture is distributed across sitting, lying down, and standing, and appears to be more diverse than online Webcam-based images captured from desktop clients [63], where users typically are located in the bedroom and are sitting.

Gyroscope data measures the rate of rotation in rad/s around a device’s x, y, and z axises. It is a
good indicator for monitoring the vibration of phone motion. In our work, we calculate the absolute rotation degree of a phone within each second to briefly measure the stability of the phone during video chat. The detailed of our calculation are shown as follows:

(1) **Resampling:** For each session, we resample the gyroscope data in 8Hz by linear interpolation.

(2) **Filtering:** A high-pass filter is applied to compensate for the drifting problem of gyroscope data.

(3) **Framing:** Sessions are segmented into multiple frames without overlapping and each frame lasts for one second.

(4) **Feature Extraction:** Absolute rotation degrees around x, y and z axes are calculated respectively by $|\theta| = \sum_{i=1}^{8} |\omega_i| (t_{i+1} - t_i)$, where $\omega_i$ is the $i$th resampled value of gyroscope data around x, y and z. Then the total rotation degree is calculated by $\sqrt{\theta_x^2 + \theta_y^2 + \theta_z^2}$.

![Figure 3.8: Distribution of Gyroscope Absolute Rotation Degree Per Second](image-url)
Figure 3.8 shows the CDF distribution of total rotation degree per second for gyroscope. We observe that among 94% of seconds, users rotate the phone less than 0.5 degree. This indicates users carry the phone quite stably during video chat to avoid poor video quality.

**Taxonomy analysis** aims to characterize the key user behaviors when using the mobile video chat application. Due to the large scale of the data we have collected (4,632 users and 1.7 million sessions), it is infeasible to label all the data. In addition, we have observed earlier that the majority of the sessions were short sessions. Therefore, we decide to focus our taxonomy analysis on **meaningful sessions** sampled from the overall data set, i.e., sessions that last 60 seconds or longer. Our reasons for focusing on meaningful sessions are three-fold: (1) We want to understand what user behavioral characteristics promote longer and potentially more effective video conversations; (2) Sessions that last at least 60 seconds contain at least 3 snapshot images and 2 audio samples, which provide adequate information to label each session; (3) There are almost 8,000 meaningful sessions in our data set, which are sufficient for our analysis. Among all the meaningful sessions, we randomly sample 1/40 of the sessions. After removing noisy sessions whose snapshots have no content, we have a set of 218 meaningful sessions with 2,113 images in total. Using the taxonomy we have defined, we then manually label each individual image. As later in the meaningful user behavior analysis, we learn that the snapshots as good representatives of user behavior are pretty similar within meaningful sessions, we apply majority voting to the image-based labels to determine the taxonomy labels for each session. There are two exceptions. First, camera position has a higher variance in sessions, so we ignore it in session-based analysis. Second, to label normal and misbehaving taxonomy for sessions – if any image in a session is labeled as misbehaving, then the whole session is labeled as misbehaving. For each session, we also considered text and local vs. remote stop information.

The taxonomy distributions of meaningful session images and meaningful sessions are summarized in Table 3.3 and Table 3.4, respectively. We can see that most meaningful sessions and their images contained a person, single user, normal user, user face, silence/noise or voice. Also, the front camera is used much more often than the back camera. It is interesting to note that among the meaningful sessions, there are fewer female sessions than male sessions, but more female images than male images. The reason is that female users are possibly more popular and tend to have longer video chat sessions, thus each session contains more
Table 3.3: Taxonomy Distribution of Meaningful Session Images

<table>
<thead>
<tr>
<th>Taxonomy</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>No Person</td>
</tr>
<tr>
<td></td>
<td>594</td>
</tr>
<tr>
<td>Group</td>
<td>No Person</td>
</tr>
<tr>
<td></td>
<td>594</td>
</tr>
<tr>
<td>Normal</td>
<td>No Person</td>
</tr>
<tr>
<td></td>
<td>594</td>
</tr>
<tr>
<td>Face</td>
<td>No Person</td>
</tr>
<tr>
<td></td>
<td>594</td>
</tr>
<tr>
<td>Gender*</td>
<td>No Person</td>
</tr>
<tr>
<td></td>
<td>594</td>
</tr>
<tr>
<td>Camera</td>
<td>Back 653</td>
</tr>
<tr>
<td>Position</td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td>No Data 687</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*There are 107 images for which we could not determine the gender.

Table 3.4: Taxonomy Distribution of Meaningful Sessions

<table>
<thead>
<tr>
<th>Taxonomy</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>No Person</td>
</tr>
<tr>
<td></td>
<td>62</td>
</tr>
<tr>
<td>Group</td>
<td>No Person</td>
</tr>
<tr>
<td></td>
<td>62</td>
</tr>
<tr>
<td>Normal</td>
<td>No Person</td>
</tr>
<tr>
<td></td>
<td>62</td>
</tr>
<tr>
<td>Face</td>
<td>No Person</td>
</tr>
<tr>
<td></td>
<td>62</td>
</tr>
<tr>
<td>Gender</td>
<td>No Person</td>
</tr>
<tr>
<td></td>
<td>62</td>
</tr>
<tr>
<td>Text</td>
<td>No Text 90</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop</td>
<td>No Data 49</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td>No Data 4</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.3.2 Mobile-Online Video Chat Comparison

We compared both mobile and online user behavior in the Omegle application, with the caveat that the information we obtain from online users is less extensive than our mobile data collection. This is because we rely on a randomized image data set provided to us by Omegle, from which we could derive such properties as gender proportion, but not session duration for example. While we could have built our own instrumented web application compliant with Omegle to measure these factors, we feel the adoption rate would not have been strong since web browsers already connect with Omegle. In contrast, our mobile application introduces new capabilities - namely mobile video chat - that drives adoption.

Figure 3.10 compares the gender distributions of mobile and online Omegle users. As shown in the figure, mobile users and Omegle users share similar ratio between number of male and female users that the mobile users are about 49% male and 12% female, and Omegle online users are 68% male and 17% female. However, mobile users have a much higher fraction of “other” types of content in which users do not show their appearances for us to determine their genders. Due to the portable feature of mobile video chat, users can engage in video chat sessions at varying locations and with different postures. Besides, many smartphones are equipped with both front and back cameras, allowing users to switch between the two cameras and show different content during video chat sessions. As a result, with mobile video chat, we expect to see more diverse “other” types of content than that of online video chat, such as the sample images shown in Figure 3.9.

![Sample images collected by our app showing more diversity in mobile video chat than those in online video chat.](content_in_bathroom.png, quilt.png, trademark.png, partial_legs.png)

Figure 3.9: Sample images collected by our app showing more diversity in mobile video chat than those in online video chat.
Previous work [63] learnt that there are nearly 24% of users showing offensive content and 5.6% of users showing potential offensive content in the services such as Chatroulette and Omegle. In our dataset, we also find that similarly, 23.5% of mobile users are misbehaviors. This also indicates that the “flasher” problem has also propagated to the mobile random video chat realm and caused nearly the same extent of troubles. However, because previous works [63, 20, 64, 38] detect misbehaviors based on the presence of facial features, the high diverse content in mobile video chat could cause poor performances when applying them on the mobile platform.

We also compare the time of use, i.e., day of week and hour of day, between mobile and online Web users. The results are shown in Figure 3.11. Omegle is a popular online video chat service, and its number of concurrent users vary between 15,000 and 45,000. For our application, the number of users at any given hour of day and given day of week varies between 15 and 80. The general trend for hour of day is similar for both mobile and online users. Overall, online users are more uniformly distributed across different time, while mobile usage sees more fluctuation (partly due to the smaller number of mobile users) and higher usage in the afternoon and late evening hours. One significant difference is on day of week: online usage is much higher on Friday evenings and most of Saturdays, while mobile usage actually sees lower activity during these two days. This may be due to different user populations for the mobile and online worlds, e.g., mobile users are more likely to be out partying on Fridays and Saturdays.
3.3.3 Taxonomy Correlation Analysis

Our taxonomy aims to characterize users’ mobile video chat behavior from multiple dimensions. Given the labeled taxonomy information, one important question we want to answer is **which user characteristics are correlated**. In other words, we want to identify behavioral characteristics which are likely or unlikely to occur together. For instance, do male and female users behave differently, or how do normal users behave compared to misbehaving users. Understanding such taxonomy correlations can offer useful insights into designing better user pairing strategies, misbehavior detection mechanisms, etc.

Since our taxonomy contains categorical rather than numerical attribute values, we utilize four correlation metrics that are typically used for categorical correlation analysis: $\chi^2$, $lift$, $all\_confidence$ (or $all\_conf$), and $cosine$. Let $A$ and $B$ be two attributes (e.g., gender and camera position) with values
$a_i (1 \leq i \leq c)$ and $b_j (1 \leq j \leq r)$ respectively, these four metrics are defined as follows:

\[
\chi^2 = \sum_{i=1}^{c} \sum_{j=1}^{r} \frac{(n_{ij} - e_{ij})^2}{e_{ij}}
\]

\[
e_{ij} = \frac{\text{count}(A = a_i) \times \text{count}(B = b_j)}{N}
\]

\[
\text{lift}_{ij} = \frac{n_{ij}}{e_{ij}}
\]

\[
\text{all}_{\text{conf}}_{ij} = \frac{n_{ij}}{\max\{\text{count}(A = a_i), \text{count}(B = b_j)\}}
\]

\[
\text{cosine}_{ij} = \frac{n_{ij}}{\sqrt{\text{count}(A = a_i) \times \text{count}(B = b_j)}}
\]

Here $N$ is the total number of samples, $n_{ij}$ is the number of samples with both $A = a_i$ and $B = b_j$, $\text{count}(A = a_i)$ and $\text{count}(B = b_j)$ are the numbers of samples with $A = a_i$ and $B = b_j$ respectively.

$\chi^2$ measures the difference between observed values $n_{ij}$ and expected values $e_{ij}$ (if $A$ and $B$ are not correlated). So a small $\chi^2$ value (close to 0) means non-correlation while a high $\chi^2$ value indicates possible correlation. Similarly, a $\text{lift}$ value of 1 means no correlation ($n_{ij} = e_{ij}$), and a $\text{lift}$ value $> 1$ (or $< 1$) indicates positive (or negative) correlation. However, both $\chi^2$ and $\text{lift}$ are sensitive to skewed distribution of the attribute values (e.g., most user are normal or very few users chat as a group). By focusing on $a_i$ and $b_j$ values and ignoring other values (i.e., null-invariant), $\text{all}_{\text{conf}}$ and $\text{cosine}$ can tolerate different data set scales and skewed attribute value distributions. Generally, $\text{all}_{\text{conf}}$ and $\text{cosine}$ values that are close to 1 indicate strong positive correlation and values that are close to 0 indicate strong negative correlation. In our analysis, we leverage $\chi^2$ and $\text{lift}$ to confirm non-correlation (i.e., $\chi^2$ is close to 0 and $\text{lift}$ is close to 1), and leverage $\text{all}_{\text{conf}}$ and $\text{cosine}$ to identify strong positive correlation (i.e., both $\text{all}_{\text{conf}}$ and $\text{cosine}$ are $> 0.85$) and strong negative correlation (i.e., both $\text{all}_{\text{conf}}$ and $\text{cosine}$ are $< 0.1$).

