Privacy-Preserving Photo Taking and Accessing for Mobile Phones

Ang Li
University of Arkansas, Fayetteville

Follow this and additional works at: https://scholarworks.uark.edu/etd

Part of the Information Security Commons

Citation

This Dissertation is brought to you for free and open access by ScholarWorks@UARK. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of ScholarWorks@UARK. For more information, please contact scholar@uark.edu.
Privacy-Preserving Photo Taking and Accessing for Mobile Phones

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Computer Science

by

Ang Li
Henan University
Bachelor of Science in Computer Science, 2010
Peking University
Master of Engineering in Software Engineering, 2013
University of Arkansas
Master of Science in Computer Science, 2017

August 2018
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council

Qinghua Li, Ph.D.
Dissertation Director

Xintao Wu, Ph.D.                 Susan Gauch, Ph.D.
Committee Member                 Committee Member

Jingxian Wu, Ph.D.
Committee Member
ABSTRACT

Today, we are living in environments that are full of cameras embedded in devices such as smart phones and wearables. These mobile devices and as well as apps installed on them are designed to be extremely convenient for users to take, store and share photos. In spite of the convenience brought by ubiquitous cameras, users’ privacy may be breached through photos that are taken and stored with mobile devices. For example, when users take a photo of a scenery, a building or a target person, a stranger may also be unintentionally captured in the photo. Such photos expose the location and activity of strangers, and hence may breach their privacy. In addition, photos that are stored on smartphones may contain private information (e.g., driver’s license) about phone owners, which raise peoples privacy concerns over unauthorized access by installed apps.

The goal of this dissertation is to protect people’s privacy in photo taking and accessing. To achieve this goal, we propose several systems to address the aforementioned privacy issues.

To protect stranger’s privacy in photo taking, we proposed two systems called PrivacyCamera and PoliteCamera. Through cooperation between the photographer and the stranger, these systems can automatically blur the stranger’s face in the photo upon the stranger’s request when the photo is being taken. Even though PrivacyCamera and PoliteCamera can successfully protect stranger’s privacy, they depend on the cooperation between the photographer and the stranger. That requires both the photographer and stranger to install the proposed systems on their mobile phones; however, this is not always possible. Therefore, we further propose a feature-based model to automatically distinguish the target from strangers in a photo, so that we can blur all strangers' faces without the cooperation. Finally, we designed PhotoSafer, a content-based and context-aware to protect private photos from unauthorized access on Android phones.
In future work, we plan to design a privacy-preserving online sharing system, which has less burden of policy settings and can protect the privacy of strangers in a photo. In addition, we will also consider designing personalized systems to protect user-specific private photos.
ACKNOWLEDGMENTS

I would like to thank all the people who have helped me during my Ph.D. study.

First and foremost, I would like to thank my advisor Dr. Qinghua Li. It has been an honor to be his first Ph.D. student. I greatly appreciate all his contributions of time, idea, and funding to make my Ph.D. study productive. His enthusiasm for his research is motivational for me, even during tough times in my Ph.D. pursuit.

My gratitude also goes to other members of my dissertation and advisory committee, including Prof. Xintao Wu, Prof. Susan Gauch, Prof. Jingxian Wu and Prof. Dale R. Thompson. Their insightful comments for my dissertation have significantly helped me improve it.

In addition the committee members, I would like to thank my co-authors Wei Du, Michael Mahler and David Darling. I have been very lucky to collaborate with these great people on different research projects. Thanks are also given to Dr. John Gauch who provided insightful suggestions on my research.

Finally, I express my deepest gratitude to my beloved family for their continued support throughout my educational career, as well as all of my friends, who have made the experience much more enjoyable.
# TABLE OF CONTENTS

1 Introduction .......................................................... 1
  1.1 Overview of This Dissertation ..................................... 2
    1.1.1 Privacy-Preserving Photo Taking for Mobile Phones .......... 2
    1.1.2 Privacy-Preserving Photo Accessing for Mobile Phones ....... 4
  1.2 Organization ...................................................... 4
Bibliography ............................................................ 5

2 PrivacyCamera: Cooperative Privacy-Aware Photographing with Mobile Phones 7
  2.1 Introduction ...................................................... 7
  2.2 System Design .................................................... 9
    2.2.1 System Overview ............................................ 9
    2.2.2 The Architecture and Workflow of PrivacyCamera .......... 11
    2.2.3 Stranger Determination .................................... 12
    2.2.4 Face Blurring ............................................. 13
    2.2.5 Analysis on the Effectiveness of Protection ............... 16
  2.3 Implementation ................................................. 19
    2.3.1 Face Detection ............................................. 19
    2.3.2 Blurring Request Collection ................................ 20
    2.3.3 Stranger Determination .................................... 20
  2.4 Evaluations .................................................... 21
    2.4.1 Experimental Methodology .................................. 21
    2.4.2 GPS Accuracy Test .......................................... 22
    2.4.3 Face Detection Test ........................................ 22
    2.4.4 Accuracy of Protection ...................................... 23
    2.4.5 Cost Evaluation ............................................ 28
  2.5 Related Work .................................................... 30
  2.6 Summary .......................................................... 31
Bibliography ............................................................ 31

3 PoliteCamera: Respecting Strangers’ Privacy in Mobile Photographing .... 35
  3.1 Introduction ...................................................... 35
  3.2 System Design .................................................... 39
    3.2.1 System Overview ............................................ 39
    3.2.2 The Architecture and Workflow of PoliteCamera .......... 40
    3.2.3 Facial Attribute Classifier ................................ 41
    3.2.4 Stranger Determination .................................... 44
    3.2.5 Target Filter .............................................. 45
    3.2.6 Face Blurring ............................................. 47
  3.3 Implementation ................................................... 47
    3.3.1 Face Detection and Preprocessing ........................... 47
    3.3.2 Facial Attribute Classifier ................................ 48
  3.4 Evaluations ....................................................... 49
6 Conclusions and Future Work .................................................. 102
  6.1 Conclusions ........................................................................ 102
  6.2 Future Work ....................................................................... 103
Bibliography ............................................................................ 104
A Appendix ................................................................................. 106
  A.1 IRB Approval .................................................................... 106
LIST OF TABLES

Table 2.1: True Protection Rate for Scenario 1 When $d = 5$ Meters ........................................ 25
Table 2.2: True Protection Rate for Scenario 1 When $d = 10$ Meters ........................................ 25
Table 2.3: False Protection Rate for Scenario 1 When $d = 5$ Meters .......................................... 26
Table 2.4: False Protection Rate for Scenario 1 When $d = 10$ Meters ......................................... 27
Table 2.5: Running Time of Blurring Faces .................................................................................. 28
Table 2.6: Power Consumption When Running in the Background .............................................. 29
Table 2.7: Power Consumption When Running in the Foreground ............................................... 30
Table 3.1: Facial Attributes Classification of Fig. 3.6(a) and Fig. 3.6(b). ........................................ 53
Table 3.2: Classification consistency of 10 persons with 2 pairs of face images each. .................. 53
Table 3.3: Effectiveness of stranger determination under different threshold with 50 tests. ........ 54
Table 3.4: True Protection Rate ................................................................................................. 57
Table 3.5: False Protection Rate ............................................................................................... 58
Table 4.1: The Range of Each Feature ....................................................................................... 70
Table 4.2: The Comparison of Classification Accuracy .............................................................. 73
Table 4.3: The Evaluation of Classification Accuracy Based on Different Feature Sets .......... 73
Table 4.4: The Comparison of False Filtering Rate ....................................................................... 74
Table 5.1: Mobile Phone Platform Usage of Survey Respondents ............................................. 81
Table 5.2: Age Group Distribution of Survey Respondents ......................................................... 81
Table 5.3: Photo Types and the Proportion of Respondents That Consider Them as Private ........ 82
Table 5.4: Number of Installed Apps That Can Access Photos .................................................. 82
Table 5.5: MobileNets with Different Width Multipliers .............................................................. 89
Table 5.6: MobileNets with Different Resolution Multipliers ...................................................... 89
Table 5.7: Dataset Distribution .................................................................................................. 90
Table 5.8: Comparison of Classification Accuracy and Model Size ........................................... 92
Table 5.9: Classification Accuracy for Each Type of Photo ......................................................... 93
Table 5.10: Confusion Matrix of Private Photo Classification ..................................................... 93
Table 5.11: Time Cost for Obtaining Photo Content Classification Result: From Database vs. Real-Time Classification ................................................................. 93
LIST OF FIGURES

Figure 2.1: Privacy issues with photos taken by mobile phones (photo by author). 8
Figure 2.2: The architecture of PrivacyCamera. 12
Figure 2.3: Detecting if a stranger is in field of view of a camera. 13
Figure 2.4: An example process of Gaussian Blur for blurring the central pixel. 15
Figure 2.5: Effect of Gaussian Blur with different radiuses (photo by author). 16
Figure 2.6: Two special cases where $\delta=0^\circ$ and $r=5$ meters. 17
Figure 2.7: True protection rate when the distance from the stranger to the camera is 0 meters, 5 meters, and 10 meters. X-axis is the relative angle $\delta$. 19
Figure 2.8: Experiment scene (photo by author). 22
Figure 2.9: GPS accuracy cumulative distribution. 23
Figure 2.10: Face detection under different lighting conditions (photo by author). 23
Figure 2.11: Field test (photo by author). 27

Figure 3.1: Privacy issues with photos taken by mobile phones (photo by author). 35
Figure 3.2: The architecture of PoliteCamera. 41
Figure 3.3: Architecture of the ABCNN network. 48
Figure 3.4: Average classification accuracy vs training epochs. 50
Figure 3.5: Classification accuracy of each attribute and average accuracy in testing. 51
Figure 3.6: Two different face images from the same person [17]. 52
Figure 3.7: Target filter test on photos taken by the authors (photo by author). 55
Figure 3.8: Target filter test on photos shared by friends on Facebook (used with permission). 56
Figure 3.9: Example Experiment Scenes (photo by author). 56
Figure 3.10: Example of a successful protection (photo by author). 57

Figure 4.1: The example of blurriness measurement in a photo [11]. 68
Figure 4.2: Orientation of the face in terms of yaw. 68
Figure 4.3: Orientation of the face in terms of pitch. 69
Figure 4.4: Orientation of the face in terms of roll. 69
Figure 4.5: Example photos of different settings in terms of inclusion of the target and the stranger. 71
Figure 4.6: Example photos of different settings in terms of the target’s face position [11]. 72

Figure 5.1: The workflow of PhotoSafer. 85
Figure 5.2: The architecture of PhotoSafer. 86
Figure 5.3: Example photos in our dataset. Sensitive information are removed from the photos. 91
List of Published Papers

1. **Chapter 2**: Ang Li, Qinghua Li, and Wei Gao, “PrivacyCamera: Privacy-Aware Photographing with Mobile Phones,” *IEEE International Conference on Sensing, Communication and Networking (SECON)*, 2016.

2. **Chapter 3**: Ang Li and Qinghua Li, “PoliteCamera: Respecting Strangers’ Privacy in Mobile Photography”, *EAI International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2018.

1 Introduction

Today, we are living in environments that are full of cameras embedded in mobile and wearable devices such as smartphones, tablets, and wearable glasses. These devices are designed to be extremely convenient for people to capture their daily life, and store those photos on such devices. However, the development of ethical guidelines and normative standards of conduct has not caught up with the rapid technological innovation. The prevalent cameras raise people’s privacy concerns that they may easily enter a physical space where they can be captured without their consent or awareness. For example, when a user takes a photo of a beautiful view or a friend using his mobile phone, frequently a stranger is also accidentally included in the photo, with the face clearly recognizable. Such photos might be further uploaded to and spread through online social networks, leading to privacy violations for the captured strangers. Although recent works [1, 2, 3, 4, 5] have revealed that photographers are willing to protect stranger’s privacy, they lack effective tools to achieve this goal. Indeed, concerns about individual privacy is one of the biggest barriers for the wide adoption of wearable cameras such as Google Glass.

Another focal concern is that more and more people prefer to store photos on their mobile phones. These photos may contain some privacy-sensitive (e.g., driver’s license) information and can be accessed by third-party apps without their consent and awareness. Even though mobile platforms such as Android provide permission-level access control over photos, it is still a coarse-grained scheme that either grants the access to all photos or denies the access to all of them. In order to protect private photos on mobile phones, some techniques [6, 7, 8] have been developed to control access to those photos. However, existing solutions either cannot provide real protection or induce high operational complexities for
To address above challenges, we also need to consider that mobile devices are resource-constrained. Due to the limited computing and energy resources, the systems need to be designed with consideration of computation cost and energy consumption. In addition, communication and storage overhead should be kept low whenever possible.

1.1 Overview of This Dissertation

The goal of this dissertation is to design mobile systems to protect strangers’ privacy in photos and protect private photos from unauthorized access by apps. For this purpose, we propose several mobile systems.

1.1.1 Privacy-Preserving Photo Taking for Mobile Phones

To protect the privacy of strangers accidentally included in photos taken by mobile phones, we adopt a cooperative privacy protection approach. At the time of taking a photo, the photographer (via his mobile phone) can automatically notify nearby strangers of the possible inclusion in the photo via peer-to-peer short-range wireless communications (e.g., Wifi Direct [9]). If a stranger does not want to appear in the photo, he can send a request to the photographer. The photographer will determine whether the requesting stranger is in the photo or not. If so, the photographer will blur the stranger’s face in the photo.

The challenge is how to determine whether a stranger who requests privacy protection is in the photo or not. There might be multiple nearby strangers who receive the notification of potential privacy leakage by the photo. Some of them may request to blur their faces but others may not request so; some of them are captured in the photo but others are not in the photo. Hence, it is not trivial to determine if the requesting stranger’s face is in the photo or not.
We design three systems to address this challenge. The first system is *PrivacyCamera* which determines whether a requesting stranger is in the photo or not based on GPS locations. Specifically, when a stranger requests face blurring, he sends his GPS location to the photographer. The photographer can then check whether the stranger is in the camera’s field of view based on their relative locations. However, the effectiveness of PrivacyCamera is limited by the accuracy of GPS locations and it does not work well indoor due to the unavailability of GPS. To address this limitation, we design another system named *PoliteCamera* which uses facial attributes that do not change frequently (e.g., black hair or blond hair) to determine whether a stranger is in a photo or not. In particular, a stranger sends his facial attributes to the photographer in the blurring request. By comparing the stranger’s facial attributes with the facial attributes of the faces in the photo, the photographer can determine whether the stranger is in the photo. In addition, we propose heuristic rules to tell the target of a photo from strangers in the photo, in case the target’s photo also issues a blurring request.

PrivacyCamera and PoliteCamera rely on the cooperation between the photographer and the strangers during photo taking. However, such cooperations are not always possible. For example, the communication channels between the photographer and the stranger may not be reliable. If the communication channel is disconnected, the blurring request cannot return to the photographer and the system will fail. To address this limitation, we adopt another approach where the photographer simply blurs all strangers’ faces. Here the challenge is to tell the target of a photo from the strangers. To address this challenge, we design a feature-based model to distinguish the target from the strangers. Specifically, we propose a set of features, and build a binary classifier based on the features to distinguish target from strangers. We implement the model based on different supervised learning algorithms and explore their performances. Moreover, we explore how feature selections affect
the effectiveness of the approach.