Note that correlation is bi-directional: if $a_i$ and $b_j$ are positively (negatively) correlated, seeing $a_i$ means that $b_j$ is more (less) likely to occur, and vice versa. However, some relations between user behaviors can be unidirectional, e.g., if $a_i$ occurs, $b_j$ is likely to occur but the reverse may not be true. We use association rules (e.g., $a_i \Rightarrow b_j$) to capture such one directional relations, and the conditional probability $Pr(b_j|a_i)$ is referred to as the confidence of an association rule.
Figure 3.12: Taxonomy correlation analysis: (L) image-based graph and (R) session-based graph. Highlighted in the graphs are strong positive correlations (both all\_conf and cosine are > 0.85), strong negative correlations (both all\_conf and cosine are < 0.1), some non-correlations ($\chi^2$ is close to 0 and lift is close to 1), and association rules with confidence > 0.8.

Figure 3.12 summarizes the key results of our taxonomy correlation analysis, including strong positive (negative) correlations, some (surprising) non-correlations, and one-directional association rules with high confidence values. Our correlation analysis has been conducted at both the image level and session level. Image-based analysis broadly refers to the multi-modal set of data closest in time to an image snapshot, including the audio snippet and sensor readings immediately preceding a snapshot. For session-based labelling, since the variance of image label within a session is very small, we adapt major voting method to find dominant class among image labels as the label for a session. While normal/misbehavior labeling is an exception that once one image in a session is labeled as misbehavior, the whole session is treated as misbehavior. Note that our sampled meaningful session data set contains 218 sessions and 2,113 images, so the image-based analysis is more stable, but the session-based analysis still offers some important insights. Next, we describe in detail the image-based correlation analysis results, then discuss the differences in the session-based correlation analysis results.

**Camera position**, i.e., front or back camera, is a key feature that allows users to show different content. As shown in Figure 3.12(L), **Front Camera** has strong positive correlations with **Person** (seeing
person in the image), **Face** (seeing face in the image), and **Normal** (normal user), and strong negative correlation with **No Face** (not seeing face in the image); while **Back Camera** has strong positive correlation with **No Face** and strong negative correlation with **Face**. This can be explained by the notion that people typically use the front camera to show their own faces and the back camera to show some other content. In addition, the observation that normal users and front camera have a strong positive correlation can be used to differentiate normal users from misbehaving ones. Since checking camera position on smartphones is an inexpensive operation, this can be particularly useful for misbehavior detection in mobile video chat services.

**Face** appearance in video chat is another important factor to consider. In video chat services, people seek other interesting people to chat with, and showing their faces help keep people engaged in video chat sessions. As shown in Figure 3.12(L), **Face** has a strong positive correlation with **Normal** and a strong negative correlation with **Misbehaving**. In other words, normal users tend to show their faces while misbehaving users tend not to show their faces. We believe the explanation is three-fold: (1) Normal users show their faces so they can chat more effectively with their partners; (2) Misbehaving users tend to hide their faces to avoid being identified; and (3) Due to the limited aperture angle of mobile phone cameras, it is difficult for a misbehavior to show both his/her face and private body parts. Given the strong positive (negative) correlations between Face and Normal (Misbehaving), image-based face detection can be quite effective for differentiating normal users from misbehaving ones [63, 64, 38].

**Group** chatting is rare in video chat services and most users choose to chat alone with their remote partners (Table 3.3, Table 3.4). Still, when group chatting does occur, we observe that **Group** has strong negative correlations with **Back Camera**, **Music & Sound**, **Silence/Noise**, and **Male**. In other words, when people chat as a group, they are less likely to use back camera, have background music/sound or silence/noise. It is also interesting to observe that male users tend not to chat in groups. In addition, based on the association rules shown in Figure 3.12(L), when users chat as a group, they are very likely to show their faces and are very likely to be normal.

**Female** users also have some interesting behavioral characteristics: They are likely to use the front camera and show their faces (Figure 3.12(L)). Specifically, based on our sampled data set, the probability for
a female user to use the front camera is 92% and the probability for a female user to show her face is 84%. The intuition is that female users tend to be popular in video chat services, i.e., more people are interested in talking to female users. Therefore, using the front camera and showing their faces can help female users to chat more effectively with their remote partners.

**Non-correlations** can sometimes indicate something interesting as well. For instance, we observe no negative correlation between **No Person** and **Voice**, i.e., even when there is no person shown in the video chat images, there can be human voice in the audio recordings. This may indicate that users sometimes show other content (not themselves) to their remote partners while talking. Another example is Gender versus Normal/Misbehaving. We originally expected to see **Male** being correlated with **Misbehaving**, i.e., male users are more likely to misbehave and misbehaving users are more likely to be male. However, we did not find such correlation in our data set. In other words, a misbehaving user can be male or female, and knowing the gender of a user does not increase or decrease his/her probability of misbehaving.

**Session**-based correlation analysis reveals fewer relations than image-based correlation analysis (Figure 3.12). This is mainly due to the fact that there are a smaller number of sessions than images. Most of the session-based relations are similar to image-based relations. One correlation that is specific to session-based analysis is the strong negative correlation between **Group** and **Remote Stop**, i.e., video chat sessions of group users are unlikely to be stopped by the remote party. The fact that people do not hang up when talking to a group of users on the remote side is interesting. Although group chatting is rare in our current data set, they do seem to keep people more engaged in a video chat session.

### 3.3.4 Acceleration Analysis

For mobile video chat, acceleration information can be easily collected and can offer useful contextual information such as how the phones are positioned/oriented during video chat. The acceleration analysis in this section tries to explore how accelerometer sensor data can help identify certain user behavioral characteristics like misbehaving in mobile random video chat services.

Figure 3.13 shows the acceleration distributions among our meaningful session dataset. It is important to note that the Adobe AIR API reverses the $x$ and $y$ axes used in the Android API, and refers to the long
side of the mobile phone panel as the $x$ axis (and the short side as the $y$ axis). Since MVChat organizes its video chat screen using the landscape layout, users tend to hold the phone horizontally and potentially rotate along the $x$ axis. As a result, acceleration centers around 0 for the $x$ axis, centers around $G$ for the $y$ axis, and spreads between $-0.5G$ and $G$ for the $z$ axis during video chat. Based on the acceleration distribution, we define four different classes to identify different acceleration orientations based on the degree of rotation along the $x$ axis. The orientation class definitions are shown in Figure 3.14, that the phone can be placed horizontally facing up and rotated along the long-side phone panel (tilting) until facing down to cover orientation Class 1, Class 3 and Class 2 sequentially. The rest phone orientations are defined as Class 4.

Figure 3.13: Acceleration distribution of meaningful sessions.
Table 3.5 lists the number of occurrences of different acceleration orientation classes among the sampled meaningful session data. We found that Class 1 and Class 2 occur much more frequently than the other two classes. Furthermore, we combine the acceleration orientation class information with camera position, and examine the corresponding user characteristic in terms of being normal or misbehaving. The results are shown in Table 3.6. As we can see that,

- Normal users are more likely to use phones with the front camera in acceleration orientation classes 1 and 2.
- Misbehaving users are more likely to hold phones in acceleration orientation classes 2 or use phones
with the back camera in acceleration orientation classes 1.

Since acceleration data is easy to collect and lightweight computed, these observations can be potentially leveraged to design effective and efficient misbehavior detectors for mobile video chat services.

Table 3.6: Combining Acceleration Orientation and Camera Position to Detect Misbehaving Users

<table>
<thead>
<tr>
<th>Acc Orientation Class</th>
<th>Camera Position</th>
<th>Taxonomy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Misbehavior</td>
</tr>
<tr>
<td>Class 1</td>
<td>Back Camera</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>Front Camera</td>
<td>12</td>
</tr>
<tr>
<td>Class 2</td>
<td>Back Camera</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>Front Camera</td>
<td>97</td>
</tr>
<tr>
<td>Class 3</td>
<td>Back Camera</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Front Camera</td>
<td>0</td>
</tr>
<tr>
<td>Class 4</td>
<td>Back Camera</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Front Camera</td>
<td>4</td>
</tr>
</tbody>
</table>

3.4 Conclusions

In this chapter, we have presented the results of a large scale study of mobile users behavior on an Android video chat application, which we design and deploy for random video chat with Omegle online users. We summarize our key findings as follows. First, we find that the majority of sessions were short, as “pure” random pairing is not effective that users search exhaustedly for a desirable match with whom they can chat for a longer time. And text message is not as frequently used on mobile random video chat as it on web video chat. The existence of “flasher” problem is also propagating to mobile video chat. Besides, mobility introduces more diversity in mobile random chat content. And normal users are highly correlated with using the front camera and showing their faces, whereas misbehaving users tend to hide their faces – which suggests the exploration of camera position and face detection for distinguishing normal users from
misbehaving ones. Additionally, groups typically imply normal behavior. Besides, surprisingly females are just as likely to misbehave as males. Finally, mobile sensor data such as acceleration provides useful contextual information, which could help to identify some user behavioral characteristics and further solve certain issue such as misbehavior detection.
Chapter 4

Meaningful User Behavior Analysis

Insights gained from our user behavior analysis (Chapter 3.3) have shown that “pure” random pairing is not quite effective for a user in the sense of generating lots of short sessions until “right” pairing shows up in a video chat session. As a result, people take great efforts going through many random pairing sessions in order to find someone interesting to chat with for a longer duration. This problem harms the user experience and inspires us to better understand key characteristics of user behavior which could help users to promote long sessions in random video chat services.

4.1 Related Work

Plenty of works focus on building effective matching algorithms for online social media. Kuan et al. [23] tries to accurately predict the male-female conversation compatibility on the Chatous, a text-based one-on-one anonymous chat network, by modeling the network as a bipartite graph based on user profiles. Similarly, Alex et al. [2] seeks to accurately predict whether a random pair of users will have a positive or negative conversation based on the chat histories and the profiles of users from the Chatous. Work by Hitsch [29] estimates mate preferences, and uses the Gale-Shapley algorithm to predict stable matches for online dating service. Another study by Samir et al. [36] comes up with an on-line deterministic algorithm for the solving weighted bipartite matching problem and the stable marriage problem. However, building such an effective pairing mechanism is challenging for our current system in three aspects:

- **Privacy**: Pairing algorithms require the context information from both sides of a conversation. Due to privacy consideration, our current system is only allowed to collect data from our local mobile
client.

- **Anonymity**: Random video chat services such as Chatroulette and Omegle allow users to chat without any personal information, which makes it hard to extract user profiles.

- **Dynamic**: Popular services like Chatroulette and Omegle have a fairly large and dynamic user set, and hundreds of users could switch status within minutes at peak time.

All of these challenges prevent us from extracting user preferences and interests to design an effective pairing algorithm for mobile random video chat services.

To overcome this great challenge, we observe that there is a small portion of users acting more effectively than the others (the majority) in promoting longer duration. We call them the “Meaningful Users” in our study. In this chapter, we aim to understand the key characteristics of effectiveness among these users’ behavior. We hope the findings learned from this meaningful user behavior analysis could guide our users to be more effective in the video chatting.