1.1.2 Privacy-Preserving Photo Accessing for Mobile Phones

To protect private photos from unauthorized access while offering friendly user experience, we design a novel content-based and context-aware private photo protection system named PhotoSafer for mobile phones. It provides real-time access control over private photos based on the photo contents and the contextual status of accesses, and discloses the specific sensitive content that a private photo contains to users before that photo can be accessed.

Our design principle is that a photo should be accessed by an app with users’ awareness and users should be allowed to decide whether the app can access that photo. A naive approach is to prompt users to check the photo and make a decision every time. However, this will definitely degrade the usability of the system and apps. To address this problem, PhotoSafer is designed to be able to automatically check whether the content of photo is private, and determine whether the user is aware of the app’s access request based on the contextual status of the phone and the app, e.g., whether the phone is locked or not, and whether the app is in the foreground or not. To minimize the time needed to identify private photo content during user operation, PhotoSafer identifies the contents of photos in advance and caches the results in a database for real-time query. In this way, it can achieve real-time response to photo access, such that the requesting app’s usability and user experience will not be affected.

1.2 Organization

The reminder of the dissertation is organized as follows. Chapter 2 presents our cooperative privacy protection system PrivacyCamera. Chapter 3 introduces our facial attributes based system PoliteCamera for protecting strangers’ privacy in mobile photographing.
ter 4 describes our feature-based model for distinguishing the target from the strangers in photos. Chapter 5 elaborates our content-based and context-aware private photo protection system PhotoSafer, which provides real-time access control over private photos based on the photo contents and the contextual status of mobile phone, along with the visibility of requesting apps. Chapter 6 concludes this dissertation and discusses future work.

Bibliography


2 PrivacyCamera: Cooperative Privacy-Aware Photographing with Mobile Phones

2.1 Introduction

Mobile phones are usually embedded with powerful cameras today [1]. For example, iPhone 6 has an 8-megapixel camera. As mobile cameras in the pocket, mobile phones are increasingly used by people to take photos anywhere and anytime. However, there exist some privacy issues associated with this convenience. When a user takes a photo of a beautiful view or a friend using his mobile phone, frequently a stranger is also accidentally included in the photo, with the face clearly recognizable. Figure 2.1 shows two examples. In Figure 2.1(a), the photographer intends to take a photo of the building but a stranger appears; in Figure 2.1(b), the photographer intends to picture the target person but a stranger is also included. In these cases, the photo can reveal the stranger’s location and even activity. For strangers who do not want to appear in the photo and get their location revealed, being accidentally included in a photo breaches privacy. Thus, this problem should be addressed.

With the development of image processing technology, there exist several softwares which can blur faces in a photo, such as Adobe Photoshop and ObscuraCam [2]. However, none of these commercial softwares can make the stranger in the photo know that he is included in the photo and give him the right to decide whether to blur his face or not. These solutions only allow the photographer to make decisions as to blurring the stranger’s face or not.

A naive solution for protecting stranger’s privacy is that each user stores a pool of familiar faces (e.g., self, family members and friends) in the phone and the phone simply blurs all other faces in the photo. However, this solution may cause unnecessary blurring.
Some strangers may not care about whether they are included in the photo or not. Blurring their faces is not needed and can unnecessarily degrade the quality of the photo.

In this chapter, we propose a mobile cooperative privacy protection system, called *PrivacyCamera* [3], to protect the privacy of a stranger who is accidentally included in a photo taken by mobile phones. PrivacyCamera can work as an App on both the photographer’s and the stranger’s mobile phone. At the time of taking a photo, it can automatically notify nearby strangers of the possible inclusion in the photo via peer-to-peer short-range wireless communications (e.g., Wifi Direct [4]). If a stranger does not want to appear in the photo, he can send a request to the photographer. The photographer will determine if the requesting stranger is in the photo or not. If so, the photographer will blur the stranger’s face in the photo.

The contribution of this chapter is summarized as follows:

- To the best of our knowledge, PrivacyCamera is the first mobile system which can notify nearby strangers of the possible inclusion in a photo when the photo is being taken, give them an option to opt out, and blur a stranger’s face upon his request.
• We design a location-based stranger determination scheme to determine if a stranger is in the photo or not based on his relative location to the photographer and the heading direction of the camera, and theoretically analyze its effectiveness.

• We design a Gaussian Blur based face blurring scheme that can smoothly blur a stranger’s face with minimal negative effect on the quality of a photo.

• We implement a prototype system on Nexus 5 phones, and evaluate the system’s performance and cost using experiments and field tests.

The rest of this chapter is organized as follows. Section 2.2 presents the design of PrivacyCamera and theoretical analysis of its performance. Section 2.3 describes the prototype implementation. Section 2.4 shows evaluation results. Section 2.5 reviews related work. Section 2.6 concludes this chapter.

2.2 System Design

This section describes the design of PrivacyCamera and analyzes its performance.

2.2.1 System Overview

Three types of entities are involved in the system: the photographer who takes photos using a mobile phone, the target that the photographer intends to picture, and the stranger who is near the photographer and might be unintentionally included in the photo. The target can be a building, a natural scenery, a person, etc. The system is proposed for outdoor usage.

The system aims to protect the stranger’s privacy through providing a method for the stranger to opt out from the photo. Our basic idea is that the photographer notifies nearby strangers of the possible inclusion in a photo at the time of taking the photo, and blur a stranger’s face in the photo upon the stranger’s request. Note that the system does not intend
to simply blur every stranger’s face in the photo. This is because blurring inevitably affects the quality of the photo, even though our design adopts an advanced blurring technique to minimize such effect. To minimize the quality degradation brought to the photo, the system only blurs a stranger’s face if he requests.

To make the system work, both the photographer and the stranger are required to install PrivacyCamera (in the form of an App) on their mobile phone. PrivacyCamera relies on the cooperation between photographers and strangers to protect privacy. Since each mobile phone user can sometimes be a photographer and sometimes a stranger, PrivacyCamera users essentially cooperatively protect each other’s privacy. It is worthwhile to note that the paradigm of inter-user cooperation has been successfully adopted in many real-world systems such as peer-to-peer file downloading systems [5] and online recommender systems [6]. This success has also motivated our system design. The more users adopt this system, the better privacy can be protected.

As the first work in this direction, this chapter starts with considering two relatively simple scenarios:

- **Scenario 1:** The target of a photo is not a person but something else such as a building. One stranger is accidentally included in the photo, and he may or may not want his face to be blurred.

- **Scenario 2:** The target is a person. One stranger accidentally appears in the photo, and he may or may not want to blur his face.

Based on our observations, these two scenarios represent a significant portion of photographing cases although not all of them. Thus, our scheme can enhance privacy in many real-world scenarios. We will address more complex scenarios in future work.

Even under these two relatively simple scenarios, the problem is still challenging.
First, there might be multiple strangers nearby who can receive the notification of possible inclusion in the photo. Some of them may request their faces to be blurred but others may not request so. Although only one stranger is included in the photo in the two scenarios, we still need to determine if the stranger in the photo is requesting for face blurring or not, which is not easy. Second, in Scenario 2, if the stranger in the photo requires face blurring, we need to make sure that the stranger’s face, not the target’s, is blurred.

For simplicity, photographer is also used to denote the photographer’s mobile phone when the context is clear. The same applies to stranger.

2.2.2 The Architecture and Workflow of PrivacyCamera

As Figure 2.2 shows, the system consists of four major modules: face detection, blurring request collection, stranger determination and face blurring. When a photographer takes a photo, the face detection module will run on the captured image. If no face is detected, no further processing is needed. If any face is detected, the blurring request collection module sends notifications to nearby strangers using peer-to-peer short-range wireless communications. If a stranger receiving the notification does not want to be included in the photo, he sends a blurring request to the photographer. Since this stranger may not necessarily be included in the photo, to help the photographer determine if this stranger is the one in the photo, this stranger puts his location (i.e., GPS coordinates) in the request. Then, the stranger determination module of the photographer will check if the requesting stranger is in the photo or not based on their relative location and the heading direction of the camera. If the stranger is in the photo, the face blurring module of the photographer will smoothly blur his face; otherwise, the request is ignored.

The design of PrivacyCamera is based on several technologies available in off-the-shelf mobile phones. Face detection can be done using APIs provided by mobile phones,
e.g., the FaceDetector [7] APIs in Android SDK. Peer-to-Peer communications between the photographer and nearby strangers can be supported by short-range wireless technologies such as WiFi Direct [4] and Bluetooth which are available on most mobile phones today, e.g., Nexus 5. We will introduce how these two modules can be implemented in our prototype system in Section 2.3. Next, we describe how to determine if a stranger is in the photo and how to blur faces.

2.2.3 Stranger Determination

This module aims to detect if a stranger who requests face blurring is included in the photo or not. This is done through checking if the stranger is in the field of view of the photographer’s camera or not.

This process is illustrated in Figure 2.3. First, we determine the camera’s heading direction using the orientation sensor embedded in mobile phones. Note that the heading direction read from compass is in degrees east of Magnetic North instead of True North (there is a declination angle between the two), and it should be converted in degrees east of True North (i.e., $\beta$ in the figure) so as to be in the same coordinate system with GPS coordinates. Then, we obtain the stranger’s relative direction to the photographer (i.e., $\alpha$ in the figure) using the GPS coordinates of the stranger and the photographer. Next, we calculate the relative angle from the stranger to the camera (denoted by $\delta$) as $\delta = |\beta - \alpha|$.
Lastly, we determine if the stranger is in the field of view of the camera or not. The horizontal view angle of the camera (denoted by $\gamma$) which specifies the effective horizontal scope of the camera can be obtained using the API of mobile OS (e.g., GetHorizontalViewAngle() on Android OS). For example, it is 60 under default focal length for the Nexus 5 phone. If $\delta \leq \gamma/2$, the stranger is in the photo; otherwise, he is not in the photo.

![Figure 2.3: Detecting if a stranger is in field of view of a camera.](image)

For Scenario 2, it is not enough to determine that the stranger is in the photo. We also need to tell which face is the target and which is the stranger. To achieve this goal, we adopt a heuristic approach. We observe from real-life experiences that when we take a photo of a target person, we usually intentionally make the target’s face larger than anyone else accidentally included into the photo. For example, if a stranger is too close to the camera and hence his face is larger than the target’s, the photographer will probably change a facing direction or ask the target to move a little so that the target is better captured into the photo than the stranger. Thus in the photo the stranger’s face should be smaller than the target’s. Based on this, the smaller face will be determined as the stranger in Scenario 2.

### 2.2.4 Face Blurring

The goal of face blurring is to mask the identifiable features of a face without reducing the quality of the photo much.

As a preparation step for face blurring, we first need to determine a blurring area
in the face which encloses the main identifiable features of the face. In this section, we use a square area as the blurring area. Specifically, we draw the square by setting the middle point between eyes as the center of the square, and setting the length of a side as 2.4 times of the distance between eyes. Our tests show that the square drawn in this way can cover the main identifiable features of a face with the minimum area (see Figure 2.5(a)). Next, we describe how to blur this square area.

To blur faces smoothly, the Gaussian Blur algorithm [8] is used in this section. The effect of Gaussian Blur is like viewing an image through a translucent screen. The basic idea of Gaussian Blur is to adjust the color value of each target pixel (under the RGB color model) as the weighted average of the color value of itself and other nearby pixels. The weights are calculated based on Gaussian function such that closer pixels have higher weights. Since the pixels closer to the target pixel usually have more similar colors with the target pixel than the pixels farther away, this blurring method can achieve smooth blurring with minimal effect on photo quality.

Given a target pixel to blur, all the pixels that will be used to blur the target pixel are enclosed in a circle with radius $R$ and centered in the target pixel. Let $C$ denote this circle. Let the target pixel be the origin of a two-dimensional coordinate system whose coordinates are $[0, 0]$. Then for pixel $[x, y]$ in the circle, where $x$ and $y$ represent the abscissa and the ordinate of this pixel, its weight is calculated as follows:

$$w(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2.1)$$

where $\sigma$ is the standard deviation of Gaussian distribution. Let $v_{[x,y]}(R,G,B)$ denote the color values of a pixel $[x, y]$. Then the blurred color values of the target pixel is computed
as follows:
\[
v_{[0,0]}(R,G,B) = \sum_{[x,y] \in C} v_{[x,y]}(R,G,B) \frac{w(x,y)}{\sum_{[x,y] \in C} w(x,y)}
\]  

(2.2)

Here, we use a simple example in Figure 2.4(a) to illustrate how a target pixel is blurred. In this example, there are 9 pixels in an image, and the central pixel is the target pixel which is marked in red color. The blur radius is set such that only the target pixel and its direct neighbors are in the circle. The coordinates of the 9 pixels are shown in Figure 2.4(a). The original RGB color values of each pixel is shown in Figure 2.4(b). To conduct Gaussian Blur, we need to calculate the weight for each pixel by Equation 2.1. Suppose \( \sigma \) in Equation 2.1 is 1.5. Then the original weight for each pixel computed by Equation 2.1 is shown in Figure 2.4(c). After that we can calculate the new color values for the target pixel using Equation 2.2, which are shown in Figure 2.4(d).

For the square blurring area, we can blur each pixel in this square from the leftmost pixel to the rightmost pixel in each row and row by row using Gaussian Blur. Given a certain \( \sigma \), the effect of Gaussian Blur is more intense as the blur radius increases. To fully hide the identifiable features in the face, a big enough blur radius should be set. Our tests show that the effect of Gaussian Blur with \( \sigma = 3 \) and \( R = 30 \) is good enough to meet our requirement. Figure 2.5 shows the blur effects under different radius values and fixed \( \sigma = 3 \). It can be seen that when the blur radius is 30 the face cannot be identified.

![Figure 2.4: An example process of Gaussian Blur for blurring the central pixel.](image-url)
2.2.5 Analysis on the Effectiveness of Protection

Suppose a stranger is included in a photo and he requests to blur his face. For the stranger’s face to be really blurred, a precondition is that the stranger determination module successfully detects the stranger as being in the photo based on the stranger’s claimed location measured by GPS. Here, we analyze the true protection rate of PrivacyCamera, which is defined as the percentage of times when the system can successfully detect a stranger as being in the photo given that the stranger is really in the photo.

The true protection rate depends on a few factors: GPS accuracy $r$, the horizontal view angle of the camera $\gamma$, the real distance between the stranger and the camera $d$, and the real relative angle from the stranger to the camera $\delta$. It actually equals to the probability that when the stranger is really in the camera’s field of view, its GPS-measured location is also in the camera’s field of view. If we draw a circle centered at the stranger’s real
location with radius $r$, a true protection happens when the GPS-measured location is within the intersection between the circle and the view of the camera. The probability is equal to the fraction of the circle in the intersection (assuming that it is equally likely for the GPS-measured location to be any point in the circle). Figure 2.6 shows two special cases when the stranger is directly facing the camera (i.e., $\delta = 0$) and the distances are 5 meters and 10 meters.

Since $r$ and $\gamma$ depend on the device, we can consider these two parameters as constants. In fact, we obtained $r$ and $\gamma$ on Nexus 5 phones (see Section 2.4), which are 5 meters and 60°, respectively. For simplicity, we use these values in our analysis. Next we analyze the true protection rate as a function of $d$ and $\delta$.