The goal of the study in this chapter is to extract the key characteristics that are commonly shared among the meaningful users’ behavior and also differentiate behavior of other type of users. However, before we dive into exploring behavioral characteristics of our meaningful users, we first demonstrates that users could behave consistently within every session. This is necessary and important because it guarantees the data we use for analysis is a good representative of common user behavior. In this chapter, our study consists of three components:

- **Session-based Behavior Consistency**: aims to analyze whether users behave consistently within a session.

- **User-based Behavior Consistency**: explores whether meaningful users behave consistently between their meaningful sessions and non-meaningful sessions.

- **Behavior Effectiveness Analysis**: investigates the key characteristics of effectiveness of meaningful users’ behavior. For example, whether certain taxonomy such as gender could promote longer sessions.
4.2 Session-based Behavior Consistency

To explore the key characteristics of effectiveness of meaningful users, we need to first extract a good representative of user behavior. Among all kinds of sensor data we collect, we think snapshots are the best indicators, because compared to other sensors, snapshots have abundant user context, like gender and appearance. They also play a very important role for users to decide whether to continue the sessions as, lots of users maintain a session based on the first impression gained from the snapshots. However, a user could have hundreds of snapshots from multiple sessions in our dataset and even within a long session, there could be several snapshots captured from it, so our first task aims to analyze whether users behave consistently within a chat session. If users behave quite consistently within sessions, then we can select several, or even one snapshot, to represent user behavior within a session.

In our study, we compute the similarities between consecutive snapshots within a session to measure the consistency of user behavior. We implement four different algorithms to measure the similarities between snapshots in different aspects and summarized as below:

- **Grayscale Histogram**: first measures the histogram of an image in grayscale. Then the histogram is grouped evenly into 64 bins as the feature vector. Finally we compute the cosine similarity between the feature vectors of two consecutive images as the similarity between the two images.

- **DCT-based Perceptual Hashing**: [66] first resizes an image to 32 by 32 pixel and applies two-dimensional DCT on the resized image. Then, the mean value of the 8 by 8 low-frequency coefficients is calculated. Next, the 8 by 8 low-frequency coefficients are normalized to a 64-sized binary vector by comparing with the median value. Finally, hamming distance is computed between two consecutive images’ binary vectors as image similarity of them.

- **Grid-based RGB Histogram**: first divides an image into m by n grids. The means of the RGB colors in each grid are calculated respectively to form a 3 by m by n length feature vector. Finally, cosine similarity is computed as the similarity between images.

- **SIFT Feature Matching**: [39] first extracts SIFT features of two consecutive images. The Eu-
clidean distance is then computed between all pairwise SIFT features from the two images. Next, for each SIFT feature in the first image, we pick up its shortest and second shortest Euclidean distances and a match is detected only if the ratio of the two distances is less than 0.6. Finally, image similarity is measured by the percentage of matched SIFT features in the first image.

As we described above, the four algorithms measure the image similarity in different aspects varying from the RGB scale to the grayscale, from the time domain to the frequency domain. The Grayscale Histogram and the Grid-based RGB Histogram both explore image similarity in the time domain but vary from the grayscale to the RGB color scale. However, the DCT-based Perceptual Hashing, which is also used by Google Image Search, analyzes image similarity in the frequency domain. Finally, the SIFT Feature Matching algorithm focuses on exploring SIFT features which are invariant to scaling, rotation and illumination changes. These ensure that we have a full understanding of user behavior consistency within video chat sessions from all aspects.

We apply the four algorithms onto the same meaningful session dataset we use in the taxonomy analysis. And to measure the overall user behavior consistency within sessions, we compute the average of all the image similarities in our meaningful session dataset. The average similarities have been shown in Table 4.1. To give a better insight of the result, we pick up 3 sample images (Figure 4.1) and also present their similarities in Table 4.1. The sample image 1 and sample image 2 are consecutive and similar images collected from the same session while the sample image 3 is a totally different image captured in another session by a different user. By comparing the overall results with selected images’ similarities, we observe that the overall image similarities achieve similar or even slightly better performances than similarities between sample image 1 & 2, and far better than those results between sample image 1 & 3. So we can draw a conclusion that people generally behave consistently within their video chat sessions.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Overall Average</th>
<th>Sample 1 vs Sample 2</th>
<th>Sample 1 vs Sample 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grayscale Histogram</td>
<td>81.0%</td>
<td>62.1%</td>
<td>38.2%</td>
</tr>
<tr>
<td>DCT-based Perceptual Hashing</td>
<td>83.7%</td>
<td>81.8%</td>
<td>59.3%</td>
</tr>
<tr>
<td>Grid-based RGB Histogram</td>
<td>81.5%</td>
<td>88.2%</td>
<td>73.4%</td>
</tr>
<tr>
<td>SIFT Feature Matching</td>
<td>16.7%</td>
<td>10.6%</td>
<td>0%</td>
</tr>
</tbody>
</table>
4.3 User-based Behavior Consistency

Based on the effectiveness of maintaining meaningful sessions (>30 seconds), we divide our users into three types:

- **Meaningful Users**: are users for whom more than 10% of sessions are meaningful. Given that most users have less than 5% probability to have sessions longer than 30 seconds, 10% is considered significant.

- **Common Users**: are non-meaningful users who have at least one meaningful session

- **Meaningless Users**: are the remaining users who have no meaningful sessions.

We filter out users who have less than 10 sessions in our dataset to ensure that all the selected users are active in participating video chatting. For each selected user, we extract all his meaningful sessions and randomly sample 5 non-meaningful sessions. Since we have proved that users behave consistently within sessions, we only sample one snapshot from each session. After filtering out the noisy users whose snapshots have no content, we pick up 95 meaningful users, 119 common users and 75 meaningless users. Similar to our meaningful session analysis, we label the snapshots using 5 different taxonomies: a) human existence; b) face existence; c) gender; d) normal existence and e) camera position.

Before we start to explore the key taxonomy characteristics of effectiveness among meaningful users, we are also curious about the question: do users, especially meaningful users behave differently between their meaningful sessions and non-meaningful sessions? In other word, how about the behavioral consistency among sessions? The study conducted in this chapter aims to eliminate the possibility that meaningful
users promote longer sessions because of different behavior patterns in meaningful sessions. And since in our study, we are only interested in the taxonomy characteristics of user behavior, instead of measuring the behavioral consistency among sessions by the four image similarity algorithms we use in Chapter 4.2, we calculate the similarities of taxonomy proportions between meaningful sessions and non-meaningful sessions on both the meaningful and the common users. Note that since meaningless users have no meaningful sessions, they don’t have these similarities. For each taxonomy, the detail of our behavior consistency measurement is shown as follow:

1. For each user, we first compute his taxonomy proportions on both meaningful sessions and non-meaningful sessions.

2. For each user, similarity is calculated between his meaningful session taxonomy proportion and his non-meaningful session taxonomy proportion.

3. For each type of users, we measure the average of all taxonomy proportion similarities from its users as behavior consistency among sessions.

It is important to note that the similarities of human existence, face existence, normal existence and camera position are measured by the absolute difference, that if a similarity is closed to 0, it means a consistent behavior pattern. On the contrary, the similarities of gender and overall difference are measured by cosine similarity (Equation 3.7), that if a similarity is closed to 1, it indicates a consistent behavior pattern. Table 4.2 shows result of our behavior consistency measurement on different taxonomies. Given the great diversity of mobile image content, we believe that both meaningful and common users behave consistently among all their sessions regardless of being meaningful or not.

<table>
<thead>
<tr>
<th>Taxonomy</th>
<th>Meaningful User User</th>
<th>Common User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>0.088</td>
<td>0.070</td>
</tr>
<tr>
<td>Face</td>
<td>0.098</td>
<td>0.106</td>
</tr>
<tr>
<td>Gender</td>
<td>0.994</td>
<td>0.964</td>
</tr>
<tr>
<td>Normal</td>
<td>0.068</td>
<td>0.051</td>
</tr>
<tr>
<td>Camera Pos</td>
<td>0.049</td>
<td>0.071</td>
</tr>
<tr>
<td>Total</td>
<td>0.980</td>
<td>0.973</td>
</tr>
</tbody>
</table>
4.4 Behavior Effectiveness Analysis

After illustrating that mobile users follow the consistent behavior pattern during the video chatting, we finally focus on exploring the key characteristic of effectiveness for maintaining longer video chat sessions. In this chapter, we will answer the question whether certain behavioral taxonomy facilitates users to maintain longer sessions? In other words, is there any great difference in certain taxonomy proportions among different types of users?

To explore whether certain behavioral taxonomy could promote longer sessions, for each taxonomy, we compare its proportion distributions on different types of users as follows:

(1) Similar to what we have done in Chapter 4.3, for each user, we first compute his taxonomy proportions on both meaningful sessions and non-meaningful sessions.

(2) For each type of users, we measure the taxonomy proportion distributions from all its users on both meaningful sessions and non-meaningful sessions.

(3) For each taxonomy, we compare its proportion distributions on different types of users including:
   a) meaningful users’ meaningful session distribution, b) common users’ meaningful session distribution and c) meaningless users’ non-meaningful session distribution.

Figure 4.2: (L) face existence distribution and (R) person existence distribution on different types of users.
From Figure 4.2(L), we can learn that nearly 60% of meaningless users have 60% probability to hide their faces, while only about 30% of meaningful users have 60% probability to hide their faces. This observation tells that showing face could, to some extent, help users to promote longer sessions. We think this is because generally people decide whether to maintain a session based on the first impression. Hiding facing prevents people making a judgement based on it and could reduce the chance to continue a conversation. A similar trend is also observed in Figure 4.2(R) that person existence could slightly help users to promote longer sessions.

Figure 4.3 reveals that female are pretty popular to maintain a meaningful session because almost 40% of meaningless users have less than 40% confidence to be female, while nearly 90% of meaningful
users have less than 80% confidence to be female. To further demonstrate this observation, we conduct another experiment, that redefines the meaningful users as users for whom more than 10% sessions are longer than 30 seconds. Based on this definition, we pick up 114 new defined meaningful users from our dataset. And for comparison purposes, we also sample a similar number of users from the rest “non-meaningful users”. For each of these meaningful and non-meaningful users, we randomly select about 20 images from their sessions. As a result, we have obtained 2,714 images for 114 meaningful users and 2,832 images for 123 non-meaningful users. We label these images for the gender taxonomy and apply majority voting to determine the gender for each user. By comparing their gender proportion, we also find the same trend that gender is the most dominating factor. In this new set of meaningful users, we have 10 males and 69 females (12.7% vs. 87.3%), while in the set of non-meaningful users, we have 56 males and 9 females (86.2% vs. 13.8%). Note that the remaining 35 ‘meaningful users’ and 58 ‘non-meaningful users” images are absent of person content so we cannot identify their genders. Furthermore, a similar gender distribution also occurs when we apply the 60 second definition to our dataset, where the skew is even more significant: 2 males and 20 females (9.1% vs. 90.9%). We believe this is partially due to the fact that there are a lot more male users than female users in video chat services (Figure 3.10), and male users appear to be more interested in talking with female users.

Finally, we also learn from Figures 4.4 that using front camera could slightly contribute to maintaining a longer session.

### 4.5 Conclusions

In this chapter, we conduct an in-depth meaningful user behavior analysis. First, we illustrate that users behave consistently within sessions by measuring the image similarities within each session using four different algorithms. We further prove that users even behave consistently among all their sessions regardless of being meaningful or not. Finally, our analysis reveals that females are highly popular in mobile random video chat service, in that users with a large enough fraction of sustained sessions are disproportionately female. And showing face with front camera provides a first impression to others and help users to maintain longer sessions.
Chapter 5

Flasher Detection

The existence of “flashers”, or misbehaving users who expose their private body parts, has been a serious problem in online video chat services which support random video chat among strangers. Moreover, as it is shown in Chapter 3.3, the presence of misbehaving problem is also propagating to the mobile random video chat realm. Our preliminary study in Chapter 3.3.3 and Chapter 3.3.4 have provided the first insight into user behavior study at scale in such kind of service, assessing a variety of correlative factors that occur between mobile sensor data and the two types of user behavior, namely normal and misbehaving user behavior. It tells us some potential features we could leverage to better solve this problem on mobile platform. However, that study does not systematically address the problem of detecting flashers with high accuracy and efficiency.

In this chapter we will describe our design, development and evaluation of mobile flasher detection classifier that uniquely incorporates multi-modal mobile sensor data to substantially improve the classification accuracy and efficiency of mobile random video chat users, based on a large scale data set derived from our MVChat service.

In our work, we follow the common definition used in the previous researches [31, 3, 20, 64, 38] and in our preliminary user behavior study, that misbehaviors (or flasher) are defined as users who expose their bodies to show obscene content to others, such as males/females showing their lower body part or females exposing their chests.
5.1 Related Work

Previous works [63] [20] [64] [6] [38] have extensively researched detecting misbehaviors in online random video chat service by building some pure image-based classifiers. The main property that is exploited in the design of these classifiers is the relationship between the presence of certain facial features and user behavior: online users who display certain facial features are found to be correlated with normal behavior, whereas online users who do not show them are more likely to be associated with flashing behavior. For details, SafeVchat [63] fuses multiple facial evidences (faces, eyes, upper body, etc.) into a probabilistic model using Dempster-Shafer Theory to classify normal and misbehaving users. A fine-grained cascaded classifier (FGC) is proposed to speed up compute-intensive processing (such as Dense SIFT, HOG) for classification [20]. However, this cascaded classifier is limited to local optimization, as it cannot handle the huge combinatorial number of feature set permutations. Besides, FGC only uses the normal prediction result as the default stopping condition without taking into account the confidence of predictions. EMeralD [64] proposes a more coarse-grained two-stage classifier (a rule-based pre-classifier as the front stage and a high complexity binary logistic regression model as the back stage) to achieve low computation and high accuracy in misbehavior classification.

We have obtained the code for the latest misbehavior classifier from Chatroulette. This state-of-the-art algorithm is summarized in Algorithm 1 and referred to as the “CR” algorithm. It is an enhanced version of pure image-based misbehavior classifier combining works of SafeVChat [63], EMeralD [64] [38] and FGC misbehavior classifier [20] to tradeoff between efficiency and accuracy. We observe that this algorithm achieves promising scalability and accuracy by simplifying Emerald’s rule-based approach that removes the binary logistic regression component. Chatroulette reports that such an approach reduces their number of server instances by a factor of three from over 100 servers to somewhat over 30 servers.

In detail, the CR algorithm extracts one facial descriptor at a time and checks whether the generated feature satisfies its predefined normal user association rules related to the existence, number of instances and the spatial distribution. For example, the FaceRule checks whether there are two or more faces detected or whether any detected face’s width is more than 2/7 of the longest diagonal between the face center and
four image points; the EyeRule checks whether a pair of eyes has been detected and whether both eyes are sufficiently small (less than 0.03 of image size) and close to each other; UpbRule checks the existence of upper body whose size is greater than 0.4 of image size. If the rule is not satisfied, CR algorithm will continue to the next stage and generate another feature, examine it with the new rule until the feature set exhausted. These features and their association rules are ordered according to their relevances in identifying normal users, with the most relevant features/rules listed at the top. This algorithm produces up to 13 labels with labels such as Face (1), Eyes (2), Upper Body (3), etc. With these output labels, then it is up to Chatroulette to decide the split point at where to start classifying users as misbehavior based on its need.

There are very limited studies on user behavior in mobile random video chat. An outline of mobile video chat issues and challenges is presented in [34]. We have conducted some brief analysis in Chapter 3.3 to understand normal and misbehaving users from the data obtained by our MVChat mobile random video chat application. However, these analysis only reports the potential correlation statistics but does not proceed to the next step of developing and evaluating a concrete classifier operating on actual mobile video chat data.

Besides all the sensors we have investigated, audio samples collected by mobile phones have also been widely studied to provide phone contexts and to support new services. SoundSense [40] presents a scalable sound prediction architecture and applies it to a daily audio diary application and a music detector application. SwordFight [67] uses audio tones exchanged between phones to localize each other and support a real-time mobile motion game.

A variety of algorithms for fusing multiple classifiers have been studied in the literature and are available for use in the Weka toolkit [62]. These include the J48 Decision Tree [46] [52], Random Forest [12], AdaBoost [68], Bootstrap aggregating (Bagging) [11] and Naive Bayes [45] [52]. And others apply a layered probabilistic representation of Hidden Markov Models to fuse multimodal sensing at multiple levels of temporal granularity to recognize office activities [49] [48] [47].
**Input** $img$: snapshot image to be classified;

**Output** $1 \sim 13$: “Normal user” prediction confidence from high to low;

\[
\text{faces} \leftarrow \text{FaceDetect}(img);
\]

\[
\text{if FaceRule(faces) then}
\]
\[
\begin{align*}
\quad & \text{return } 1 \\
\end{align*}
\]

\[
\text{else}
\]
\[
\begin{align*}
\qquad & \text{eyes} \leftarrow \text{EyeDetect}(img);
\end{align*}
\]

\[
\text{if EyeRule(eyes) then}
\]
\[
\begin{align*}
\quad & \text{return } 2 \\
\end{align*}
\]

\[
\text{else}
\]
\[
\begin{align*}
\qquad & \text{upbs} \leftarrow \text{UpbDetect}(img);
\end{align*}
\]

\[
\text{if UpbRule(upbs) then}
\]
\[
\begin{align*}
\quad & \text{return } 3 \\
\end{align*}
\]

\[
\text{else}
\]
\[
\begin{align*}
\qquad & \text{mouths} \leftarrow \text{MouthDetect}(img);
\end{align*}
\]

\[
\vdots
\]

\[
\text{end}
\]

\[
\text{end}
\]

**Algorithm 1:** CR algorithm: State-of-the-art classification algorithm for flasher detection in online video chat.

Finally, to improve the efficiency and reduce the computation required for sensing and processing, researchers have conducted studies to explore some policies based on expected-value-of-information (EVI) for selective perception [48] [47].

### 5.2 New Problem for Mobile Random Video Chat

Before developing our own classifier to identify misbehaving users in mobile random video chat application, we want to first investigate whether previous image-only face-centric classifiers developed for
online webcam-based video chat services are sufficiently accurate for mobile random video chat services. In other words, we want to first answer the questions: in the new realm of mobile video chat, is the presence or absence of facial features still as strong an indicator of a normal or misbehaving user?

To measure the performance of existing pure image-based misbehavior classifiers, we choose as our baseline the CR algorithm, which is state-of-the-art for desktop-based video chat services. As mentioned earlier, CR generates 13 classification labels as output and a split threshold is then used to determine how many features need to be detected for a user to be classified as normal. Figure 5.1 shows the accuracy of CR using different split thresholds on our mobile user data. There is a slight peak at 8, which achieves the best accuracy of $\sim 63\%$ which is not acceptable performance for misbehavior classifier on mobile.

![Figure 5.1: Classification accuracy of the CR algorithm on mobile video chat data using different split thresholds.](image)

We posit the following reasons to explain the relatively limited performance of the CR algorithm:

1. As shown in Figure 5.2 and discussed in our previous study, mobility results in much more diverse image content. The CR algorithm relies heavily on the presence of facial features to predict whether a user
is normal or misbehaving. In the online case, the absence of a face implies a misbehaving user with a very high confidence. However, in mobile video chat, we notice that there are a large number of video sessions that do not contain any faces and at the same time do not contain any objectionable content. Instead, they focus on the background or interesting objects around the users. Our earlier study in Chapter 3.3 shows that in a mobile video chat application, nearly 40% of video chat users show “others” type of content which does not include a human. Also, due to the lack of front camera on some low-end Android devices, some users cannot show their faces while chatting.

(2) Mobility also results in poor quality images that are blurred. Furthermore, the distance between a user and the mobile camera is much shorter than that for desktop webcam-based users. Along with a limited wide angle camera, this results in many partial faces in the images (see Figure 5.3).

(3) Facial feature association rules defined in the CR algorithm are not as applicable on the mobile platform. For example, the EyeRule in the CR algorithm tries to detect a valid pair of eyes, where the distance between the eyes is fixed between 20 and 70 pixels. But since the mobile phone allows the user to put the camera close to their face, the faces can be much larger on the screen than typical webcams, resulting in larger distances between the eyes (Figure 5.3, third image). Also, the UpbRule in the CR algorithm looks for an upper body whose size is bigger than 40% of the image size. For the same reason of closeness between the mobile camera and the user, this rule is less applicable for mobile video chat images.

![Figure 5.2: Examples of mobile phone-captured images that contain objects or background but no human.](image)
In sum, similarly as our study shown in previous chapter (Chapter 3.3), there is a wider diversity of scenes captured by the mobile video chat camera compared with a desktop webcam used by online video chat users. This greater diversity breaks down the relationship between the presences of facial features and normal behavior. That is, in a mobile setting, we find that there are many more kinds of normal behavior that do not show a face and these does not necessarily involve flashing behavior. In addition, there are other challenges introduced by the mobile environment, such as the appearance of partial faces and motion-blurred scenes that rarely occur in desktop milieus. So we have to make a conclusion that current face-centric misbehavior classifications perform poorly on mobile random video chat data.

### 5.3 Flasher Detection Algorithm

Our first focus in this section is to investigate whether additional sensing modalities can be leveraged to improve mobile flasher classification performance. Today’s smartphones provide a wealth of contextual data from their mobile sensors, such as three-axis acceleration, orientation, light intensity, audio, front/back camera state, and image snapshots. We want to characterize which of these mobile sensors helps to meaningfully improve the accuracy of classification, and show that by fusing together classifiers based on images as well as key mobile sensing modalities like acceleration, camera state, and audio, the resulting fused classifier is substantially more accurate than the basic image-only face-centric classifier. Moreover, we seek to demonstrate that by integrating multiple consecutive predictions within a session by our image-based multi-sensor classifier, accuracy can be further improved.

Besides accuracy, another key focus of this research is to improve the efficiency of classification, namely reducing the running time of the fused classifier. This is particularly helpful for mobile devices that...
are comparatively resource-constrained.