Generally speaking, given a certain $\delta$, the true protection rate will be higher when $d$ increases, since longer $d$ can better tolerate the inaccuracy of GPS. Here, we want to find out the upper bound and lower bound of the true protection rate. Since the effective range of face detection is 10 meters as shown in Section 2.4.3, the case with $d = 10$ meters is considered as the upper bound. The worst case happens when the stranger and the photographer are at the same location, i.e., $d = 0$. Besides, we also consider the case with $d = 5$ meters as a reference point in the middle, based on our experience that a stranger is more than 5 meters away from the camera in most cases.
The true protection rate for the lower bound case is pretty straightforward to derive. Since $\gamma$ is 60°, the true protection rate equals to the probability that the GPS-measured location is within the view of the camera, which is $\frac{60}{360} = 16.7\%$.

The true protection rate for the other two cases $d = 5$ and $d = 10$ with changing $\delta$ can be deduced using geometry. We omit the detailed process due to the space limitation. For $d = 5$ meters, the true protection rate is given in Equation 2.3; for $d = 10$ meters, the true protection rate is given in Equation 2.4.

\[
P_1 = \frac{1}{3} + \frac{h_1 \times \sqrt{r^2 - h_1^2} + h_2 \times \sqrt{r^2 - h_2^2}}{\pi r^2},
\]

where $h_1 = r \times \sin(30 - \delta)$, $h_2 = r \times \sin(30 + \delta)$ and $\delta \in [0, 30]$.

\[
P_2 = \frac{2 \arccos \frac{h}{r}}{360} + \frac{h \times \sqrt{r^2 - h^2}}{\pi r^2},
\]

where $h = d \times \sin(30 - \delta)$ and $\delta \in [0, 30]$.

Based on these two equations, we can calculate the theoretical true protection rate for any specific relative angle $\delta$ in these two cases. Figure 2.7 shows the numerical results where the x-axis is the relative angle $\delta$. The upper bound achieves to 100% when the stranger is directly facing to the camera (i.e., $\delta = 0$) with the distance of 10 meters. When the stranger and the photographer stand at the same spot, the true protection rate is 16.7%, which is the lower bound. For the case with $d = 5$ meters, the true protection rate decreases from 60% to 46% when the relative angle increases. When the real distance is between 5 meters and 10 meters, the true protection rate is expected to sit between the red dashed line and the green dashed line.
Figure 2.7: True protection rate when the distance from the stranger to the camera is 0 meters, 5 meters, and 10 meters. X-axis is the relative angle $\delta$.

2.3 Implementation

We implemented a prototype system on Nexus 5 phones. The system uses Android 5.1.1 OS and Android 4.3 APIs. This section describes the implementation of major modules. The face blurring module is implemented as described in Section 2.2.4 and thus not described in details here.

2.3.1 Face Detection

The face detection module is implemented based on the `FaceDetector` [7] class provided in Android SDK. Faces in an image can be detected by calling the `findFaces` method of `FaceDetector`. This method detects faces by finding pupils in the image. The `findFaces` method returns a number of detected faces in the image and populate them into an array of `FaceDetector.Faces` class [9].

From each instance of the `FaceDetector.Faces` class, we can obtain the distance between the two eyes of a face in pixels and the coordinates of the middle point between the two eyes. As introduced in Section 2.2.4, the face blurring module uses these information to determine the blurring area for a face.
2.3.2 Blurring Request Collection

This module enables the photographer to send notifications to nearby strangers, and enables each stranger to send a blurring request as well as his location to the photographer. In our prototype, Wifi Direct [4, 10, 11] is used to implement the peer-to-peer communications between the photographer and the strangers. The photographer first discovers nearby peers by calling the `discoverPeers` method of `WifiP2pManager` system service. Then it sends a notification to each peer and collects the blurring request from the peer.

**Location Acquisition** The stranger gets his location using the `LocationManager` [12] service. Depending on the device, several technologies can be applied to determine current location, including GPS and cellular network. In the prototype, we check the availability of GPS and cellular network in turn. Then, we can find current location by calling corresponding method of GPS or cellular network. Finally, the longitude and latitude of current location are obtained.

2.3.3 Stranger Determination

We introduced how to detect if a stranger is in the photo or not in Section 2.2.3. Here, we describe how to obtain the parameters used in that approach (see Figure 2.3) on a phone.

The camera’s heading direction (β in Figure 2.3) can be obtained by reading sensor data from gyroscope embedded in Android phones. However, the returned value is in degrees east of Magnetic North instead of True North. To be in the same coordinate system with GPS coordinates, we convert it to be in degrees east of True North by adding the declination angle between Magnetic North and True North. The declination angel can be obtained by calling the native method in Android APIs.

We can obtain the relative direction from the stranger to the photographer (α in
Figure 2.3) by calling the `bearingTo` method of the `Location` object, passing in the stranger’s current location. Since the original bearing value returned from `bearingTo` is within the range from negative 180° to positive 180°, we normalize it to be within the range from 0° to 360°.

With the camera’s heading direction and the relative direction from the stranger to the photographer, we can easily calculate the relative angle $\delta$ in Figure 2.3 and normalize it to be within 0° and 180°.

The horizontal view angle of the camera $\gamma$ in Figure 2.3 can be obtained by calling the `getHorizontalViewAngle` method of the `Camera.Parameters` object. In our current prototype, only the rear-facing camera without zooming in and zooming out has been considered. However, the value with zooming in and zooming out can be obtained similarly, and our general approach is applicable to those cases as well.

2.4 Evaluations

2.4.1 Experimental Methodology

The experiments are conducted outdoors on our campus and under fine weather. Figure 2.8 shows two typical scenes of experiments. In our experiments, standard GPS instead of assisted GPS (A-GPS) [13] is used for location acquisition since A-GPS has lower accuracy. Before conducting each test, we wait around 30 seconds to make sure that the GPS receiver is able to get the latest location information. Also, each test is done at a different location. In addition, we do not zoom in and zoom out the rear-facing camera, and just use the default focal length.
2.4.2 GPS Accuracy Test

As GPS is used to determine the location of strangers, the accuracy of location obtained from the GPS receiver of a phone is an important factor that determines the performance of the system. The accuracy level of GPS may vary from tens of meters to millimeters [14]. The actual accuracy depends on many factors, such as sky blockage, receiver quality and atmosphere condition [15]. For high-quality consumer-grade GPS receivers, the accuracy can be within 5 meters under the open sky and 10 meters under closed canopies [16].

To examine the accuracy of GPS receivers on mobile phones, we conducted 100 tests at different locations. Each test calls the `getAccuracy` method of the `Location` object in Android to get an approximate accuracy at the current location in meters. The approximate accuracy is defined in the following way: if we draw a circle with the center at the current location and the radius equal to the accuracy, there is a 68% possibility that the true location is inside the circle. The test results are shown in Figure 2.9. The average accuracy is about 5 meters, and in the 66% of the tests the accuracy is no more than 5 meters.

2.4.3 Face Detection Test

This part evaluates the effectiveness of the face detection module in detecting faces in a photo. Considering that the strength of light might affect detection, we conducted tests in
Figure 2.9: GPS accuracy cumulative distribution.

Figure 2.10: Face detection under different lighting conditions (photo by author).

the morning, at noon and in the evening. As shown in Figure 2.10, even under dark lighting conditions, the face detection module can effectively detect the face in photos. Additionally, we changed the locations of the person to be detected within the field of view of the camera. Specifically, since the horizontal view angle of Nexus 5 is 60°, we did tests when the relative angle from the person to the camera (δ in Figure 2.3) is 0°, 10°, 20° and 30°. The results show that faces can be successfully detected when the distance between the person and the camera is within 10 meters at any relative angles, but cannot be detected when the distance is over 11 meters. When the distance is between 10 meters and 11 meters, faces can sometimes be detected.

2.4.4 Accuracy of Protection

This part evaluates the effectiveness of our system in protecting the stranger’s privacy.
**True Protection Rate for Scenario 1**

This group of tests considers Scenario 1 where one stranger appears in the photo. Suppose the stranger wants to blur his face. We evaluate the true protection rate. In our tests, the stranger stands 5 meters and 10 meters away from the camera. The stranger and the photographer are positioned in ways such that the relative angle between the stranger and the camera’s heading direction (δ in Figure 2.3) is 0° (i.e., the camera directly faces the stranger), 15°, and 30°. The camera’s heading direction is randomly set in each test. If the relative angle calculated by the stranger determination module is no more than 30°, the stranger is successfully detected as being in the image and his face is blurred. Figure 2.11(a) shows an example where the stranger’s face is successfully blurred.

Table 2.1 and Table 2.2 show the true protection rates when the distance is 5 meters and 10 meters respectively. In each table, we also show the relative angles calculated by the stranger determination module to provide more information. In both cases, the true protection rate decreases when the relative angle increases, i.e., when the stranger is closer to the edge of the camera’s view. The true protection rate is higher when the distance is 10 meters than when it is 5 meters, because longer distance can better tolerate the inaccuracy of GPS location. Moreover, we can find that these test results are consistent with our theoretical analysis shown in Figure 2.7.

**False Protection Rate For Scenario 1**

Suppose in Scenario 1, the stranger in the photo (denoted by A) does not request to blur his face. However, another nearby stranger B who is not in the photo may submit a blurring request. In this case, we define false protection rate as the percentage of times when the stranger B not in the photo is mistakenly detected as being in the photo and stranger A’s face is hence falsely blurred. In the tests, stranger B and the photographer are positioned
Table 2.1: True Protection Rate for Scenario 1 When $d = 5$ Meters

<table>
<thead>
<tr>
<th>Test</th>
<th>Relative angle to camera</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0°</td>
</tr>
<tr>
<td>Test 1</td>
<td>20°</td>
</tr>
<tr>
<td>Test 2</td>
<td>8°</td>
</tr>
<tr>
<td>Test 3</td>
<td>17°</td>
</tr>
<tr>
<td>Test 4</td>
<td>12°</td>
</tr>
<tr>
<td>Test 5</td>
<td>40°</td>
</tr>
<tr>
<td>Test 6</td>
<td>34°</td>
</tr>
<tr>
<td>Test 7</td>
<td>37°</td>
</tr>
<tr>
<td>Test 8</td>
<td>18°</td>
</tr>
<tr>
<td>Test 9</td>
<td>20°</td>
</tr>
<tr>
<td>Test 10</td>
<td>23°</td>
</tr>
<tr>
<td>True Protection Rate</td>
<td>70%</td>
</tr>
</tbody>
</table>

Table 2.2: True Protection Rate for Scenario 1 When $d = 10$ Meters

<table>
<thead>
<tr>
<th>Test</th>
<th>Relative angle to camera</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0°</td>
</tr>
<tr>
<td>Test 1</td>
<td>23°</td>
</tr>
<tr>
<td>Test 2</td>
<td>3°</td>
</tr>
<tr>
<td>Test 3</td>
<td>8°</td>
</tr>
<tr>
<td>Test 4</td>
<td>7°</td>
</tr>
<tr>
<td>Test 5</td>
<td>5°</td>
</tr>
<tr>
<td>Test 6</td>
<td>0°</td>
</tr>
<tr>
<td>Test 7</td>
<td>22°</td>
</tr>
<tr>
<td>Test 8</td>
<td>21°</td>
</tr>
<tr>
<td>Test 9</td>
<td>19°</td>
</tr>
<tr>
<td>Test 10</td>
<td>31°</td>
</tr>
<tr>
<td>True Protection Rate</td>
<td>90%</td>
</tr>
</tbody>
</table>

in ways such that the relative angle from stranger $B$ to the camera ($\delta$ in Figure 2.3) is $30^\circ$, $60^\circ$, $90^\circ$, $120^\circ$, $150^\circ$ and $180^\circ$. If the relative angle calculated by the stranger determination module is no more than $30^\circ$, stranger $B$ is falsely detected as being in the photo. Table 2.3 and Table 2.4 show the results when the distance between $B$ and the photographer is 5 meters and 10 meters respectively. Similar to the true protection rate case, the relative angles calculated by the stranger determination module are also shown. We can see that the false protection rate decreases when the relative angle increases and when the distance
increases which is reasonable.

We further evaluate the false protection rate in a more noisy environment, where there are five strangers like stranger B in the above test who are not in the photo but submit a blurring request. In this case, we define false protection rate as the percentage of times when anyone of these five strangers not in the photo is mistakenly detected as being in the photo and stranger A’s face is hence falsely blurred. In this group of tests, the distance between strangers and the photographer is between 5 and 10 meters, and the relative angle from strangers to the camera is within 30° to 90°. All the strangers’ locations are randomly selected within these ranges. Over 20 independent tests, the false protection rate is as low as 10%.

Table 2.3: False Protection Rate for Scenario 1 When \( d = 5 \) Meters

<table>
<thead>
<tr>
<th>Test</th>
<th>Relative angle to camera</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>30°</td>
</tr>
<tr>
<td>Test 1</td>
<td>15°</td>
</tr>
<tr>
<td>Test 2</td>
<td>20°</td>
</tr>
<tr>
<td>Test 3</td>
<td>22°</td>
</tr>
<tr>
<td>Test 4</td>
<td>16°</td>
</tr>
<tr>
<td>Test 5</td>
<td>54°</td>
</tr>
<tr>
<td>Test 6</td>
<td>41°</td>
</tr>
<tr>
<td>Test 7</td>
<td>39°</td>
</tr>
<tr>
<td>Test 8</td>
<td>29°</td>
</tr>
<tr>
<td>Test 9</td>
<td>44°</td>
</tr>
<tr>
<td>Test 10</td>
<td>61°</td>
</tr>
<tr>
<td>False Protection Rate</td>
<td>50%</td>
</tr>
</tbody>
</table>

True Protection Rate for Scenario 2

The true protection rate for Scenario 2 depends on two factors. One factor is the true protection rate for Scenario 1, and the other factor is the accuracy of correctly telling the stranger’s face from the target’s face. We first ran tests to evaluate the accuracy of correctly telling the stranger’s face from the target’s. In our tests, the stranger stands farther from
Table 2.4: False Protection Rate for Scenario 1 When $d = 10$ Meters

<table>
<thead>
<tr>
<th>Test</th>
<th>Relative angle to camera</th>
<th>30°</th>
<th>60°</th>
<th>90°</th>
<th>120°</th>
<th>150°</th>
<th>180°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td></td>
<td>29°</td>
<td>71°</td>
<td>110°</td>
<td>146°</td>
<td>127°</td>
<td>169°</td>
</tr>
<tr>
<td>Test 2</td>
<td></td>
<td>2°</td>
<td>83°</td>
<td>113°</td>
<td>135°</td>
<td>144°</td>
<td>171°</td>
</tr>
<tr>
<td>Test 3</td>
<td></td>
<td>44°</td>
<td>82°</td>
<td>118°</td>
<td>157°</td>
<td>142°</td>
<td>178°</td>
</tr>
<tr>
<td>Test 4</td>
<td></td>
<td>38°</td>
<td>75°</td>
<td>99°</td>
<td>126°</td>
<td>138°</td>
<td>174°</td>
</tr>
<tr>
<td>Test 5</td>
<td></td>
<td>35°</td>
<td>41°</td>
<td>102°</td>
<td>137°</td>
<td>129°</td>
<td>159°</td>
</tr>
<tr>
<td>Test 6</td>
<td></td>
<td>21°</td>
<td>65°</td>
<td>99°</td>
<td>140°</td>
<td>136°</td>
<td>167°</td>
</tr>
<tr>
<td>Test 7</td>
<td></td>
<td>46°</td>
<td>82°</td>
<td>97°</td>
<td>128°</td>
<td>131°</td>
<td>192°</td>
</tr>
<tr>
<td>Test 8</td>
<td></td>
<td>25°</td>
<td>56°</td>
<td>98°</td>
<td>156°</td>
<td>168°</td>
<td>203°</td>
</tr>
<tr>
<td>Test 9</td>
<td></td>
<td>29°</td>
<td>68°</td>
<td>86°</td>
<td>117°</td>
<td>152°</td>
<td>163°</td>
</tr>
<tr>
<td>Test 10</td>
<td></td>
<td>17°</td>
<td>76°</td>
<td>83°</td>
<td>105°</td>
<td>149°</td>
<td>168°</td>
</tr>
<tr>
<td>False Protection Rate</td>
<td></td>
<td>40%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

(a) Scenario 1 (b) Scenario 2

Figure 2.11: Field test (photo by author).

the camera than the target as assumed in Section 2.2.3. They do not stand in a line so that both of their faces appear in the photo. To make sure both faces can be detected, we keep them within 10 meters from the camera. Over 20 tests, we found that the system can always successfully tell the stranger’s face. Figure 2.11(b) shows an example where the stranger’s face is successfully blurred without affecting the target person. As a result, the true protection rate in Scenario 2 should be the same as that of Scenario 1.
Table 2.5: Running Time of Blurring Faces

<table>
<thead>
<tr>
<th>Distance</th>
<th>Blur Radius</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>5 meters</td>
<td>4ms</td>
</tr>
<tr>
<td>10 meters</td>
<td>2ms</td>
</tr>
</tbody>
</table>

2.4.5 Cost Evaluation

Communication Delay

This part evaluates the round-trip delay from the time the photographer sends out a notification to the time he receives a request from the stranger. In the tests the stranger stands 5 meters and 10 meters away from the camera, and 10 tests were run for each distance. The average delays are 188ms and 193ms when the distance is 5 meters and 10 meters, respectively. Hence, the communication delay is short.

Running Time of Blurring Faces

This part evaluates the time needed to blur a face on a mobile phone. Two factors affect the time, the distance from the stranger to the camera and the blur radius of Gaussian Blur. The distance has an effect since it affects the size of the blurring area. If the blur radius is larger, more computations are needed for blurring each pixel of the blurring area. In these tests, we set the distance as 5 meters and 10 meters, and set the blur radius as 10, 20 and 30. Table 2.5 shows the results, where each data point is the average of 5 tests. When the distance increases, the running time decreases. This is because longer distance means smaller face in the photo and hence smaller blurring area. When blur radius increases, the time increases due to the higher computation load. In all these cases, the running time of blurring faces is very small.
Power Consumption

To measure the power consumption of our system on the phone, we utilize a widely used App called PowerTutor [17] which can accurately monitor the power consumption of different Apps. We compare the consumption of our system with Google Maps and Chrome.

First, to evaluate our power consumption when no photos are taken, we tested these Apps running in the background for 5 minutes. Table 2.6 shows their average power consumption. It can be seen that PrivacyCamera consumes much lower power than the other two Apps.

<table>
<thead>
<tr>
<th></th>
<th>Google Maps</th>
<th>Chrome</th>
<th>PrivacyCamera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Power (mW)</td>
<td>25</td>
<td>32</td>
<td>10</td>
</tr>
</tbody>
</table>

Then we measure the power consumption of one conversation between the photographer and the stranger (including sending notification and receiving blurring request) and blurring one face on the phone. For comparison, we also measured the power consumption of visiting one web page in Chrome and searching for one location in Google Maps. Table 2.7 shows the results, where each data point is the average result of 10 tests. In our system, each communication conversation only consume 0.12J, which is the lowest among the tested operations. The power consumption of blurring one face is 7.5J. Based on this number, a fully-charged battery (3.8V, 2300 mAh) of Nexus 5 phone can support the blurring of 4195 faces before being depleted. Thus, the power consumption is low. We noticed that the power consumption for blurring one face is higher than visiting one web page and searching one location. However, users usually do not take photos as often as they visit web pages and searching locations. Thus, we expect that the overall power consumption of PrivacyCamera should be lower than Chrome in practice.
Table 2.7: Power Consumption When Running in the Foreground

<table>
<thead>
<tr>
<th>Test Application</th>
<th>Operation</th>
<th>Average Energy Usage(J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Maps</td>
<td>Search for 1 location</td>
<td>2.4</td>
</tr>
<tr>
<td>Chrome</td>
<td>Visit 1 web page</td>
<td>1.4</td>
</tr>
<tr>
<td>PrivacyCamera</td>
<td>Conduct 1 conversation</td>
<td>0.12</td>
</tr>
<tr>
<td>PrivacyCamera</td>
<td>Blur 1 face</td>
<td>7.5</td>
</tr>
</tbody>
</table>

2.5 Related Work

Jung and Philipose [18] propose a method to protect video privacy. The wearable camera will stop recording a person when it detects that the person is making certain gestures, e.g., waving hands. MarkIt [19] detects the sensitive objects predefined by users in videos and covers the sensitive objects with markers before releasing the video to third-party applications. Jana et al. [20] design an OS abstraction Recognizer to enforce fine-grained access control in augmented reality system. It can reduce the quality of raw sensor data when third-party applications request to access it. A similar system is SemaDroid [21]. Jana et al.[22] implement a privacy protection layer to restrict untrusted applications to access input data from perceptual sensors. For example, a person’s face sketch can be transformed depending on different privacy levels.

Schiff et al. [23] propose a system to detect persons that wear special tracking markers and block their faces from photos. However, people must wear special markers beforehand which does not apply to our considered problem. Bo et al. [24] design a protocol to protect the privacy of people being photographed based on people’s privacy desires, which are contained in a physical tag. In their approach, however, people are required to wear clothes with QR-code as privacy tags. Wang et al. [25] propose an approach to protect people’s privacy based on the recognition of their visual fingerprints in images or videos, including motion patterns and visual appearance (e.g., clothing color). However, it requires people to upload their visual fingerprints to the server whenever their visual fingerprints such as clothing are
changed, which needs intensive intervention by people. Moreover, their approach relies on a server to do the detection which does not fit our scenario. Templeman et al. [26] prevent private images from being shared with others based on attributes extracted from the image such as location and content. PlaceAvoider [27] is a system that can notify the photographer when an application is going to capture images in sensitive areas (e.g., bedroom). Pidcock et al. [28] propose a system to notify bystanders of nearby mobile sensing activities. Tan et al.[29] propose a system to protect photo privacy in Android phones. The system can recognize the photos that contain persons known to the phone owner, and denies third-party applications to access these photos. However, none of the above approaches can be applied to protect stranger’s privacy in our scenarios.

2.6 Summary

We designed a system PrivacyCamera to protect strangers’ privacy who are accidentally included in a photo taken by mobile phones. The system can notify nearby strangers of the possible inclusion in a photo and allow them to decide if to blur their faces in the photo. We designed techniques to detect if a stranger requesting face blurring in in the photo or not based on GPS locations. We implemented a prototype system, and evaluated the system’s performance and cost through experiments as well as field tests. Evaluations show that the system can accurately detect the stranger and blur his face to protect his privacy.

Bibliography


3 PoliteCamera: Respecting Strangers’ Privacy in Mobile Photographing

3.1 Introduction

Nowadays mobile phones usually have built-in cameras that facilitate capturing photos. For instance, iPhone 7 is embedded with a 12-megapixel camera [1]. However, an increasing privacy concern has arisen as more and more pictures are taken in people’s daily lives. When a user takes a photo of a scenery or a friend with a mobile phone, it is likely that a stranger can also be accidentally included in the photo, with the face clearly recognizable. Fig. 3.1 illustrates two examples. In Fig. 3.1(a), the building is the target but a stranger is captured; in Fig. 3.1(b), the photographer intends to picture the target person but two strangers are accidentally included. In these examples, the photo can breach the stranger’s privacy by revealing the stranger’s location and activity. Thus strangers privacy should be protected.

![Figure 3.1](image-url): Privacy issues with photos taken by mobile phones (photo by author).

Several recent works have been done to protect strangers privacy in photos through...
blurring their faces. They differ in the way of determining whether a stranger is in the photo or not. Our proposed PrivacyCamera [5] in Chapter 2 checks whether a stranger is in a photo or not based on GPS locations of the photographer and the stranger. Due to the dependence on GPS location, it does not work well indoor due to the unavailability or inaccuracy of GPS. He et al. [6] design a system for protecting photo privacy that identifies a stranger in a photo by recognizing his motion patterns and visual appearance (e.g., clothes color) profiled into the system in advance. However, users’ visual fingerprints need to be updated whenever they change (e.g., changing clothes), which is not convenient. Zhang et al. [7] propose a server-based system to protect privacy of photographed users that compares the portrait of a user uploaded to the server and the portrait of the persons included in photos. Their scheme considers full portrait captured in the photo (i.e., the whole body), which is quite different from this chapter that only considers face. Also, their scheme assumes a trusted server from the privacy perspective, which is not always available.

In this chapter, we use facial attributes that do not change frequently (e.g., black hair or blond hair) to determine whether a stranger is in a photo or not. Since such facial attributes are relatively stable, if a person is in a photo, by comparing the faces in the photo with his recent profile photo in facial attributes, the person can be correctly matched to his face in the photo. Also, in photographing scenarios, it is not very likely that the facial attributes of two nearby strangers are exactly same, since the number of persons in a limited geographic area around the photographer is usually not large. That means if facial attributes can be accurately identified from photos, mismatch between faces and strangers will be of a low chance. Thus intuitively facial attribute-based face-stranger matching is a promising method to explore.

Based on facial attributes, we design a cooperative scheme PoliteCamera [8] to protect the privacy of strangers who are unintentionally included in photos taken by mobile phones.
PoliteCamera works as an application on the mobile phones for both the photographer and the stranger. When a photographer takes a photo, he (via the mobile phone) will notify nearby strangers of the potential risk of being included in the photo via peer-to-peer short-range wireless communications (e.g., WiFi Direct [9]). If a stranger prefers not to be included in the photo, he can send a blurring request to the photographer together with his facial attributes included in the request. The photographer will check whether the requesting stranger’s face appears in the photo or not based on the facial attributes sent from the stranger and the facial attributes of faces captured in the photo. If the attributes of a face in the photo match those of the requesting stranger, that face is considered to be the stranger’s and it will be blurred in the photo.

The set of facial attributes will be carefully selected so that a combination of attribute values is specific enough to differentiate different strangers nearby the photographer but is not specific enough to uniquely identify who the requesting stranger is in the real world. The number of possible attribute value combinations should be reasonably large (e.g., tens of thousand). Then the probability for two different strangers to have the same combination is low, since the number of strangers around a photographer is usually small. The number of possible combinations should also not be too large. In this way, each combination could be owned by many people in the real world, and thus cannot be used to infer who the stranger is. As described later, approximate match instead of exact match will be used in PoliteCamera, which makes linking multiple appearances of the same person difficult. Thus, the privacy risk of re-identification will be low. Moreover, privacy-preserving computing technologies can also be applied to complete the matching of facial attributes without sending the stranger’s facial attributes to the photographer in cleartext, and in this way further protect the stranger’s facial attributes from the photographer (see Section 3.2.4 for a discussion).

The privacy protection offered by PoliteCamera is based on the cooperation between
photographers and strangers. Although these two roles are separately discussed, real-world users can take either role in different scenarios. Since every user can be a stranger in many scenarios, users have a motivation to use this system, and participation in this system means mutually protecting each others privacy and benefiting everyone including self. This inter-user cooperation design is also motivated by many real-world systems such as collaborative filtering recommender systems [10] and peer-to-peer video streaming systems [11]. Users’ privacy can be better protected when more people use this system. Although it is not a perfect solution for the problem, it still significantly advances the state of the art in this domain.

The contribution of this chapter is summarized as follows:

• We propose a facial attribute-based system PoliteCamera for protecting strangers’ privacy in mobile photographing. To the best of our knowledge, PoliteCamera is the first scheme that makes nearby strangers aware of possible inclusion in a photo when the photo is being taken, allowing them to determine whether to blur their face in the photo or not, and protects strangers’ privacy under both indoor and outdoor scenarios, without using any trusted server, human gesture, or special wearables.

• We design a novel adapted balanced convolutional neural network (ABCNN) that can simultaneously predict multiple facial attributes from a photo, and use it to determine the existence of requesting strangers in a photo.

• To avoid identifying the real target persons of a photo as a stranger, a heuristic approach is employed to effectively filter targets to prevent incorrect blurring.

• The proposed system is implemented, and extensively evaluated on real datasets and in the field. Experimental results show the excellent performance of the system.
The rest of this chapter is organized as follows. Section 3.2 introduces the design of PoliteCamera. Section 3.3 presents implementation. Section 3.4 shows evaluation results. Section 3.5 reviews related work. Section 3.6 concludes this chapter.

3.2 System Design

This section describes the design of PoliteCamera.

3.2.1 System Overview

Three types of entities are involved in the system: the *photographer* who takes a photo, the *target* who is intentionally captured by the photographer, and the *stranger* who is near the target and might be accidentally included in the photo.

The system is designed to protect the stranger’s privacy by giving an option to the stranger to opt out from the photo. The general idea is that the system notifies nearby strangers the possible inclusion in a photo, and blurs a stranger’s face if the stranger sends a blurring request. A naive approach is to blur every stranger’s face in the photo. However, this is not an ideal solution, since blurring will inevitably affect the quality of the photo. To minimize the effect on photo quality, our design only blurs a stranger’s face if he requests to do so. We assume PoliteCamera is installed on both the photographers and the stranger’s mobile phone. Each user of PoliteCamera provides one of his photos to the PoliteCamera app upon the installation of the system. Each users facial attributes are learned from this base photo and stored in the system for future use. (The base photo can be updated by the user but this does not need to be done frequently since facial attributes do not change frequently.) When a stranger requests a photographer to blur his face, he can send these attributes to the photographer and the photographer will determine whether his face is in the photo based on these facial attributes and blur his face if so.
There are two challenges with the approach. Firstly, there might be multiple nearby strangers who receive the notification of potential privacy leakage by the photo. Some of them may request to blur their faces but others may not request so. Hence, we need to determine if the requesting strangers' face is in the photo or not, which is not trivial. Secondly, when the target is a single person or multiple persons, we need to keep the target unblurred even if the target’s phone mistakenly sends out a blurring request. Telling the target from the stranger is necessary but difficult.

3.2.2 The Architecture and Workflow of PoliteCamera

As Fig. 3.2 shows, the system consists of six major modules: *face detection and pre-processing, blurring request and collection, facial attributes classifier, target filter, stranger determination* and *face blurring*. When a photographer takes a photo, the face detection module will run on the captured image. If any face is detected, the notification of possible inclusion in the photo will be sent to nearby strangers via peer-to-peer short-range wireless communications. If a stranger would like to blur his face in the photo, he sends a blurring request to the photographer. To help the photographer determine if the requesting stranger is in the photo, this stranger also sends his pre-computed facial attributes (e.g., gender, obtained from his face image when initializing the PoliteCamera app). Upon receiving blurring requests, the photographer crops all the faces in the picture, and then feed them into the pre-trained facial attributes classifier. By comparing the facial attributes of requesting strangers and the attributes of detected faces in the photo, the stranger determination module of photographer can identify those requesting strangers captured in the photo. If a requesting stranger is in the photo, the face blurring module of the photographer smoothly blurs the corresponding face; otherwise, the request is ignored. In case the target mistakenly sends a blurring request, the target filter module distinguishes the target from the stranger.
based on specific defined rules, and keeps the target unblurred in the photo.