5.3.1 Enhanced Image-based Classification

Based on the observations above, we propose an enhanced image-based classifier, which improves upon the CR algorithm in two ways: 1) incorporates new features to detect images that are normal but do not contain any humans; and 2) improves the accuracy of previous facial feature detector to be more applicable on mobile video chat data. In particular, we incorporate the following five features:

**Face size and proportion:** Because of the short distance between user and the mobile camera, faces in mobile video chat tend to be much larger, occupying most of the screen space. So, we filter out all faces whose sizes are less than 1/6 of the image.

**Pair of eyes:** In low-quality mobile video chat images, it is difficult to detect a single eye. So, we focus on detecting only a pair of eyes that are close to each other and located in the same horizon.

**Skin proportion:** Skin proportion is a good feature to separate images containing humans from pure background content. We convert images to the YCbCr color space, which has been shown to be robust against large variations in lighting conditions and effective in skin detection [15].

**Number and distribution of SIFT points:** Scale-Invariant Feature Transform (SIFT) is well-known for object detection and was previously used for flasher detection [20]. Our analysis shows that images that do not contain humans tend to have either very few or a very large number of SIFT points and these points are typically scattered randomly. On the other hand, images that contain humans seem to have a medium number of SIFT points and those points are mostly concentrated around the facial area. In our study, we use the standard deviations of SIFT points’ \(x\) and \(y\) positions to capture the distribution of SIFT points in an image.

**Color histogram distribution:** Images with only background content have very simple color hue, and generate sharp peaks and long tails in their color histograms. This pattern can also be used to differentiate between images containing human from images containing only simple background. In our experiment, we use sixteen bins to measure R/G/B color histogram and calculate the standard deviations for the histogram distributions.
5.3.2 Sensor-based Classification

Mobile devices are equipped with a variety of sensors, such as accelerometer and gyroscope. In addition, newer smartphones are equipped with two cameras, both front and back. These sensors can offer some useful contextual information about user behavior during a video chat. The question is which sensors and how they can be leveraged for flasher detection. In our study, our investigation only focuses on the three-axis accelerometer data, which are available across all smartphone platforms. And the reasons we only use accelerometer not gyroscope come as follows:

- From gyroscope data, our preliminary user behavior study has already learned that in mobile video chat applications, users try their best to hold the phone still for a better video quality. This results in very limited vibrations.
- In an earlier study (Chapter 3.3.4), it is also shown that since the mobile client is deployed in a landscape layout, users mostly hold their phones horizontally and occasionally rotate around x-axis to adjust orientation. Most of the time, phones stay in the stable status.
- A number of low-end Android devices are not equipped with gyroscope.
- Acceleration together with gyroscope are used for precisely monitoring phone orientation in complicated rotation motions, which require careful calibration in advance. These are impractical and unnecessary in our application.

Our preliminary analysis (Chapter 3.3.4) indicates that normal or misbehaving users’ video chat content is highly correlated with the position that a mobile camera focuses on. And this information can be partially estimated by phone orientation along with active camera position (front or back). Besides, during a chat session, normal users tend to keep their phones stable, while flashers’ phones have more slight vibrations.

In this work, for each snapshot image, we collect acceleration data within the two seconds before and two seconds after when the snapshot is captured. After applying a smoothing function to this four-second accelerometer data, we calculate the mean and standard deviation values along the three axes. These values
are combined with the active camera (front or back) information to represent phone orientation and vibration during video chat.

### 5.3.3 Audio-based Classification

Besides image-based and sensor-based features, we also investigate the potential of using audio data to classify normal vs. misbehaving users in mobile video chat. We first labeled our audio data using six different categories: 1) Deep Breath; 2) Music; 3) TV; 4) Quiet; 5) Talk; 6) Others (ambient noise with unrecognized audio). Figure 5.4 shows the number of normal and misbehaving users in the six different audio categories. We see that at least users who “Talk” are usually normal users while “Quiet” users are more likely to be misbehaving.

![Figure 5.4: Number of normal vs. misbehaving users in different audio categories.](image)

A lot of research has been done to predict audio categories by analyzing audio signals and has achieved very promising results [37] [40]. Our audio class prediction algorithm is based on these earlier works and consists of the following four steps.
(1) **Framing**: segments each 10-second audio clip into non-overlapping 64-ms frames.

(2) **Intra-frame Feature Extraction**: extracts the following features from each audio frame:

- (i) **Root Mean Square (RMS)**: captures the overall energy of a frame.
- (ii) **Spectral Entropy**: indicates the frequency pattern of the audio frame. A high entropy resulting from flat spectrum strongly suggests silent audio.
- (iii) **Zero Crossing Rate (ZCR)**: measures sign change rate of a signal, which is effective in speech recognition and music information retrieval.
- (iv) **Bandwidth**: ambient sound typically has a small bandwidth and music consists of a wider mixture of frequencies.
- (v) **First 13 Mel-frequency Cepstral Coefficient (MFCCs)**: is a better approximation for human auditory system and has been proved to be effective to identify finer-grained audio categories.

(3) **Inter-frame Feature Extraction**: considers \( n \) consecutive frames as a frame window. We average the features extracted from the individual frames of a frame window and calculate the standard deviation to measure feature changes among the frames.

(4) **Audio Category Prediction**: feeds the features into a J48 classifier to make a prediction for each frame window. The audio class that receives the majority votes among multiple frame windows is picked as the final prediction for an audio clip. The only exception is that, once an audio frame is determined to be in the “Talk” category, the whole audio clip is assigned to the “Talk” category.

5.3.4 **Session-based Classification**

Our analysis also indicates that people tend to behave consistently during a video chat session and seldom switch between normal and flashing behaviors. Motivated by this observation, we propose a session-based flasher detection mechanism that leverages the temporal modality and takes as input the classification
results of multiple image snapshots and their corresponding sensor readings to generate a more reliable normal vs. misbehaving user prediction for a whole session.

Our session-based classification algorithm works as follows. For each snapshot image and its corresponding sensor data, our image-based multi-sensor classifier gives a binary prediction (Normal vs. Misbehaving) along with a confidence value. A value in the range of $[0, 0.5)$ (or $(0.5, 1]$) indicates misbehaving (or normal), and the lower (or higher) the value, the higher the likelihood that the user is misbehaving (or normal). We apply a 6-bin discretization on the binary prediction and the confidence values, specifically,

- $strong_{normal} : normal + conf \in [0.75, 1]
- medium_{normal} : normal + conf \in (0.65, 0.75)
- weak_{normal} : normal + conf \in (0.5, 0.65)
- weak_{mis} : misbehaving + conf \in (0.4, 0.5)
- medium_{mis} : misbehaving + conf \in (0.25, 0.4)
- strong_{mis} : misbehaving + conf \in [0, 0.25]$

Then, for each session, we calculate the number of occurrences in each bin. We also measure the min, max, mean and standard deviation of all the prediction confidence values in a session. All these features are fed into a Naive Bayes classifier to generate the final prediction for each session.

### 5.3.5 Cascaded Fusion Classification

Given multiple features, one straightforward classification approach is to simply combine all features together. However, this is wasteful, since not all features are necessary when classifying a specific instance. For example, a user can be classified as normal with high confidence when a face is detected. Motivated by this observation, we propose cascaded classification. As illustrated in Figure 5.5, a cascaded classifier consists of a sequence of classifiers ordered by certain criteria (objective function $F_{obj}$) such as average acquisition time or accuracy. Samples that need to be classified pass through the classifiers in stages. At the $i$-th stage, a new feature $f_i$ is extracted (acquisition time $t_i$) and fed into classifier $C_i$ (alone or with previously extracted features) for classification. If the classification confidence of $C_i$ on a sample is above
the confidence threshold $\sigma_i$, the classification process stops and the decision made by classifier $C_i$ on the sample is accepted as the sample’s final classification. Otherwise, the sample is passed to the next stage classifier $C_{i+1}$ for further processing. Since samples classified with high confidence at earlier stages do not need to go through later stages, the overall classification time can be reduced, while still ensuring high classification accuracy. The challenge lies at the design of the objective function, which determines the ordering of the classifiers to be used in each stage.

In general, since the number of candidate feature sets is large, it is difficult to search all permutations to find the global optimal choice for $F_{obj}$. Instead, some local optimal algorithm is adopted (e.g., [20]).

![Figure 5.5: $K$-stage cascaded classifier.](image)

5.3.5.1 Image-based Multi-sensor Cascading

Given each image snapshot and its corresponding sensor data, our image-based multi-sensor classifier fuses together multi-modal features in order to make a classification of normal vs. misbehaving users. Specifically, we aim to build a seven-stage cascaded classifier corresponding to the seven features (face, eye
pair, skin proportion, SIFT number and distribution, color histogram, phone orientation, and audio category). Using cascaded fusion, the key is to utilize at earlier stages features that are efficient to compute and have high classification confidence, thereby reducing/avoiding more complex feature computations at later stages. The compact set of features and the relatively small number of permutations (!) of the seven features make it possible for the global optimization. To balance between classification accuracy and efficiency requirements, we design the following cascaded fusion classifier objective function:

\[ F_{\text{obj}} = T_{\text{save}} + \alpha \cdot P_K \]  

\[ T_{\text{save}} = \frac{T_K \cdot N_K}{N_0} \]  

\[ T_i = (T_{i-1} - t_i) \cdot \frac{N_{i-1}}{N_i} \quad i \in [0, K - 1] \]  

\[ T_0 = \sum_{i=1}^{K} t_i \]  

\[ \gamma = \frac{T_0}{T_0 - T_{\text{save}}} \]  

Here, \( P_K \) is the overall classification accuracy after the \( K \)-th stage. \( N_{i-1}(i \in [1, K]) \) is the number of samples passed into the \( i \)-th stage for classification. Note that \( N_K \) equals \( N_{K-1} \) since the last stage classifies all remaining samples. \( t_i \) is the acquisition cost for feature \( f_i \). Assuming at the beginning of the cascaded classification process, each of the \( N_0 \) samples is allowed \( T_0 \) amount of time for classification, i.e., going through all \( K \) stages. As more samples are filtered out and classified at earlier stages, the total amount of unused time \( (T_{i-1} - t_i) \cdot N_{i-1} \) is divided over the remaining \( N_i \) samples, which is less than or equal to \( N_{i-1} \). Therefore, \( T_{\text{save}} \) measures the amount of time saved per original sample, and \( \gamma \) measures the speedup ratio compared with the full-fusion process with all features. Finally, parameter \( \alpha \) controls the importance ratio between the acquisition cost requirement (average execution time) and the accuracy requirement.

Finally, there are mainly two differences between building FGC classifier in [20] and cascade classifier in our work: (i) in FGC papers, because of the huge number of candidate features, they seek for local optimal at each stage to build cascade classifier. While in our work, since we only have at most 7 potential features, we directly pursue the global optimal classifier by experimenting on all possibilities and only focus on the performance(objective function) at the final stage; (ii) also in FGC papers, the rejected condition is
the prediction of a sample at that stage is Normal while in our work, samples are rejected if the confidence of its prediction is lower than confidence threshold at that stage.

5.3.5.2 Session-based Cascading

Given the temporal ordering of data in a video chat session, the sequence of snapshot images and their corresponding sensor data to be used for classification is fixed in a session. Thus the key issue for session-based cascading is determining the actual number of cascade stages to include in order to achieve a good balance between classification accuracy and efficiency. In other words, our goal is to explore the impact (or tradeoff) of different numbers of cascade stages on the overall session-based classification performance.