The design of PoliteCamera depends on several available technologies in mobile phones. In particular, face detection and preprocessing can be implemented using APIs provided by the operating system on mobile phones, such as the FaceDetector APIs in Android SDK. Similarly, peer-to-peer short-range wireless communications can be set up by available technologies on most modern mobile phones, such as WiFi Direct [9] and Bluetooth. We will introduce the implementation of these two modules in Section 3.3. Next, we will illustrate more details about the rest four modules.

3.2.3 Facial Attribute Classifier

Given an input face image in pre-defined dimensions, this module aims to simultaneously output a set of facial attributes associated with this input image. In particular, each facial attribute is a binary label, where +1 indicates the presence of the corresponding attribute, and -1 means its absence. In this chapter, we propose to train a facial attribute classifier through the ABCNN model where a weighted objective function is constructed to maximize the prediction accuracy.

Formally, let $\mathbb{I}$ be the set of input images, and $N$ be the number of facial attributes. For a given image $x \in \mathbb{I}$, let $y_i \in \{-1, +1\}$ be the binary label of the $i$th attribute, where $i \in \{1, 2, \ldots, N\}$ is the index of facial attributes. Let $\mathbb{H}$ be the hypothesis space of possible decision functions, and $f_i(\theta^T x)$ be the decision function, where $\theta = \{\theta_1, \theta_2, \ldots, \theta_N\}$ is the network weights. Hence, the loss function of the $i$th facial attribute can be defined
as $L_i(f_i(\theta^T x), y_i)$. Let $\mathbb{E}(L_i)$ be the expected loss over the range of inputs $I$. Then the optimization task is to minimize the expected squared error for each attribute.

$$\forall i : f_i = \arg\min_{f_i \in H} \mathbb{E}(L_i) \quad (3.1)$$

For each input $x$ and attribute $i$, the corresponding classification result $c_i(x)$ and the according accuracy $acc_i(x)$ can be obtained from the output of $f_i(x)$ described as:

$$c_i(x) = \begin{cases} +1 & f_i(x) > 0 \\ -1 & \text{otherwise,} \end{cases}$$

and

$$acc_i(x, y) = \begin{cases} +1 & y_i(x)c_i(x) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

As discussed above, the traditional approach treats facial attributes as $N$ independent tasks, and each classifier is trained independently. The typical loss function for the $i$th facial attribute is constructed by choosing the hinge-loss function, which is shown as:

$$\arg\min_{\theta_i} L_i(f_i(\theta^T x), y_i) = \arg\min_{\theta_i} \max(0, 1 - y_i(x)f_i(\theta^T x))) \quad (3.3)$$

However, a problem with the traditional approach is that training independent classifiers cannot learn the latent correlations between attributes. To exploit such correlations, the classifier should be constructed to learn all of these facial attributes simultaneously. In addition, the attribute label distribution in the training set should match with the corresponding distribution in the testing set. Therefore, it is necessary to balance the dataset to train a better classifier. One way to obtain a balanced dataset is to perfectly collect evenly distributed dataset of images for each attribute. However, it will cause extra efforts since most
of data in real application is not evenly distributed, and finding such dataset may be very challenging especially at a large scale. An alternative solution is to modify the loss function in order to simulate a balanced dataset. In our proposed ABCNN, some changes are made to the objective function to address the imbalance between the training dataset and the test dataset. Specifically, a mixed objective function is proposed by considering the distribution difference between training data and testing data as adapted weights. Firstly, the training distribution $S_i$ for each attribute $i$ is computed by calculating the fraction of positive samples $\text{Train}^+_i$ (0 < $\text{Train}^+_i$ < 1) and fraction of negative samples $\text{Train}^-_i$ (0 < $\text{Train}^-_i$ < 1) in the training set. Given the binary testing target distribution $\text{Target}^+_i$ and $\text{Target}^-_i$ (where $\text{Target}^+_i + \text{Target}^-_i = 1$), an adapted weight is assigned for each class of attribute $i$, as shown in Eq. 3.4 and Eq. 3.5:

$$p(i|+) = 1 + \frac{\Delta T^+}{\text{Target}^+_i + \text{Train}^+_i}$$

(3.4)

$$p(i|-) = 1 + \frac{\Delta T^-}{\text{Target}^-_i + \text{Train}^-_i}$$

(3.5)

where $\Delta T^+ = \text{Target}^+_i - \text{Train}^+_i$ and $\Delta T^- = \text{Target}^-_i - \text{Train}^-_i$. It can be seen from the above equations that we will increase the weight of the $i$th facial attribute if the fraction of positive or negative labels in the training data is less than the testing data. The intuition is that the increment of those weights will help balance the distribution difference between training data and testing data. Correspondingly, we will decrease the fraction weights of positive or negative labels in the training data if it is higher than that in the testing data. Then, these adapted weights are incorporated into the mixed objective function. Instead of using the hinge-loss function, a weighted mixed task square error is adopted as the loss function, and the optimization problem of ABCNN can be expressed as:

$$\forall i : \arg \min_{\hat{f}_i \in \mathbb{H}} \mathbb{E}(L(x, y)) = \arg \min_{\hat{f}_i \in \mathbb{H}} \mathbb{E}(\sum_{i=1}^{N} p(i|y_i(x))||\hat{f}_i(x) - y_i(x)||^2)$$

(3.6)
The optimization problem aims to find the optimal decision function \( f \) that has the smallest error between predictions and target labels. Over an \( M \)-element training set \( X \) with labels \( Y \), from Eq. (6) we can get:

\[
\forall i : \arg \min_{f_i \in \mathbb{H}} \mathbb{E}(L(X, Y)) = \arg \min_{f_i \in \mathbb{H}} \mathbb{E}(\sum_{j=1}^{M} \sum_{i=1}^{N} p(i|Y_j|x) ||f_i(x) - Y_j||^2) \tag{3.7}
\]

The ABCNN architecture can be built by replacing the standard loss layer of a deep convolution neural network (DCNN) with a layer implementing Eq. 3.7. After the above classifier is trained, we can predict facial attributes by inputing a face image with fixed dimensions (which are consistent with that of training images) to the classifier.

### 3.2.4 Stranger Determination

This module aims to determine if a requesting stranger is included in the photo or not and which face matches the stranger. This is done through thresholding the difference between the facial attributes of the detected faces and those of the requesting stranger. In fact, facial attributes predicted by the classifier is a vector of binary values, where \(+1\) indicates the presence of the corresponding attribute, while \(-1\) represents its absence. The difference is defined as the number of different attributes between two faces under the same set of attributes. Formally, let \( N \) be the number of attributes associated with a face. For a given face, its corresponding attributes vector \( V = [a_1, \ldots, a_N] \), where \( a_i \in \{-1, +1\} \) represents the \( i \)th facial attribute. We use \( V_r \) and \( V_s \) to represent the facial attributes of the requesting stranger and a specific detected face respectively. The inner product of \( V_r \) and \( V_s \) is \( V_r \cdot V_s = \sum_{i=1}^{N} V_r[i]V_s[i] \). If all the attributes are identical that inner product should be \( N \). The \( V_r[i]V_s[i] \) is \(-1\) only when the \( i \)th attribute in \( V_r \) and the \( i \)th attribute in \( V_s \) are different. Hence, the difference can be obtained as:

\[
diff = \frac{N - V_r \cdot V_s}{2} \tag{3.8}
\]
As discussed before the facial attributes cannot be used to uniquely identify a stranger. In order to further protect the stranger’s facial attributes from the photographer, inner product computation can be done with a two-party privacy-preserving scheme [12]. Usually the predication results from two images from the same person cannot match exactly due to angle difference or some other reasons. Thus a threshold is set to tolerate such minor deviations. The rule is that only the difference between facial attributes of the requesting stranger and any specific detected face is less than or equal to the threshold, we consider the detected face belongs to the requesting stranger. Our evaluations show that it is a good choice to set 1 as the threshold.

3.2.5 Target Filter

This module is designed to distinguish the target from the stranger in a photo, so that the target’s face will not be blurred even if the target mistakenly sends a blurring request. Specifically, if the target of a photo is one or multiple persons, the task is filtering out the targeted faces; if the target is a building or something else, we would like to avoid the stranger being mistakenly identified as the target. Therefore, a heuristic approach is proposed to achieve this goal. Based on our observations from real-world experience, the target is usually associated with the following properties in the photo:

- One common goal of taking photos is recording beautiful moments. The target is likely to be smiling when he is being pictured, since smiles make a person more attractive and confident.

- The photographer usually intentionally makes the target’s face significantly larger than others who are accidentally included in the photo. For instance, if a stranger is too close to the camera and hence his face is larger than the target’s, the photographer will usually stop picturing or move a little bit so that the target is better captured into the
photo. Moreover, considering that there might be multiple targets appearing in the photo but with slightly different face sizes (e.g., a group of people taking a picture), we expect to filter all targets in the photo by comparing a detected face with the largest face in the photo, which is considered as one of the targets’ faces by default. If the size difference is less than a pre-defined threshold, we consider the detected face as one of the target faces.

- Similarly, the photographer usually puts the target in a dominant position of the photo. The central region is one of the most popular options, which can highlight the target in the photo.

Consequently, smiling, face size and face position can facilitate determining if a face belongs to the target or not. Based on these observations, we propose three rules to determine whether a person in the photo is a target or not.

1. The person is smiling.

2. The person’s face is the largest one in the photo or slightly smaller than the largest one by a pre-defined threshold. Based on our test, we find that the average size difference between two targets’ faces in a photo is around 10%. Hence, if more than one face is detected, we compare the largest one with the others. If the size difference between the largest one and a certain face is less than or equal to 10%, we consider that face as one target face. Otherwise, the detected face will not be treated as a target.

3. The person’s face appears at the central region of the photo. The central region is defined as the middle section of horizontal trisections of a photo.

However, it is too strict if we determine a detected face is the target only when all those three rules are satisfied, since sometimes not all of them are satisfied. For instance, the target is
not always smiling when the photo is taken. Considering this, we determine that the face is the target if at least two of the three rules are satisfied.

### 3.2.6 Face Blurring

The purpose of face blurring is to mask the features of a face in order to make the face not recognizable, without degrading the quality of photo much. Similar to our previous work [5], we adopt an approach based on the Gaussian Blur algorithm [13] to smoothly blur faces. To conduct face blurring, we need to determine a blurring area in the face enclosing the main identifiable features of the face. In particular, we draw a square whose side length is 2.4 times of the distance between eyes, and whose center is the middle point between eyes. Then the Gaussian Blur operation can be performed in the square blurring area.

### 3.3 Implementation

The facial attribute classifier was implemented using Python 2.7 and MxNet [14], which is an open-source deep learning framework. WiFi Direct was used to conduct peer-to-peer communications between the stranger and the photographer. The face blurring module was implemented as same as our previous work [5], so some details are omitted here.

#### 3.3.1 Face Detection and Preprocessing

Face detection is based on the FaceDetector class provided in Android SDK. Faces in an image can be detected by calling the findFaces method of FaceDetector. This method detects faces by finding pupils in the image, and returns a number of detected faces into an array of FaceDetector.Faces class. For each instance of Face class, the distance between two eyes of a face and the coordinate of the middle point between two eyes can be obtained. Then we crop each detected face with a square area, which is the same as the blurring square
described in Section 3.2.6. Also, the size of the cropped square is used to represent the size of the corresponding face in the target filter module. To prepare for target filtering, we need to detect the position of each face in the photo. To do so, we evenly divide the picture into three regions (left, middle, right) along the horizontal direction. Then for each detected face, we calculate the middle point between its eyes. If the middle point is located in the middle region, we say this face is in the central region.

3.3.2 Facial Attribute Classifier

This module aims to predict a set of facial attributes from a given face image. As described in Section 3.2.3, we use ABCNN to predict the facial attributes and ABCNN is implemented by the Python interface of MxNet [14]. In particular, we build the ABCNN network by replacing the final loss layer of a 16-layer VGG network from [15] by the loss function in Eq. 3.7, and the architecture shown in Fig. 3. The architecture consists of 16 weight layers, including 13 convolution layers and 3 fully connected layers, which are associated with over one million weights. Since the network only accepts RGB image input with dimensions of 128*128 pixels, each cropped face obtained from the face detection and preprocessing module should be scaled to that size before being sent into this classifier.

![Figure 3.3: Architecture of the ABCNN network.](image)

In this chapter, the ABCNN network is trained on the CelebA dataset [16], which is a large-scale facial attributes dataset. It contains 20 images for each of over 10K celebrities,
hence with a total of more than 200K images. The first 160K images are used for training, and the remaining 40K images are used for validation and testing, specifically, 20K for validation and 20K for testing. For our implementation, we use a set of pre-cropped and aligned face images provided by the CelebA dataset, and scale the dimensions of training RGB images from 178*218 pixels to 128*128 pixels. Each image in the CelebA dataset is annotated with binary labels of 40 facial attributes (e.g., ‘Young’ and ‘Male’). However, in this work, we choose 16 out of the 40 attributes that do not change frequently for the same person as our considered attributes. The 16 chosen facial attributes include \{Arched Eyebrows, Bushy Eyebrows, Big Lips, Big Nose, Point Nose, Black Hair, Blond Hair, Brown Hair, Gray Hair, Eyeglasses, Bald, High Cheekbones, Narrow Eyes, Oval Face, Male, Young\}. In addition, since the ‘smiling’ attribute is required for target filtering, we also add it into the classifier (note that it is not used for stranger determination but only for target filtering).

3.4 Evaluations

To train the classifier, we set the batch size to 384 images per training iteration, and hence the training process requires approximately 420 iterations to finish a full epoch on the training set. The learning rate is initialized as 0.05, and reduced by a factor of 0.8 every four epochs until it decays to 0.000001. We train the ABCNN for 110 epochs with all images from training set on two NVidia K80 GPUs.

3.4.1 Model Selection

Classification accuracy is defined as the number of correctly predicted cases divided by the number of testing images. From Eq. 3.2, we can derive the classification accuracy of
each attribute $i$:

$$e_i(X, Y) = \frac{1}{N_{\text{test}}} \sum_{j=1}^{N_{\text{test}}} acc_i(X_j, Y_j) \quad (3.9)$$

Consequently, we can evaluate the average classification accuracy by calculating the average classification accuracy over all the $N$ attributes:

$$E(X, Y) = \frac{1}{N} \sum_{i=1}^{N} e_i(X, Y) \quad (3.10)$$

The ABCNN prediction model is trained on the training dataset, but the number of training epochs needed is determined based on the validation dataset. Specifically, the accuracy trend when the number of training epochs increases is shown in Fig. 3.4. As the training continues, the accuracy over the training dataset keeps increasing. However, training for more epochs means higher cost. Thus, based on the maximum accuracy over the validation dataset, we stop training the ABCNN network after 80 epochs (with 89.84% validation accuracy) and use the resulted model for performance evaluations in order to guarantee the coverage of the model without too high cost.

![Figure 3.4: Average classification accuracy vs training epochs.](image)

Then based on Eq. 3.9 we evaluate the classification accuracy of each facial attribute on the test dataset, including 16 attributes used for stranger determination and the ‘Smiling’
attribute for target filtering. The average accuracy over those 16 attributes is also tested according to Eq. 3.10. As Fig. 3.5 shows, the average accuracy is 88.53% (see the horizontal dashed line) which is pretty high. Out of the first 16 facial attributes, 6 attributes outperform the average performance, including *Bushy Eyebrows, Black Hair, Blond Hair, Gray Hair, Eyeglasses, Bald* and Male. For example, the classification accuracies of *Eyeglasses* and *Bald* achieves 98.31% and 98.34%, respectively.

To measure the performance of our proposed ABCNN in predicting the facial attributes, we compared it with the state-of-art algorithm proposed in [17]. They also construct a multi-task training classifier and the corresponding facial attribute prediction and average accuracy are represented with the blue dashed line and the horizontal blue solid line in Fig. 5, respectively. In addition, we also compared the proposed ABCNN with [16] which uses the basic CNN model to select features and inputs them to the SVM classifier for training. Its performance is displayed by green line and green dashed line for facial attributes prediction accuracy and average accuracy, respectively. ABCNN outperforms both the multi-task training classifier in [17] and the CNN-SVM model [13].

![Figure 3.5](image)

**Figure 3.5:** Classification accuracy of each attribute and average accuracy in testing.
### 3.4.2 Classification Consistency

Since the facial attributes are used for stranger determination, the trained classifier is expected to make consistent predictions given a specific person. That is, given two different face images of the same person, ideally all the 16 facial attributes obtained from the two images are identical. To evaluate classification consistency, we use the LFW image database [17] that has been widely used in the literature. Since images in the LFW database are organized by person, it is more efficient to sample images for experiment. In this experiment, we randomly pick 50 persons, and a pair of different face images of each person (see Fig. 3.6 as an example). The classification results over the two images in Fig. 3.6 are presented in Table 3.1. It can be seen that the classified facial attributes of these two images are exactly the same except ‘Big Lips’, ‘Brown Hair’ and ‘High Cheekbones’. Out of the 50 persons, the classification results for 32 persons are fully consistent. For the rest 18 persons, 7 persons have 15 identical attributes, 8 persons have 14 identical attributes, and the remaining 3 persons have 13 identical attributes.

![Figure 3.6](image_url)

**Figure 3.6**: Two different face images from the same person [17].

Besides, we examine the classification consistency on persons with more than 4 face images in the LFW dataset. In particular, we pick 2 pairs of different face images for each of those 10 persons. Then we compare the predicted attributes pair by pair, and hence perform 20-pair comparisons. As Table 3.2 shows, 8 pairs of face images are labeled with the exactly
Table 3.1: Facial Attributes Classification of Fig. 3.6(a) and Fig. 3.6(b).

<table>
<thead>
<tr>
<th>Facial Attributes</th>
<th>Fig. 3.6(a)</th>
<th>Fig. 3.6(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arched Eyebrows</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Bushy Eyebrows</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Big Lips</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Big Nose</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Pointy Nose</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Black Hair</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Brown Hair</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Blond Hair</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Gray Hair</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Eyeglasses</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Bald</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>High Cheekbones</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Narrow Eyes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Oval Face</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Young</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

the same attributes, and only 6 pairs are labeled with 3 or more different attributes.

Table 3.2: Classification consistency of 10 persons with 2 pairs of face images each.

<table>
<thead>
<tr>
<th>Number of Identical Attributes</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Pairs</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Furthermore, we examine the possibility of two different persons being predicted with identical attributes. We randomly pick 100 persons from the LFW dataset, and perform facial attribute classification on a face image of each person. Then, we compare facial attributes of every person with those of the other 99 persons and hence 4950 pairs are compared in total. Only 144 pairs have exactly the same attributes. All these results show that the classification consistency is high.

3.4.3 Optimal Thresholding

The above consistency experiments show that facial attributes of two face images from the same person may not be perfectly identical. Hence, a scheme that depends on exactly
matching of facial attributes between two faces will not work for stranger determination. The stranger determination is implemented by thresholding the difference of facial attributes between two compared face images to allow a reasonable difference between these two faces. Hence, it is needed to find a proper threshold. The goal is that we can obtain more true positives without causing too many false positives under the threshold. Here, true positive means two different face images of the same person being determined as the same person. False positive means images of two different persons being determined are from the same person. In this experiment, we pick 50 persons from the LFW database, and two different face images with each person. In order to evaluate false positive, 50 tests are conducted. In each test, we pick one face image from the above 50 persons as the target, and choose another face image from a different person to compare with the target. From the above classification consistency evaluations, we consider 0, 1 and 2 as reasonable threshold candidates and show the results in Table 3.3. Based on these results, we choose 1 as the threshold in stranger determination which has good performance in both true positive and false positive.

<table>
<thead>
<tr>
<th></th>
<th>Threshold=0</th>
<th>Threshold=1</th>
<th>Threshold=2</th>
</tr>
</thead>
<tbody>
<tr>
<td># True Positives</td>
<td>36</td>
<td>45</td>
<td>48</td>
</tr>
<tr>
<td># False Positives</td>
<td>1</td>
<td>3</td>
<td>12</td>
</tr>
</tbody>
</table>

### 3.4.4 Effectiveness of Target Filter

This test aims to examine how well the target filter module can detect the target from a photo. In this experiment, we perform target filter on field photos from two different sources where multiple targets might be in one photo. We use false filtering rate to measure the performance, which is defined as the percentage of times when not all targets in the photo are successfully detected or any stranger appearing in the photo is mistakenly detected as the target.
First, we evaluate the effectiveness of target filter on 100 photos, which we have pictured by mobile phone in the past, and at least one target person is included in each photo. The result shows that the false filtering rate is only 8%, which means the target filter only fails to detect the target in 8 photos. Fig. 3.7 illustrates two example photos of our test. Fig. 3.7(a) is a successful example, but Fig. 3.7(b) is a failed example. The reason for unsuccessful target detection is that the face is not at the central region of the photo, and Smiling attribute is falsely predicted as No. Based on our proposed three rules, only the rule based on face size can be satisfied, and hence the target is not successfully detected.

Then we pick 100 photos shared by our friends in Facebook from 10/01/2016 to 12/26/2016. At least one target person is included in each photo. Fig. 3.8 shows some example photos, where faces are blurred upon the friends request. Similar to the above test, we run target filtering on these 100 photos. The false filtering rate is 12%, which means the target filtering operation fails in 12 photos. We look into each of those 12 photos, and find the same reason causing false target filtering. When multiple targets are shown in the photo, the target at the rightmost or leftmost is detected as out of the central region of the photo. Also, this target was not smiling when the photo was taken or the Smiling attribute is falsely predicted as No. As a result, in those cases, the rules based on face position and smiling cannot be satisfied, and hence the target filter cannot successfully detect all the targets in the photo. However, the overall target filtering accuracy is still high.
Figure 3.8: Target filter test on photos shared by friends on Facebook (used with permission).

3.4.5 Accuracy of Protection

This part evaluates the effectiveness of our system in protecting the stranger’s privacy. The experiments are conducted on our campus. Fig. 3.9 shows two example experiment scenes.

Figure 3.9: Example Experiment Scenes (photo by author).

**True Protection Rate:** This group of tests considers the scenario where one target person and two strangers appear in the photo. We assume either one of the two strangers or both of them request face blurring. The *true protection rate* is defined as ratio of times when the faces of the requesting strangers are blurred in the photo. For each requesting stranger, we conduct 10 tests separately. Fig. 3.10 shows an example where the right stranger’s face is successfully blurred. Table 3.4 shows the true protection rate which is high.

**False Protection Rate:** Again we consider the scenario where one target person and two strangers appear in the photo. Suppose the two strangers in the photo do not request
Table 3.4: True Protection Rate

<table>
<thead>
<tr>
<th># Requesting Strangers</th>
<th>True Protection Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90%</td>
</tr>
<tr>
<td>2</td>
<td>80%</td>
</tr>
</tbody>
</table>

to blur their faces but other nearby strangers who are not in the photo submit blurring requests. In this case, we define false protection rate as the percentage of times when any of two strangers in the photo is mistakenly detected as a requesting stranger and hence falsely blurred. To evaluate the false protection rate in a noisy environment, we conduct simulations with 1, 3, 5 and 10 nearby requesting strangers separately. Specifically, in each test, we randomly pick a certain number of entities from the LFW database, who act as nearby requesting strangers, and one face image for each selected person. For each specific number of requesting strangers, 50 tests are conducted separately. Table 3.5 shows results with different number of requesting strangers. We can see that false protection rate increases with the increasing number of nearby requesting strangers. This is because the more nearby requesting strangers, the higher possibility of their facial attributes being overlapped with that of strangers in the photo. Note that the false protection rate is as low as 3% with only one nearby requesting stranger. Even under noisy environment with 3 nearby strangers who request face blurring, the false protection is only 8%. The false protection rate increases to 24% with 10 nearby requesting strangers, but this case does not occur often in the real
world.

<table>
<thead>
<tr>
<th># Nearby Requesting Strangers</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Protection Rate</td>
<td>3%</td>
<td>8%</td>
<td>14%</td>
<td>24%</td>
</tr>
</tbody>
</table>

### 3.5 Related Work

See Chapter 2.5 for references of photo and video privacy.

**Facial Attributes Classification** Kumar et al. [28] propose an approach to train facial attribute classifiers. Features from manually-picked facial regions for each facial attribute are separately optimized using AdaBoost algorithms. In addition, independent SVM classifiers are trained by feeding optimized features. In this approach, various features are learnt for each facial attribute, and an independent SVM classifier is separately trained. Even though it is a valid approach, it is not efficient for feature extraction and classification.

Recently, with the increasing popularity of convolution neural network (CNN), it has been leveraged to extract more sophisticated features of facial attributes. For instance, Kang et al. [29] propose gated CNNs, which aim to determine which regions of a face are most correlated to corresponding attributes. Then, the output of such CNNs is encoded into a global feature vector for training independent binary SVM classifiers. Zhang et al. [30] apply CNNs to learn facial attributes, which are used to infer social relations between pairs of identities with an image. Liu et al. [16] design three CNNs, including two localization networks (L Nets) and an attribute recognition network (ANet). LNet is designed for localizing features in face images, while ANet is trained on face identities and attributes to extract features. Then, independent SVM classifiers are trained on those extracted features. However, none of them can be directly used for imbalanced distributed datasets.
3.6 Summary

We proposed a system PoliteCamera to protect strangers’ privacy who are accidentally captured in a photo taken by mobile phones. The system can inform nearby strangers that they are possibly included in a photo and give them an option to blur their faces in the photo. A novel ABCNN structure is designed to predict facial attributes, where the facial attributes are used to determine whether a requesting stranger is in the photo and which face in the photo belongs to him. We implemented a prototype system, and evaluated its performance through experiments. The accuracy of the facial attributes prediction is better than the state of the art, and experimental evaluations demonstrate that the system can effectively protect strangers privacy.

Bibliography


4 Feature-Based Model for Differentiating the Target From Strangers in Photos

4.1 Introduction

Nowadays, powerful cameras are embedded in mobile phones. It changes the way people to record their daily life, an increasing number of people tend to take photos using mobile phones anywhere and anytime. However, as introduced in Chapter 2 and Chapter 3, the convenience brought by mobile cameras cause privacy issues. When a user takes a photo of a beautiful view or a friend using his mobile phone, frequently a stranger is also accidentally included in the photo, with the face clearly recognizable. In such cases, the photo can reveal the stranger’s location and even activity, and hence breaches his privacy.

Even though we propose two systems PrivacyCamera and PoliteCamera in Chapter 2 an Chapter 3 to protect strangers’ privacy in photos, these solutions depend on the cooperation between the photographer and the stranger. Specifically, both the photographer and the stranger need to install those systems on their mobile phones, and the stranger who does not want to appear in the photo needs to send either GPS location information or facial attributes to the photographer for determining whether he is in the photo or not. Although those solutions can provide effective privacy protections, there still exist some limitations. First, we cannot guarantee every user will install those systems. Therefore, even if the photographer is willing to protect the strangers’ privacy, the protection cannot be successful if some of the strangers do not install those systems. Second, the communication between the photographer and the stranger may not be reliable. If the communication channel is disconnected, the blurring request and associated information such as GPS location and facial attributes cannot return to the photographer, and hence the system cannot perform stranger
determination and provide effective privacy protection.

Can we protect strangers’ privacy in photos while addressing the aforementioned limitations? We can perfectly address those issues if we can get rid of the cooperation between the photographer and the stranger. Specifically, if all the strangers’ faces can be automatically detected and blurred in a photo, the cooperation is not necessary. However, there is a challenge in this case. Since the target might be a person that is intentionally captured in the photo and should not be blurred, how to automatically distinguish the target from the stranger in a photo is a very challenging task.

Even though we proposed a heuristic rule based approach to distinguish the target from the stranger in a photo in Chapter 3.2.5, the performance is not very good through experiments. In that approach, we proposed three properties as rules based on our observations from real-world experience: smiling, face size and face position. Then, we make the distinction based on such rules. Although the proposed approach works for a majority of cases, there are some exceptions in special cases.

In this chapter, we explore more features to improve distinction accuracy. In addition, we build a binary classifier based on these features using supervised machine learning method.

The contribution of this chapter is summarized as follows:

• We propose a set of features, based on which we build a binary classifier to distinguish the target from the stranger in a photo.

• We implement the classifier based on proposed features using several supervised learning algorithms, and explore their performance.

• We explore deeper in how feature selections affect classification accuracy, and find out the features that are important to the model.
The rest of this chapter is organized as follows. Section 4.2 introduces the machine learning algorithms that are applied in this chapter. Section 4.3 proposes a set of features to improve the distinguishment between the target and the stranger. Section 4.4 describes the dataset collection. Section 4.5 shows evaluation results. Section 4.6 concludes this chapter.

4.2 Background

This section introduces the supervised learning algorithms that are applied in this chapter.

**Decision Tree.** Decision tree [1] is a non-parametric supervised learning algorithm that can be used for classification. The decision tree model aims to predict the value of a target variable by learning simple decision rules that are inferred from training data. To construct a decision tree, the most significant attribute will be set as the root. Then, the dataset can be split based on the values of attribute, and repeat this process until leaves. The significance of an attribute is measured by information gain [2]. The larger the information gain, the more significant the attribute. Once the decision tree is built, it is easy to interpret and can be visualized.

**Random Forest.** Random forest [3] is an ensemble learning algorithm based on decision trees. The prediction is computed as the averaged prediction of each individual decision tree. Compared with the decision tree, we build each tree in the ensemble using a sample drawn with replacement from training data. Besides, for constructing the tree, we choose the attribute that is the best split among a random subset of the features instead of the best split among all features. Even though the bias of the forest slight increases due to the randomness, the variance usually decreases more than compensating for the increase in bias. The random forest has a significantly low risk of overfitting and hence make good generalization.
Gradient Boosting Decision Tree (GBDT). GBDT [4, 5] is another typical ensemble method based on decision trees. Compared with random forest, GBDT is composed of shallow decision trees instead of fully grown decision trees. In addition, in order to train the GBDT model, we assign different weights for samples in training data. The weights will be dynamically adjusted based on the classification accuracy on each sample. The sample with worst performance will be assigned largest weight, so that it can attract the model’s more attention. Furthermore, the prediction is made by the weighted average prediction of each individual decision tree.

Support Vector Machine (SVM). SVM [6] is a supervised machine learning algorithm that can be employed for classification, regression and outliers detection. SVMs are more commonly used in classification problems, which is also our focus in this chapter. SVMs are implemented by finding a hyperplane that can best divides variables into two classes. Support vectors are referred to critical data points nearest to the hyperplane. If we remove them the position of the dividing hyperplane will be altered. It is very effective in high-dimensional spaces. In addition, it is flexible to specify the decision function by using different kernel functions.