5.4 Evaluation

In this section, using real-world data collected from our mobile video chat system, we first evaluate classifier performance on individual features. We show that by fusing together features obtained from image, audio, and multiple sensors, we can significantly improve the classification accuracy, compared with the baseline CR algorithm, which is state-of-the-art for online video chat systems. Also, we show that session-based classification further improves the classification accuracy by leveraging the temporal modality containing multiple classification results within a session. Finally, we measure the execution time of different features, and correspondingly the tradeoff between accuracy and efficiency using image-based multi-sensor cascading and session-based cascading methods.

Let \( R_n \) and \( R_m \) be the sets of normal and misbehaving users identified by a given classifier, respectively. And let \( I_n \) and \( I_m \) be the sets of true normal and true misbehaving users. We consider the following five different classification quality metrics:
\[
\text{Accuracy} = \frac{|R_n \cap I_n| + |R_m \cap I_m|}{|R_n| + |R_m|}
\]

\[
\text{Normal Precision} = \frac{|R_n \cap I_n|}{|R_n|}
\]

\[
\text{Normal Recall} = \frac{|R_n \cap I_n|}{|I_n|}
\]

\[
\text{Misbehaving Precision} = \frac{|R_m \cap I_m|}{|R_m|}
\]

\[
\text{Misbehaving Recall} = \frac{|R_m \cap I_m|}{|I_m|}
\]

### 5.4.1 Dataset

Also according to a previous study on mobile video chat (Chapter 3.3), most video chat sessions are short because users keep requesting the next random user pairing until they have found someone interested to chat for a longer session. In our analysis, we focus on these “meaningful” sessions whose durations are more than 90 seconds, which allow us to collect at least 4 snapshot images and 3 audio clips for each session and are sufficient for session-based classification evaluation. From our dataset, we obtain nearly 350 labeled misbehaving sessions and 1450 labeled normal sessions. The ratio is approximately 1 to 4, which is consistent with the finding in Chapter 3.3. To deal with this skewed distribution and avoid over-training for the normal category, we then pick a balanced dataset containing 348 misbehaving sessions and 357 normal sessions. For all these sessions, we only consider the first 90 seconds. And the four snapshot images contained in each 90-second session are split into four subsets. In all image-based classifier evaluation, the four subsets of data are evaluated separately using ten-fold cross validation and the average is reported as the overall performance.

When labeling images as normal or misbehaving, we follow a procedure that is similar to the one used in Chapter 3.3. Misbehaving users were identified as displaying naked lower bodies for males and naked lower and/or upper bodies for females. Two people labeled the same data set. If an image received conflicting labels, then the two labelers would meet to resolve the conflict.

Audio labeling was also performed by two people with a similar procedure. Apart from our test audio
dataset collected from the mobile video chat clients, we also captured a 10-minute training dataset for each of the six predefined audio categories described earlier. After experimenting with different frame sizes, we found that when window size $n = 16$, our predictor provides the best performance. This results in each frame window to be approximately 1 second in length ($16 \times 0.064 = 1.024\text{s}$).

### 5.4.2 Image-based Classifier Performance

We begin by examining the best classification performance that could be achieved using image-only features. The baseline classifier we use is the CR algorithm, which is a face-centric, image-only algorithm that is currently used by Chatroulette, and is considered state-of-the-art for flasher detection in online video chat systems. In our enhanced image-based classifier, we consider five features (number of faces, existence of eye pair, color histogram statistics, SIFT feature vectors, and skin proportion). We report the classification performance using each individual features, as well as the fused features using Random Forest.

Table 5.1: Classification Quality Comparison of Different Image Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Accuracy</th>
<th>Normal Precision</th>
<th>Normal Recall</th>
<th>Misbehaving Precision</th>
<th>Misbehaving Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR Algorithm (baseline)</td>
<td>0.630</td>
<td>0.694</td>
<td>0.518</td>
<td>0.608</td>
<td>0.765</td>
</tr>
<tr>
<td>Face+Histogram+Skin+SIFT+Eye</td>
<td>0.689</td>
<td>0.713</td>
<td>0.646</td>
<td>0.669</td>
<td>0.733</td>
</tr>
<tr>
<td>Face+Histogram+Skin+SIFT</td>
<td>0.680</td>
<td>0.700</td>
<td>0.644</td>
<td>0.663</td>
<td>0.717</td>
</tr>
<tr>
<td>Face+Histogram+Skin</td>
<td>0.665</td>
<td>0.681</td>
<td>0.637</td>
<td>0.651</td>
<td>0.693</td>
</tr>
<tr>
<td>Face+Histogram</td>
<td>0.633</td>
<td>0.644</td>
<td>0.616</td>
<td>0.623</td>
<td>0.651</td>
</tr>
<tr>
<td>Face</td>
<td>0.588</td>
<td>0.962</td>
<td>0.195</td>
<td>0.546</td>
<td>0.992</td>
</tr>
<tr>
<td>Eye</td>
<td>0.520</td>
<td>0.558</td>
<td>0.249</td>
<td>0.509</td>
<td>0.798</td>
</tr>
<tr>
<td>Histogram</td>
<td>0.594</td>
<td>0.600</td>
<td>0.591</td>
<td>0.587</td>
<td>0.596</td>
</tr>
<tr>
<td>Skin</td>
<td>0.565</td>
<td>0.570</td>
<td>0.573</td>
<td>0.560</td>
<td>0.557</td>
</tr>
<tr>
<td>SIFT</td>
<td>0.568</td>
<td>0.574</td>
<td>0.574</td>
<td>0.562</td>
<td>0.563</td>
</tr>
</tbody>
</table>

Table 5.1 summarizes the classification performance when different image-based features are used. In particular, we find that all proposed features are important factors in improving the accuracy of the image-based classifier. Compared with the 0.630 accuracy achieved by the baseline CR algorithm, our enhanced

---

1 We evaluated different fusion techniques and Random Forest achieves the highest accuracy.
image-based classifier, which combines all five features (face, eye, histogram, skin, SIFT), achieved an improved accuracy of 0.689.

### 5.4.3 Audio Category Classifier Performance

![Audio category prediction performance](image)

**Figure 5.6: Audio category prediction performance**

**Table 5.2: Confusion Matrix of Audio Category Prediction**

<table>
<thead>
<tr>
<th></th>
<th>Deep Breath</th>
<th>Music</th>
<th>Others</th>
<th>Quiet</th>
<th>TV</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Breath</td>
<td>52</td>
<td>6</td>
<td>131</td>
<td>2</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>Music</td>
<td>2</td>
<td>35</td>
<td>5</td>
<td>0</td>
<td>19</td>
<td>12</td>
</tr>
<tr>
<td>Others</td>
<td>67</td>
<td>27</td>
<td>1193</td>
<td>104</td>
<td>49</td>
<td>93</td>
</tr>
<tr>
<td>Quiet</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>122</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TV</td>
<td>6</td>
<td>61</td>
<td>55</td>
<td>4</td>
<td>200</td>
<td>73</td>
</tr>
<tr>
<td>Talk</td>
<td>10</td>
<td>14</td>
<td>60</td>
<td>2</td>
<td>40</td>
<td>344</td>
</tr>
</tbody>
</table>
Figure 5.6 shows the performance of our audio classifier. Overall, our audio classifier achieves 70% accuracy. The Quiet category achieves 95% accuracy; the Talk category achieves 73% accuracy and the Others category is correctly identified 78% of the time. The Deep Breath, Music and TV category predictors perform poorly, correctly identifying only 24%, 48% and 50% of the time respectively. And Table 5.2 further reveals the confusion matrix of our audio classifier, i.e., the number of audio instances in each category that are (mis-)classified into other audio categories. And we see that Deep Breath is often misclassified as Others due to ambient noise in the background, and the TV category is likely to be misclassified as Talk, Music or Others since TV audio could contain variants of these types of sound as well.

5.4.4 Multi-sensor Fusion Classifier Performance

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Normal Precision</th>
<th>Normal Recall</th>
<th>Misbehaving Precision</th>
<th>Misbehaving Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhanced Image</td>
<td>0.689</td>
<td>0.713</td>
<td>0.646</td>
<td>0.669</td>
<td>0.773</td>
</tr>
<tr>
<td>Audio</td>
<td>0.606</td>
<td>0.675</td>
<td>0.431</td>
<td>0.574</td>
<td>0.787</td>
</tr>
<tr>
<td>Acc. + Camera Position</td>
<td>0.769</td>
<td>0.764</td>
<td>0.787</td>
<td>0.775</td>
<td>0.750</td>
</tr>
<tr>
<td>Acc. + Camera Position + Enhanced Image</td>
<td>0.804</td>
<td>0.810</td>
<td>0.807</td>
<td>0.803</td>
<td>0.805</td>
</tr>
<tr>
<td>Acc. + Camera Position + Enhanced Image + Audio</td>
<td>0.820</td>
<td>0.822</td>
<td>0.821</td>
<td>0.817</td>
<td>0.818</td>
</tr>
</tbody>
</table>

A major goal of this paper is to understand whether and to what extent multi-modality mobile sensor data can help improve the flasher detection performance, compared with previous face-centric image-only classification. Here, we evaluate the classification quality using multiple sensors. In particular, we examine the following three modalities:

1. **image**: the enhanced image-only classifier developed earlier that combines face, eye, skin proportion, sift and histogram distributions

2. **orientation**: the phone orientation-related features processed from 3-axis accelerometers and camera position (front/back)

3. **audio**: the predicted major audio category as well as a vector containing the predicted occurrences of each audio category.
Table 5.3 shows the classification quality of different sensor modalities as well as the fused results using Random Forest. We find that the mobile accelerometer and camera sensors used by the Orientation classifier result in a strong gain in accuracy to 0.769 compared to our previous best enhanced image-only classifier of 0.689. Moreover, we find that when fusing the orientation and enhanced image classifiers together, we can achieve an additional gain in accuracy to 0.804. Finally, when we fuse all three modalities (mobile sensor + enhanced image + audio), we observe a final overall accuracy gain to 0.820. In addition, we see that normal and misbehavior precision and recall values are all improved up to 0.82 as we fuse together more sensing modalities. This demonstrates that combining contextual information from multiple sensing modalities substantially improves the overall classification performance of flasher detection on mobile video chat data.

Table 5.4: Muti-sensor fusion classifier performance for different fusion techniques

<table>
<thead>
<tr>
<th>Performance</th>
<th>Accuracy</th>
<th>Normal Precision</th>
<th>Normal Recall</th>
<th>Misbehavior Precision</th>
<th>Misbehavior Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muti-sensor + J48</td>
<td>0.769</td>
<td>0.782</td>
<td>0.752</td>
<td>0.756</td>
<td>0.785</td>
</tr>
<tr>
<td>Muti-sensor + RandomForest</td>
<td>0.820</td>
<td>0.822</td>
<td>0.821</td>
<td>0.817</td>
<td>0.818</td>
</tr>
<tr>
<td>Muti-sensor + AdaBoost</td>
<td>0.748</td>
<td>0.745</td>
<td>0.762</td>
<td>0.751</td>
<td>0.732</td>
</tr>
<tr>
<td>Muti-sensor + Bagging</td>
<td>0.798</td>
<td>0.797</td>
<td>0.805</td>
<td>0.798</td>
<td>0.790</td>
</tr>
<tr>
<td>Muti-sensor + NaiveBayes</td>
<td>0.734</td>
<td>0.808</td>
<td>0.624</td>
<td>0.687</td>
<td>0.846</td>
</tr>
</tbody>
</table>

We also evaluate our multi-sensor fusion classifier’s performance over different fusion algorithms. We choose to compare five different fusion algorithms (J48 Decision Tree, Random Forest, AdaBoost, Bootstrap Aggregating (Bagging) and Naive Bayes) on our dataset, using default parameter settings from the Weka toolkit. As shown in Table 5.4, other fusion algorithms’ quality values are mostly below 0.80 and/or unbalanced between precision and recall, while the Random Forest achieves the highest accuracy of 0.820 and balanced results over precision and recall for normal and misbehaving users.

5.4.5 Session-based Classifier Performance

Here, we evaluate the classification quality of our session-based classifier. We consider the first $x(x = 1, 2, 3, 4)$ image predictions in each session, and different policies to combine the prediction results.
The policies include Major Voting, Normal (Misbehaving) Dominated which predicts the user to be Normal (Misbehaving) when at least one image prediction is Normal (Misbehaving), average confidence, and Naive Bayes (used by our proposed session-based method).

The results are summarized in Table 5.5. When only the first image prediction is used, an accuracy of 0.815 is achieved. Using more image predictions generally improves the classification accuracy. The most gain is achieved with Naive Bayes (our approach) and all 4 image predictions, which resulted in an accuracy of 0.859. We also find policies such as Normal/Misbehavior Dominated perform worse with more image predictions, since they use partial result (ignore confidence) and focus on local information.

Table 5.5: Session-based Classifier Quality Comparison

<table>
<thead>
<tr>
<th>First x Image Predictions Used</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major Voting</td>
<td>0.815</td>
<td>0.823</td>
<td>0.850</td>
<td>0.848</td>
</tr>
<tr>
<td>Normal Dominated</td>
<td>0.815</td>
<td>0.823</td>
<td>0.812</td>
<td>0.807</td>
</tr>
<tr>
<td>Misbehavior Dominated</td>
<td>0.815</td>
<td>0.817</td>
<td>0.803</td>
<td>0.780</td>
</tr>
<tr>
<td>Average Confidence</td>
<td>0.815</td>
<td>0.838</td>
<td>0.855</td>
<td>0.845</td>
</tr>
<tr>
<td>Naive Bayes (our approach)</td>
<td>0.815</td>
<td>0.843</td>
<td>0.854</td>
<td>0.859</td>
</tr>
</tbody>
</table>

5.4.6 Classifier Efficiency on Mobile Devices

Flasher detection is a computation intensive task. For example, using the CR algorithm, Chatroulette needed 30–40 servers running 24/7 to identify misbehaving users in a timely fashion. For mobile video chat, one option is sending all sensor data back to central servers for classification, but that incurs significant network traffic and delay as well as extra workload on the central servers. One key question that we want to address here is whether our multi-sensor misbehavior detection method is feasible for mobile devices.

We have implemented all our sensor feature extraction functionalities using the C language on Android phones and used JNI to call them [51, 4, 5]. During our experiments, we ran our mobile video chat application and maintained an active video chat session in the foreground to emulate a practical execution scenario. In the background, we executed all feature extraction operations immediately once sensor data is
captured by our video chat application to mimic real-world conditions. We also built a lightweight resource measurement Android service that runs continuously in the background to monitor phone resource usage. We conducted experiments on two different types of mobile phones: 1) **HTC One**: an advanced quad-core 1.7 GHz Android phone with 2 GB of memory; and 2) **Galaxy Nexus**: a medium range dual-core 1.2 GHz Android phone with 1 GB of memory, representative of many phones with similar capabilities in the market nowadays.

Figure 5.7 shows the average CPU utilization of our fusion classifier on different phones. The average CPU utilization is defined as:

\[ U_{\text{aver}} = \frac{\sum_{i=1}^{N} U_i}{N} \]  

(5.7)

\( N \) indicates the CPU number, for HTC One \( N = 4 \) and for Galaxy Nexus \( N = 2 \). \( U_i \) measures the \( i \)th CPU utilization.

The vertical blue lines in the figures indicates the start point of a new video chat session. We can see the average CPU utilizations of both devices have a great boost once a video chat session started. The gaps between every two red lines represent the CPU performance of extracting image-based features while video chat running and the periods between each two green lines display the CPU performance of processing audio features during an active video chat session. From the figures we observe:

- Both image and audio feature extraction will generate a peak of average CPU utilization
- HTC One can easily supports our fusion classifier running on it
- For Galaxy Nexus, video chat application has already added lots of burden on CPU usages. Adding extra feature extraction operations nearly runs out of the mobile resource which is likely to freeze ongoing video chat and influence the user experience. However during our experiment, we don’t meet noticeable impact on video quality during the feature extraction operations.
Moreover, Table 5.6 shows the average acquisition time for extracting different features on mobile devices. Note that the feature acquisition time dominates the overall classification time since classifiers are pre-trained and takes minimum time to execute while features need to be extracted at runtime. We notice that face detection is the most computationally intensive task, followed by audio and eye feature extraction. On dual core Galaxy Nexus device, the audio and eye feature are nearly as costly as face detection, whereas they are less than half as costly each on the quad-core Android phone. On the other hand, extracting features such as acc. plus camera state, skin proportion and histogram have only negligible impact on the overall running time on both dual and quad core phones. Overall, the fused classifier takes almost three times longer to run on a dual-core over a quad-core phone. The most important finding here is that our classifier runs reasonably efficiently on both phones (2–6 seconds). And since we could not detect any noticeable impact on the quality of the video chat service that runs concurrently with the classifier, this proves it is feasible to run our classifier on the phone itself, thus taking off significant burden from the servers. We also measure the energy usage of our fusion classifier. Compared with the large battery drain cost by the mobile video chat application, the energy used by our classifier is negligible.
Table 5.6: Acquisition Time Comparison of Different Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>HTC One</th>
<th>Galaxy Nexus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face</td>
<td>1.014s</td>
<td>2.373s</td>
</tr>
<tr>
<td>Eye</td>
<td>0.404s</td>
<td>1.665s</td>
</tr>
<tr>
<td>SIFT</td>
<td>0.226s</td>
<td>0.335s</td>
</tr>
<tr>
<td>Skin</td>
<td>0.033s</td>
<td>0.040s</td>
</tr>
<tr>
<td>Histogram</td>
<td>0.032s</td>
<td>0.040s</td>
</tr>
<tr>
<td>Audio</td>
<td>0.469s</td>
<td>1.673s</td>
</tr>
<tr>
<td>Acc. + Camera State</td>
<td>0.003s</td>
<td>0.002s</td>
</tr>
<tr>
<td>Total</td>
<td>2.181s</td>
<td>6.128s</td>
</tr>
</tbody>
</table>

5.4.7 Cascaded Classifier Performance

Given the execution time measurements obtained above for individual features, we can now evaluate to what extent cascaded fusion can help to further reduce the classifier’s running time while maintaining certain accuracy requirements. We conduct this evaluation for both cascaded fusion scenarios:

- Image-based multi-sensor cascading which is based on a single snapshot image and its corresponding sensor data
- Session-based cascading which utilizes multiple image-based multi-sensor predictions in a session.

5.4.7.1 Image-based Multi-sensor Cascading Performance

The experiments about the classifier performance on mobile devices have indicated that our fusion algorithms on low-end phones such as Galaxy Nexus have a risk to run out of mobile resources and cause a temporary freeze on mobile video chat application. To reduce the occurrence of this situation and at the same time maintains an reasonable classification capacity, we apply a cascade classifier onto the seven effective sensor features discovered in the earlier section, namely face, eyes, skin, sift, color histogram, Acc plus front/back camera position and audio. Our multistage image-based cascade classifier will predict samples
with strong patterns (patterns with high confidence during the training step) at early stage, prevent extracting features and prediction at later steps to reduce prediction time as well as save resource usage.

However, different permutations of the cascaded stages will result in different running time and overall accuracy. Certain permutations may result in high-complexity classifiers being placed in later stages so that they are executed less often, hence reducing the average run time. Our goal is to find the ordering of the stages that maximizes execution speedup, overall accuracy, or some combination of both factors. So in each stage, the classifier will extract one more feature and combine it with all features obtained in previous stages to make a prediction. Samples resulting in a higher prediction confidence than the confidence threshold for this stage are filtered out (along with the corresponding classification results), and the rest of the samples are passed on to the next stage for further classification. Since there is only a reasonable set (7!) of candidate features, it is feasible to explore all possible permutations of cascaded classifiers and only optimize the final stage’s objective function to pursue the globally optimal choice.

Given the measured feature acquisition time \( t_i \ (i \in [1, K]) \) shown in Table 5.6, we can calculate \( T_0 \), which is the total amount of time needed per sample to go through all \( K \) stages. In our remaining evaluations, we use the running time measured on HTC One and \( T_0 = 2.181s \) as the all-stage execution time per sample.

Our first experiment explores the influence of the confidence threshold \( \sigma \) on the performance of the cascaded classifier. We are especially interested in two special cases for optimizing the objective function:

- the case where we seek an ordering of the cascade that maximizes efficiency or time saved (\( \alpha = 0 \)), which we refer to as “best average saving time” \( T_{\text{save}}(BAST) \)

- the case where we seek a cascade ordering that maximizes accuracy (\( \alpha = \infty \)), which we refer to as “best final accuracy” \( P(BFA) \)
For all our experiments, we use the same confidence threshold for every stage. And as mentioned before, Weka generates a bidirectional confidence distribution for binary classification. For example, Figure 5.8 indicates that misbehaving prediction has confidence between 0 and 0.5 and the lower the value, the more confident the prediction; while normal prediction has confidence values between 0.5 and 1 and the higher the value, the more confident the prediction. Because of this, we need two distinct confidence thresholds $\sigma_1$ and $\sigma_2$ for misbehaving and normal user classification respectively. Since the bidirectional confidence distribution is nearly symmetric, in our study, for simplicity we define a new factor named “confidence cutting threshold” $\rho$ which is derived from the confidence threshold $\sigma$:

$$\rho = |\sigma - 0.5|$$ (5.8)

Then $\rho$ can somehow represent the value for both $\sigma_1$ for misbehaving users and $\sigma_2$ for normal users such that when $\rho = 0.3$, $\sigma_1 = 0.2$ and $\sigma_2 = 0.8$. 

Figure 5.8: Bidirectional confidence distribution generated by Weka for binary classification.
Figure 5.9 shows the results of our first experiment. From the figure, we make several important observations:

- For the BAST criterion, increasing the confidence cutting threshold for each stage of the cascaded classifier results in an optimal cascaded classifier with lower saving time but higher accuracy. This is because a high confidence threshold adds a high stopping requirement at each stage, which makes the overall cascaded classifier more conservative and causes more samples to proceed deeper into the cascade’s stages. While for the BFA criterion, there is the same trend found for speed up ratio but the accuracy goes down when confidence cutting threshold is very high ($\rho = 0.4$ and $\rho = 0.45$). This indicates for some samples, too many indicators might make negative contributions for the predictions.