Multi-Layer Perceptron (MLP). A MLP is class of feedforward artificial neural network [7, 8]. There can be one more hidden layers between the input and output layer. Except for the input nodes, each node in the hidden layer adopts a non-linear activation function. It can learn a non-linear function estimator for both classification and regression problems, which is the critical difference from a linear perceptron.

4.3 Feature Selection

In Chapter 3, we proposed a heuristic rule based approach to distinguish the target from the stranger in a photo. Three features smiling, face size and face position are
considered in that approach. In this chapter, in addition to those features, we take more features into account to improve the classification accuracy based on our observation from hundreds of online photos in different settings.

**Blurriness**  Since the target is intentionally captured by the photographer, he is usually located in focus area of a photo. Compared with focus area, the area out of focus is usually more blurry. How can we measure the blurriness of a given area in a photo? One straightforward approach [9] is to compute the Fast Fourier Transform of that area and check the distribution of low and high frequencies. If there is a low amount of high frequencies, that area can be considered as blurry. However, it is very challenging to define what is an optimal threshold. In this chapter, we adopt the variation of Laplacian [10] to measure the blurriness as shown in Equation 4.1. The reason why we choose this method is due to the definition of Laplacian itself, which is computed to measure the second derivative of an image. The high variance indicates that there are rapid intensity changes, which can be used to infer that the image is in focus. Generally, given an area in a photo, the higher the Laplacian variance, the more possible that area is in focus. As shown in Figure 4.1, the target’s face is much clearer than the strangers’ faces in this photo, which is consistent with that the blurriness of the target’s face is much larger than that of the strangers’ faces.

\[
Var(Lap(x, y)) = Var\left(\frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}\right), \text{ where } I(x, y) \text{ is the pixel intensity values.} \tag{4.1}
\]

**Pitch**  As Figure 4.2 shows, the pitch can be defined as the head rotation angle around the \(x\) axis. The pitch is measured in range of \(-30^\circ\) to \(30^\circ\), which is a reasonable range that can cover most cases. Beyond that range, the face is not guaranteed to be detected. As a result, if one’s face can be detected, we treat the pitch of it as \(30^\circ\) or \(-30^\circ\) even if the pitch is over that.
Figure 4.1: The example of blurriness measurement in a photo [11]

Yaw  As shown in Figure 4.3, the yaw is an estimate of head rotation angle around the y axis, ranging from -30° to 30°. Beyond that range, the face is hard to be detected by the system.

Figure 4.3: Orientation of the face in terms of yaw.
**Figure 4.2**: Orientation of the face in terms of pitch.

**Roll**  Figure 4.3 shows the orientation of the face in terms of roll. It ranges from -180° to 180°.

Moreover, the smiling feature is defined as a binary variable in Chapter 3, but we change it to a numerical variable in this chapter. Finally, we can build a binary classifier based on these seven features, and the range of each feature is listed in Table 4.1.
Table 4.1: The Range of Each Feature

<table>
<thead>
<tr>
<th>Feature</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smiling</td>
<td>[0, 100]</td>
</tr>
<tr>
<td>Face Size</td>
<td>(0, 1]</td>
</tr>
<tr>
<td>Face Position</td>
<td>(0, 1)</td>
</tr>
<tr>
<td>Blurriness</td>
<td>[0, +∞)</td>
</tr>
<tr>
<td>Pitch</td>
<td>[-30, +30]</td>
</tr>
<tr>
<td>Yaw</td>
<td>[-30, +30]</td>
</tr>
<tr>
<td>Roll</td>
<td>[-180, +180]</td>
</tr>
</tbody>
</table>

4.4 Dataset

Since there is no public available dataset for distinguishing the target from the stranger in photos, we collect our own dataset and manually label the data. The dataset consists of 214 photos with 389 detected faces in total. Out of 389 detected faces, 203 faces are labeled as stranger, and the other 186 faces are labeled as target.

To manually label the data, the captions of collected photos can give us some hints in most cases. For example, if a caption refers that the photo is related to a celebrity, it is obvious that the celebrity in the photo is the target. In other cases, we label the target and the stranger in a photo based on our photographing experience.

To collect the dataset, we mainly focus on the photos including celebrities that are taken in public scenarios, such as airports and attractions. The reason that we consider such scenarios is that the celebrity in the photo is usually the target and easy to identify, and some strangers are often captured in the photo. The most important principle for collecting the dataset is to cover as many settings as possible. For example, as shown in Figure 4.5 the dataset contains the photos that include only the target, only the stranger and both of them.

In addition, from the perspective of features, we aim to cover both regular cases and special cases. For instance, in terms of the face position, the target is usually close to the
Figure 4.5: Example photos of different settings in terms of inclusion of the target and the stranger.
center of the view. However, there exist special cases where the target is located aside in a photo. Figure 4.6 shows aforementioned two cases in terms of the target’s face position in a photo. The similar data-collecting principle is also applied to other considered features.

![Figure 4.6: Example photos of different settings in terms of the target’s face position [11].](image)

**Figure 4.6:** Example photos of different settings in terms of the target’s face position [11].

### 4.5 Evaluations

In this section, we evaluate the performance of the proposed feature-based model. We train the model based different supervised learning algorithms using Scikit-learn [12]. We also compare the performance of each model and explore how feature selections affect the classification accuracy.

#### 4.5.1 Classification Accuracy

In Table 4.2, we compare the classification accuracy of each model. The GBDT model beats the other models with 93.27% accuracy, and thus we adopt this model as our final classifier for the rest of experiments. However, the classification of each model is over 91%, which demonstrates our proposed features are effective for distinguishing the target from the stranger in photos. It also means that the feature-based model is not sensitive to different learning algorithms.
4.5.2 Exploration of Feature Selection

After comparing the classification accuracy, we want to explore how different feature selections affect the performance. Even though the decision tree model is not the best one in terms of performance, its structure can show the significance of each feature. After looking into the tree structure, the ‘face size’ is the most important feature and the ‘face position’ is another important feature. However, it shows that yaw is the least significant feature. To explore further in feature selection, we compare the classification accuracy of the GBDT model based on different feature set by removing the significant features and the least important feature and keeping the reminded features same. As shown in Table 4.3, the classification accuracy dramatically decreases to 73.07% when we remove both ‘face size’ and ‘face position’ features. If we remove the ‘face size’ or ‘face position’ feature alone, the accuracy drops to 82.05% and 84.61% separately. However, the accuracy keeps stable when ‘yaw’ is moved out, which indicates ‘yaw’ is not a useful feature for our task, and hence we can remove it when training the model.

Table 4.3: The Evaluation of Classification Accuracy Based on Different Feature Sets

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Feature Set</td>
<td>93.27%</td>
</tr>
<tr>
<td>Without ‘Face Size’</td>
<td>82.05%</td>
</tr>
<tr>
<td>Without ‘Face Position’</td>
<td>84.61%</td>
</tr>
<tr>
<td>Without ‘Face Size’ and ‘Face Position’</td>
<td>73.07%</td>
</tr>
<tr>
<td>Without ‘Yaw’</td>
<td>93.27%</td>
</tr>
</tbody>
</table>
4.5.3 Performance Comparison with Heuristic Rule Based Target Filter

Next, we compare the performance of our proposed feature-based model with the heuristic rule based target filter that is presented in Section 3.4.4. To make performance evaluation, we use 42 photos from our dataset with 80 detected faces. We use true detection rate to measure the performance, which is defined as the percentage of times when the faces are correctly detected as the target or the stranger in photos. As shown in Table 4.4, the true detection rate of the feature-based model is 92.5%, but the heuristic rule based model only achieves 81.25% true detection rate. Therefore, the feature-based model is more effective to detect the target from a photo than the heuristic rule based model.

<table>
<thead>
<tr>
<th></th>
<th>Feature Based Model</th>
<th>Heuristic Rule Based Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>True detection rate</td>
<td>92.5%</td>
<td>81.25%</td>
</tr>
</tbody>
</table>

4.6 Summary

We proposed a feature-based model to distinguish the target from the stranger in a photo. Based on this model, we can protecting strangers’ privacy in a photo by automatically detecting them and blurring their faces in the photo, and hence there is no need for the cooperation between the photographer and the stranger compared with the solutions in Chapter 2 and Chapter 3. We implemented the model based on different supervised learning algorithms and compared their performance. Evaluations show that the feature-based model can effectively tell the target from the stranger in a photo. In addition, the performance of the feature-based model is better than that of the heuristic rule based approach. Moreover, we explored how feature selections affect the performance.
Bibliography


5  PhotoSafer: Content-based and Context-aware Private Photo Protection for Smartphones

5.1 Introduction

Smartphones have shifted the way people take, store and share photos. There is an increasing number of people who are able to take photos with smartphones anytime, anywhere. Also, almost all social networks allow users to share photos from corresponding smartphone apps (e.g., Instagram). Consequently, more and more people prefer to store photos on their smartphones for convenience, even though some photos are private and sensitive (e.g., drivers license). It is reported that the average person has 630 photos stored on their phones [1]. However, many installed apps on smartphones have access to stored photos and the network, which may cause leakage of private photos to remote parties. This raises a privacy concern that users private photos might be accessed by apps without their awareness.

The Android platform offers users two approaches for controlling permissions. At the early stage of Android, users are asked to grant permissions when they install an app. Specifically, an app will disclose the full list of resources that it wants to access at installation. Either all requested permissions are granted, or the entire installation is aborted. Prior research has shown that most users do not care about or understand these disclosures at installation [2]. With the evolution of the Android platform, a new permission scheme has replaced the install-time disclosures for enhancing smoothness of installation process. In particular, users need to grant permissions only when an app requests a sensitive resource for the first use. The users’ decisions to these permission requests will be applied to all future requests by that app for the same permission. However, this scheme only considers
the user’s preference for permission requests when an app is used for the first time. An app
once granted access to a photo at the first access will be able to access all photos for all the
time. It does not account for the fact that the user’s preference for subsequent permission
requests might change under different contextual circumstances. For instance, a user is
willing to upload a photo that was taken in a private gathering through a social-network
app. However, the same user might feel uncomfortable for the same app running in the
background to access such private photos without his awareness.

To protect private photos, some apps have been developed [3, 4, 5], which apply
authentication techniques (e.g., password and fingerprint) to control access to those photos.
However, they either significantly affect the usability and user experience or cannot really
secure private photos. Specifically, users are usually required to manually identify and import
private photos from the native photo gallery app on Android to such third-party apps. It
is very challenging and boring for users to manually select private photos from hundreds
or even thousands photos on their mobile phones. Moreover, some of these apps only copy
private photos to a specific protected folder but still keep them in the native photo gallery
app, which requires users to remove those private photos from the native photo gallery app.
If a user forgets to do so, no protection can be provided. Even worse, some apps merely move
user-specified private photos to a hidden folder, which can be easily detected and accessed
without any challenge by using existing file management apps [6, 7, 8]. In addition, when a
user wants to share private photos with other people through social network apps such as
Instagram, since these social network apps usually only allow users to choose to-be-shared
photos from the native gallery app or the file management system, it is inconvenient for users
if private photos are kept in separate app-specified folders. Hence, existing solutions cannot
really secure private photos while offering friendly user experience.

Some work has been done for refining Android permission systems. Nauman et al.
[9] and Jeon et al. [10] designed fine-grained permissions for Android, but do not specifically protect stored private photos. CHIPS [11] is a face-recognition-based access control system for stored photos on Android phones, but can only protect photos that contain pre-specified faces, which cannot be applied to other types of private photos (e.g., credit card).

To this end, we design a novel content-based, context-aware private photo protection system named PhotoSafer [12] for smartphones, which provides real-time access control over private photos based on the photo contents and the contextual status of accesses, and discloses the specific sensitive content that a private photo contains to users before that photo can be accessed. Our contributions are as follows:

- We analyze the top 200 free apps on the Apkpure, which is a very popular third-party Android app store, for evaluating the potential privacy risks that current apps pose to private photos.

- We conduct an online survey with 112 respondents to investigate mobile phone users’ privacy concerns about private photos, including common types of private photos, awareness of photo-accessing operations by apps, etc.

- We design a novel content-based, context-aware private photo protection system PhotoSafer, which can automatically identify private photos and perform real-time access control over private photos based on the context status of mobile phone and whether the requesting apps are running in the foreground.

- We implement a prototype system on Nexus 5 phones, and evaluate the system’s performance.

The rest of this chapter is organized as follows. Section 5.2 presents how this work is motivated, including a permissions analysis of 200 popular apps and an online survey.
Section 5.3 introduces the design and workflow of PhotoSafer. Section 5.4 describes the prototype implementation. Section 5.5 shows evaluation results. Section 5.7 reviews related work. Section 5.6 discusses the limitations of this work. Section 5.8 concludes this chapter.

5.2 Motivation

To better understand the privacy issues with photos stored on mobile phones, we firstly analyze the requested permissions of 200 apps to demonstrate the potential risk of unauthorized access to private photos, and then investigate users’ concerns about private photos in the real world through an online survey.

5.2.1 Permission Analysis

Let us first analyze what permissions are required to access stored photos on the Android platform. For an Android device, photos are stored in the external storage directory that can be either a physical removable memory card or a logical partition in the device’s memory. Hence, to access stored photos, an app has to be granted the permission `READ_EXTERNAL_STORAGE`, which is the only required permission. However, the external storage directory is not the repository only for photos, but also for other files such as songs. As a result, the correlation between the permission `READ_EXTERNAL_STORAGE` and photo access control is not intuitive to average users. In addition, due to the aforementioned limitations of the Android permission system, users are allowed to choose whether an app can access to all stored photos, but cannot realize selective control over any individual photo.

Next, we analyze apps’ requested permissions to examine the potential risk of unauthorized access to stored photos. Due to the download restrictions of Google Play, we analyze the top 200 free apps (e.g., Facebook, Twitter, etc.) from Apkpure [13], which is a popular third-party Android app store. We particularly identify apps that request
both `READ_EXTERNAL_STORAGE` and `INTERNET` permissions, since the combination of these two permissions allow potential leakage of private photos to another party. The analysis tool `Androguard` [14] is used to extract the requested permissions of each analyzed app. It is found that 164 out of the 200 apps request both `READ_EXTERNAL_STORAGE` and `INTERNET` permissions. That means 82% of the top 200 free apps on the Apkpure have complete access to stored photos on a user’s mobile phone, and could even leak these photos through the Internet. Thus, there is a necessity for finer-grained access control on private photos.

5.2.2 Online Survey

PhotoSafer’s design is also motivated by an online survey which is designed to investigate mobile phone users’ concerns about unauthorized access to private photos. The survey was conducted with user consent under an IRB approval from the University of Arkansas. The survey is available online [15], and the results here show statistics of all the 112 responses collected on December 22, 2017. The mobile phone platform usage of respondents is described in Table 5.1, and the age distribution of survey respondents is shown in Table 5.2.

<table>
<thead>
<tr>
<th>Mobile Phone Platform</th>
<th>Proportion of Respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>77.7%</td>
</tr>
<tr>
<td>iPhone</td>
<td>21.4%</td>
</tr>
<tr>
<td>Windows Phone</td>
<td>0.9%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Age Group</th>
<th>Proportion of Respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 20 years</td>
<td>5.4%</td>
</tr>
<tr>
<td>20-30 years</td>
<td>84.8%</td>
</tr>
<tr>
<td>30-40 years</td>
<td>9.8%</td>
</tr>
</tbody>
</table>

Participants were asked whether they store private photos (driver license, passport, etc.) on their mobile phones. An overwhelming majority (88.6%) deemed that some private
photos are stored on their mobile phones. To explore which specific type of photos are considered as private by respondents, this survey provides different options for participants. As shown in Table 5.3, almost every participant considered photos that contain Photo ID, Legal Documents and Family as private, and over a half (57.9%) agreed that nude photos are also sensitive. In a consequence, the above four types of photo contents are used as references to identify different categories of private photos for this work. Even though those types do not cover all cases, they can represent a significant portion of private photos in the real world.