- For the BAST criterion, with a reasonable confidence cutting threshold ($\rho = 0.25$ or $\rho = 0.3$), the optimal cascaded classifier can achieve a final accuracy of about 0.77 and meanwhile reduce average running time by a factor of 6.3. Table 5.7’s columns 2 and 3 illustrate the optimal ordering of the stages and the flow of samples through this cascaded classifier, where again $N_{i-1}$ means the number of samples passed into the $i$-th stage, and each run starts with a balanced data set of 2820 total samples from 705 sessions that are then processed through the cascaded classifier.

- For the BFA criterion, with the same threshold ($\rho = 0.25$ or $\rho = 0.3$), the optimal cascaded
classifier can maintain even better performance (0.83 accuracy) than our fusion classifier while achieving a modest 4.4 time speedup in execution time. Table 5.7 columns 4 and 5 show the optimal ordering of stages to maximize accuracy, and the sample flow through the cascaded classifier.

Table 5.7: Structure and Performance of Optimal Cascaded Classifier for BAST and BFA when $\rho = 0.75$

<table>
<thead>
<tr>
<th>Stage $i$</th>
<th>BAST</th>
<th>BFA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>New Features $N_i$</td>
<td>New Features $N_i$</td>
</tr>
<tr>
<td>1</td>
<td>Skin 1810</td>
<td>Acc+Cam 1203</td>
</tr>
<tr>
<td>2</td>
<td>Acc+Cam 793</td>
<td>Sift 1203</td>
</tr>
<tr>
<td>3</td>
<td>Histogram 568</td>
<td>Skin 848</td>
</tr>
<tr>
<td>4</td>
<td>Audio 422</td>
<td>Histogram 651</td>
</tr>
<tr>
<td>5</td>
<td>Eye 345</td>
<td>Face 520</td>
</tr>
<tr>
<td>6</td>
<td>Face 297</td>
<td>Audio 393</td>
</tr>
<tr>
<td>7</td>
<td>Sift 0</td>
<td>Eye 0</td>
</tr>
</tbody>
</table>

Our second experiment seeks to understand the tradeoff between efficient and accurate classifications for a wider range of values $\alpha$ of the defined objective function $F_{\text{obj}}$, not just the two extrema of BAST and BFA. We let $\alpha$ range across the values $\{0 \text{ (BAST)}, 0.001, 0.01, 0.1, 1 \text{ (equal weight), } 10, 100, 1000, \infty \text{ (BFA)}\}$. 
Figure 5.10: Optimal cascaded classifier performance on different control balance and confidence threshold.

Figure 5.10 shows the tradeoff function between speedup ratio and accuracy for the optimal cascaded classifier across a range of $\alpha$, and also shows different curves that correspond to different confidence cutting thresholds $\rho$. Based on this figure, we make the following observations:

- Each function ($\rho$ constant) exhibits a similar shape wherein the low $\alpha$ values are clustered together on the upper left, followed by an inflection point farther down and right that represents the balance point $\alpha = 1$, followed by a clustering of the high $\alpha$ values even further down and to the right.

- Overall, execution time speedups range from a factor of 1.25 to 13.36 times while the final accuracy correspondingly decreases from 0.829 to 0.732.

- Similar to our previous observation for just the two extrema, fixing $\alpha$ across our large range still results in the trend wherein a lower confidence threshold $\rho$ generates an optimal cascaded classifier with a higher factor of time savings and generally a lower final accuracy only except when $\rho$ is pretty high (0.95).

Figure 5.10 helps to quantify the general tradeoff between efficiency and accuracy for our cascaded classifier on different requirements. If we wish to achieve a certain efficiency speedup target, we can pre-
cisely assess how much the accuracy will be sacrificed. Whereas, if we wish to boost the accuracy to a given level (higher $\rho$), then we will be able to determine the amount of reduction in the speedup factor of the execution time. For example, if we desire to push the edge on speed, then we can achieve an approximately 13X speed gain, with nearly the same accuracy at about 0.819. If our goal is to push the edge on accuracy, then we can obtain a better accuracy of 0.831 - the best that even the non-cascaded multi-sensor fusion classifier wasn’t able to achieve - at the cost of earning only a 3X speed gain.

5.4.7.2 Session-based Cascading Performance

For our session-based classifier, features are derived from the prediction results generated by our image-based multi-sensor fusion classifier. Since the image sequence is fixed for a session, a full stage (4 stages in our study) session-based cascaded classifier have only one possible ordering. So in our evaluation, we focus on: 1) what performance our full stage cascaded classifier could achieve; 2) how accuracy and efficiency change when the cascade starts at later predictions namely a partial stage cascaded classifier.

Table 5.8: Session-based Cascaded Classifier Performance

<table>
<thead>
<tr>
<th>Cascade start image id</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.848</td>
<td>0.854</td>
<td>0.856</td>
<td>0.859</td>
</tr>
<tr>
<td>Aver Image Used for Prediction</td>
<td>1.29</td>
<td>2.17</td>
<td>3.11</td>
<td>4</td>
</tr>
<tr>
<td>Speedup Ratio</td>
<td>3.10</td>
<td>1.84</td>
<td>1.29</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.8 indicates with a full stage cascaded session-based classifier, we can achieve more than 3X speed gain at the cost of only 1.1% accuracy reduction. Also it indicates by waiting a little longer to start the cascaded prediction, classification can reach nearly the same accuracy as non-cascaded classifier at the cost of executing a very small amount of image predictions. For example, if we decide to start the cascaded classifier after the second image-based prediction is generated, the overall session-based classifier can reach 0.854 accuracy by only requiring 2.17 image predictions which could achieve more than 2X speed gain.
5.4.7.3 Multi-level Cascading Performance

Finally, we combine our image-based multi-sensor fusion cascaded classifier with session-based cascaded classifier together, building a two-level cascaded misbehavior classification. In image-based fusion cascaded classifier, we choose $\rho = 0.3$ and $\alpha = 1$ and the classifier achieves $13.15X$ speed gain with accuracy equal to 0.819. In session-based cascaded classifier, the cascade classifier chooses to start once the second prediction is generated. In total, our multi-level cascaded classifier reaches 0.843 accuracy while taking only 0.408 seconds for a session prediction which achieves a significantly 21X speed up compared with non-cascaded session-based classification (which takes $2.181 \times 4 = 8.724$ seconds).

5.5 Conclusions

In this chapter, we present a multi-modal fusion framework for accurate and efficient misbehavior detection in a mobile random video chat application. First, we show that traditional face-centric image-based classification developed for online video chat users achieves only 0.63 accuracy when applied to a real world data set of normal and misbehaving mobile video chat users. We further show that our enhanced image classifier could improve overall accuracy to 0.69. Then, we demonstrate that an image-based multi-sensor fusion classifier that integrates mobile accelerometer data along with front/back mobile camera position and audio category can substantially improve the overall accuracy to 0.82. Next, we explore the temporal modality and by leveraging four image-based predictions within a session, our session-based classifier achieves a further improvement to 0.86 accuracy. Additionally, we demonstrate the feasibility of running the fused multi-modal misbehavior classifier on mobile devices. And finally, we design and evaluate a multi-level cascaded classifier to quantify the tradeoff between efficiency and accuracy. We show that with a certain configuration, our classifier could achieve 21X speedup gain with a reasonable 0.84 accuracy.
Chapter 6

Conclusions & Future Work

In this thesis, I focus on studying user behavior on a new emerging realm, the mobile random video chat service. Because of the great population gained by online random video chat services and the rapid developments of mobile hardware capability and cellular network connectivity, we expect mobile random video chat services will increase significantly in volume and frequency in the future.

To conduct a first-ever detailed study of mobile random video chat user behavior at scale, we design and deploy an Android-based, Omegle compliant, mobile random video chat system, named MVChat. The MVChat system helps us collect hundreds of gigabytes of behavioral data from thousands of users. Using the collected data, we analyze user behavior from multiple aspects and observe several interesting behavioral patterns. For example, most of sessions are short as “pure” random pairing is not effective for matching users’ preferences; text message is not frequently used in mobile video chat; the presence of misbehaving problem has also propagated onto mobile platform and mobility introduces more diversity than web online content. Next, through our taxonomy analysis, we learn that normal users are highly correlated with front camera and showing their faces, whereas misbehaving users tend to hide their faces. Additionally, we learn that surprisingly females are just as likely to misbehave as males. Finally, a brief study of acceleration data reveals that mobile sensor could be leveraged to better detect misbehavior on mobile platform.

Then we analyze meaningful user behavior in depth to figure out the key characteristic of effectiveness in promoting longer video chat sessions. The study first demonstrates that people behave consistently within video chat sessions. Based on that observation, we further prove that mobile users even behave consistently among sessions regardless of meaningful or not. Finally, our study reveals females are highly popular, in
that users with a large enough fraction of sustained sessions are disproportionately female. And showing
face with the front camera could contribute to promote longer sessions.

To relieve the negative impact caused by the presence of obscene content and improve user satisfac-
tion of mobile random video chat services, we present a multi-modal fusion classifier for mobile flasher
detection. The classifier improves the previous image-based face-centric classifiers by integrating multi-
dimensional sensors such as acceleration, camera position and audio. We also leverage the temporal modal-
ity to further improve classification accuracy. Last but not least, we apply a multi-level cascaded classifier
to quantify the tradeoff between efficiency and accuracy.

However, the insights gained from our meaningful user behavior analysis are straightforward and su-
perficial for guiding users to behave more effectively in mobile random video chat. Even though users follow
these insights, the “pure” random pairing mechanism is still ineffective for matching all users’ preferences.
To overcome this great challenge, it is necessary to build an effective matching algorithm for random chat
services. To achieve this, researchers need to build their own random video chat systems, which have a full
control of the pairing mechanism. This provides a way to collect the paring information from the both sides
of each conversation and to access the picture of entire user set. Meanwhile, the abundant sensor data from
the smartphones, such as audio, snapshot, camera state, acceleration and gyroscope, can be investigated to
extract the user profiles and preferences. For example, the content of video and audio plays a very impor-
tant role in maintaining long sessions. For long sessions, those data indicate the matched interests shared
between the two sides of a conversation. Applying the advanced audio and image recognition algorithms
could help to learn the preference of users. Finally, the duration of a chat conversation has been proved to
be a good indicator to measure the quality of this random pair. In the study of building effective matching
algorithms, it could be used to evaluate the effectiveness of a user participating in the chatting and measure
the performances of proposed matching algorithms. Exploring content from these aspects could greatly
help researchers to build an effective online randomized matching algorithm for mobile random video chat
services.


[23] Jessie Duan, Sean Scott, and Yongxing Deng. Predicting edge properties in a bipartite network.


[27] Goggle captcha.


[64] Xinyu Xing, Yu-li Liang, Sui Huang, Hanqiang Cheng, Richard Han, Qin Lv, Xue Liu, Shivakant Mishra, and Yi Zhu. Scalable misbehavior detection in online video chat services. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’12, pages 552–560, New York, NY, USA, 2012. ACM.