**Table 5.3:** Photo Types and the Proportion of Respondents That Consider Them as Private

<table>
<thead>
<tr>
<th>Photo Type</th>
<th>Proportion of Respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photo ID (e.g., driver license, passport)</td>
<td>97.4%</td>
</tr>
<tr>
<td>Legal Documents (e.g. SSN)</td>
<td>97.4%</td>
</tr>
<tr>
<td>Family (e.g., family party)</td>
<td>76.5%</td>
</tr>
<tr>
<td>Nudes</td>
<td>57.9%</td>
</tr>
</tbody>
</table>

Furthermore, as shown in Table 5.4, 67.5% of participants agreed that there are more than 10 installed apps on their mobile phones that are granted access to photos. Also, for each of the participants, there is at least one installed app that has access to the photos stored on her/his mobile phone. However, the results show that most of participants (87.5%) do not clearly know whether any installed app can access photos in the background or not without their awareness. As a result, it is an urgent necessity to design a system to protect private photos from being accessed without users’ awareness.

**Table 5.4:** Number of Installed Apps That Can Access Photos

<table>
<thead>
<tr>
<th>Number of Installed Apps</th>
<th>Proportion of Respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>10+</td>
<td>67.5%</td>
</tr>
<tr>
<td>6-10</td>
<td>21.9%</td>
</tr>
<tr>
<td>1-5</td>
<td>10.4%</td>
</tr>
</tbody>
</table>
5.3 System Design

This section describes the design of PhotoSafer.

5.3.1 System Overview

Our goal for designing PhotoSafer is to protect private photos from unauthorized access by mobile apps, without changing the way apps access photos and how users store photos on their mobile phones. In addition, the system should not affect the usability of apps and user experience; i.e., the access control enforcement decision must be made within a reasonable amount of time.

Our basic idea is that when an app requests to access a particular photo, users should be aware about it and decide whether the app can access that photo. The naive approach is to prompt users to check the photo and make a decision every time. However, this will definitely degrade the usability of the system and apps. To address this problem, PhotoSafer is designed to be able to automatically check whether the content of photo is private, and determine whether the user is aware of the app’s access request based on the contextual status of the phone and the app. Specifically, when the phone is locked, the user is not operating the phone for photo access and thus most likely does not know that an app is accessing the photo. Even when the phone is unlocked, the app which requests access to the photo might be running in the background. In this case, the user probably also does not know that the app is accessing the photo. Generally, a user is aware of the photo access if the app is running in the foreground, since usually the access request is triggered by the user in this scenario. In this case, the system can automatically check the photo content through a trained classifier and inform the user whether the photo contains private information and what private information it is. The system also allows the user to determine whether the access request should be permitted. To minimize the time needed to identify private photo
content during user operation, PhotoSafer caches the contents of photos in a database in advance. In this way, PhotoSafer can achieve real-time response to photo access, such that the requesting app’s usability and user experience will not be affected. To make the system work, PhotoSafer needs to be integrated into the Android kernel as a system service, since it needs privileges to interpose photo access. When PhotoSafer is initialized, it will feed all stored photos into a trained classifier to identify photo contents (e.g., photo ID), and then the result will be stored in a database. Whenever a new photo is added, it will be fed into the classifier and the classification result will be updated into the database.

The workflow of PhotoSafer is shown in Figure 5.1. When an app requests to access a specific photo, the photo access will be interposed and system status will be checked. If the phone is locked, then the access request will be automatically denied. However, if the phone is unlocked, the system will continue checking the app’s status. If the app that requests photo access is running in the background, then the access request will also be automatically denied. On the contrary, if the app is running in the foreground, the photo content can be immediately obtained by querying the database, where the content type of each photo is stored. Finally, if the photo is classified as public, the access permission will be automatically granted. Otherwise, an alert will be prompted by informing the user of what private information is contained in the photo and requesting the user to determine permission. If the user trusts the app and grants permission to it, then the photo access will be continued; otherwise it will be denied.

5.3.2 Architecture

As Figure 5.2 shows, the system consists of four major modules: photo access interposition, status checker, photo content classifier and photo content database. We can divide the overall workflow of PhotoSafer into three steps. First, when the system is initialized
Figure 5.1: The workflow of PhotoSafer

on a phone, the pre-trained photo content classifier performs classifications on all stored photos, and the results will be stored in the photo content database. In the database, each record consists of a tuple \((\text{photo}\_\text{id}, \text{content}\_\text{type})\), where \text{photo}\_\text{id} is the unique identifier in each photo’s universal resource identifier (e.g., content://com.android.providers.media.documents/document/image\%photo\_id) in Android and \text{content}\_\text{type} represents whether a specific photo is not private or contains which specific type of private content, such as (‘10001’, ‘public’) and (‘10002’, ‘photo ID’). Then, when an app requests to access a specific photo, the photo access interposition module will interrupt the app’s operation and trigger the status checker module to check the system status and the app’s current running status. If the phone is either locked or the app is running in background, the access request will be directly denied. Otherwise, a query with the photo’s \text{photo}\_\text{id} will be sent to the photo content database. Finally, if the returned result from photo content database shows the photo
is public, the access request will be automatically permitted, and photo access interposition will resume the app’s operation. However, if the photo contains some private information, an alert will be prompted by describing what private information is contained in the photo and requesting the user to make the decision of whether the photo access interposition will resume the app’s operation.

![Figure 5.2: The architecture of PhotoSafer](image)

The design of PhotoSafer is based on several technologies available in off-the-shelf mobile phones. Photo access interposition can be done by modifying the Android kernel. The status checker can be implemented by Android APIs `KeyguardManager` and `ActivityManager`. The photo content database can be implemented by using SQLite. Next, we will describe how to identify private photos.

### 5.3.3 Photo Content Classifier

Given an input photo in pre-defined dimensions, this module aims to detect whether that photo contains some specific private information. This is done by training a deep convolutional neural network (DCNN) to detect the private content of photos.

Formally, let $\mathbb{P}$ be the set of input photos. For a given photo $x \in \mathbb{P}$, let $y \in \{1, 2, 3, 4, 5\}$ encode the categorical labels ‘public’, ‘photo id’, ‘legal document’, ‘family’, ‘nude’ of the photo. Let $\mathbb{H}$ be the hypothesis space of possible decision functions, and
\(f(\theta^T x)\) be the decision function, where \(\theta = \{\theta_1, \theta_2, \ldots, \theta_N\}\) is the network weights. Hence, the loss function can be defined as \(L(f(\theta^T x), y)\). Let \(\mathbb{E}(L)\) be the expected loss over the range of inputs \(P\). In this work, we use cross-entropy to estimate the loss, and hence the optimization task is to minimize the expected cross-entropy loss.

\[
f = \arg \min_{f \in H} \mathbb{E}(L) \tag{5.1}
\]

For each input \(x\), the corresponding classification result is \(f(x)\), and hence the according accuracy \(acc(x)\) can be defined as:

\[
acc(x, y) = \begin{cases} 
+1 & y = f(x) \\
0 & \text{otherwise}
\end{cases} \tag{5.2}
\]

However, the main challenge for training a DCNN to identify private photos is acquiring a sufficient number of private photos to train on. Generally, a DCNN requires a relatively large set of training data to perform well. To address this challenge, we adopt the transfer learning [16] approach to train our DCNN model. Specifically, we pretrain a DCNN model on a large dataset ImageNet [17], which contains 1.2 million images with 1000 categories. Then, we tune the parameters of the output layer in the pretrained model on the small number of private photos while keeping the parameters of all the other layers unchanged.

5.4 Implementation

Due to the time limitation, we implemented PhotoSafer as a standalone app on Android phones instead of integrating it into the Android kernel. However, the current implementation will not affect the evaluation for our proposed scheme, and we plan to implement its integration with Android kernel in future work. Generally, the app works like the native
photo gallery app that comes with the Android system. The prototype app was specifically designed so that it will access some private photos under different system status and app-running status. The photo content classifier was implemented using Python 2.7 and Tensorflow [18], which is an open-source deep learning framework. The other modules were implemented by available technologies, as mentioned in Section 5.3.2.

5.4.1 Photo Content Classifier

This module aims to identify whether a given photo is public or contains some specific private information. As described in Section 5.3.3, we use transfer learning to train a DCNN model. In particular, we build the classifier using the Python APIs of Tensorflow and adopting MobileNets [19]. The MobileNets are a class of DCNNs that are specifically designed for efficiently running on mobile devices. The significant difference between the MobileNets architecture and a traditional DCNNs is that instead of a single 3x3 convolution layer followed by batch norm and rectified linear unit (ReLU), MobileNets split the convolution into a 3×3 depthwise convolution layer and a 1×1 pointwise convolution layer. It has been demonstrated that the computing operations and model size will be significantly reduced in this way. MobileNets are usually not as accurate as traditional DCNNs, but it provides a trade-off between accuracy and resource usage. Specifically, MobileNets offer two parameters to tune the resource and accuracy trade-off: width multiplier and resolution multiplier. The value of width multiplier should be set between 0 and 1, while the resolution multiplier might be various. The width multiplier allows us to adjust the thickness of the DCNN, and the resolution multiplier changes the input dimensions of images, which can reduce the internal representation complexity at every layer. Table 5.5 shows that given a fixed resolution multiplier, with the increase of width multiplier the number of computing operations and parameters will dramatically increase. However, when the width multiplier
is fixed, the larger the input dimension, the more the required computing operations.

Table 5.5: MobileNets with Different Width Multipliers

<table>
<thead>
<tr>
<th>Width Multiplier</th>
<th>ImageNet Accuracy</th>
<th>Million Operations of Mult-Add</th>
<th>Million Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNet_1.0_224</td>
<td>70.6%</td>
<td>569</td>
<td>4.2</td>
</tr>
<tr>
<td>MobileNet_0.75_224</td>
<td>68.4%</td>
<td>325</td>
<td>2.6</td>
</tr>
<tr>
<td>MobileNet_0.5_224</td>
<td>63.7%</td>
<td>149</td>
<td>1.3</td>
</tr>
<tr>
<td>MobileNet_0.25_224</td>
<td>50.6%</td>
<td>41</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 5.6: MobileNets with Different Resolution Multipliers

<table>
<thead>
<tr>
<th>Resolution Multiplier</th>
<th>ImageNet Accuracy</th>
<th>Million Operations of Mult-Add</th>
<th>Million Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNet_1.0_224</td>
<td>70.6%</td>
<td>569</td>
<td>4.2</td>
</tr>
<tr>
<td>MobileNet_1.0_192</td>
<td>69.1%</td>
<td>418</td>
<td>4.2</td>
</tr>
<tr>
<td>MobileNet_1.0_160</td>
<td>67.2%</td>
<td>290</td>
<td>4.2</td>
</tr>
<tr>
<td>MobileNet_1.0_224</td>
<td>64.4%</td>
<td>186</td>
<td>4.2</td>
</tr>
</tbody>
</table>

In this work, we fix the input dimension as $224 \times 224$, but change the width multiplier for comparisons in Section 5.5.2. Firstly, we train the MobileNets on ImageNet with fine-tuning parameters. After that, we fine-tune the output layer of pretrained model with our collected dataset of private photos, but keep parameters of the other layers unchanged. The details of the dataset are described in Section 5.5.1.

5.5 Evaluation

In this section, we systematically evaluate the performance of PhotoSafer through a number of experiments. To better illustrate the benefits provided by our proposed system, we also make clear comparisons against existing approaches. Particularly, we conduct the following experiments. Firstly, we conduct extensive experiments to measure the private photo identification accuracy. Secondly, we test the time taken by the system to obtain photo content classification results from the database. The evaluations for classification accuracy are made on Ubuntu 17.04 64-bit machine with 32G RAM and one NVIDIA TITAN Xp GPU. The other experiments are conducted on Nexus 5 phones.
5.5.1 Dataset

Since private photos that are shared on a public domain are limited, it is a challenging task to collect private photos for training deep learning models. Furthermore, there is no standard definition of ‘private photo’ applicable for every user, since it is a very subjective determination. Thus it is hard to collect one private photo dataset to cover all cases. However, Zerr et al. [20] published a dataset collected from Flickr, which is the only publicly available dataset for photo privacy research at this time. This dataset consists of 37,535 photos, which are labeled as Private, Public and Undecided. Since the private photos in this dataset do not include most of the private types reported from our survey, we only use the Public photos from this dataset as public photos in our dataset. Additionally, we collect 3,097 private photos in four common types as shown in Table 5.3 from Google Image, with some example photos shown in Figure 5.3. 80% of the dataset is used for training, and the remained 20% is used for testing. The distribution of each type of photo is illustrated in Table 5.7.

<table>
<thead>
<tr>
<th>Photo Type</th>
<th>Number of Photos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photo ID (e.g., driver license, passport)</td>
<td>1353</td>
</tr>
<tr>
<td>Legal Documents (e.g. SSN)</td>
<td>469</td>
</tr>
<tr>
<td>Family (e.g., family party)</td>
<td>682</td>
</tr>
<tr>
<td>Nudes</td>
<td>543</td>
</tr>
<tr>
<td>Public</td>
<td>14664</td>
</tr>
</tbody>
</table>

5.5.2 Classification Accuracy

As described above, we trained MobileNets models with a fixed input dimension of 224 × 224 but with different width multipliers. To be specific, we set the width multiplier as 1.0, 0.75 , 0.5 and 0.25 separately. To compare the classification accuracy, we also compare our MobileNets model with an Inception_v3 model [22] that is trained on the same dataset.
Figure 5.3: Example photos in our dataset. Sensitive information are removed from the photos.
In addition, all the models are trained in two ways. Firstly, we directly train each model with our dataset. Secondly, we adopt transfer learning to train each model; i.e., we firstly train each model with ImageNet and then fine-tune the model with our dataset.

As shown in Table 5.8, the classification accuracy of each model that is trained in transfer learning is higher than that of each directly trained model. In particular, we observe that the accuracy improves between 12% and 17%. It also shows the Inception_v3 model has a slightly higher accuracy than the MobileNet_1.0_224 model, but the model size is much bigger than the MobileNet_1.0_224 model. This means it requires much more computation resources for only a little performance improvement, which is not a good fit for resource-constrained mobile phones. In addition, with respect to MobileNets models, with the decreasing width multiplier the classification accuracy becomes lower and the model size is smaller. Based on above comparisons, we choose the MobileNet_1.0_224 model with transfer learning as our final classifier due to its high classification accuracy and reasonable model size.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Model Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception_v3</td>
<td>80.3%</td>
<td>87.4 MB</td>
</tr>
<tr>
<td>MobileNet_1.0_224</td>
<td>77.5%</td>
<td>17.1 MB</td>
</tr>
<tr>
<td>MobileNet_0.75_224</td>
<td>72.6%</td>
<td>10.5 MB</td>
</tr>
<tr>
<td>MobileNet_0.5_224</td>
<td>68.7%</td>
<td>5.5 MB</td>
</tr>
<tr>
<td>MobileNet_0.25_224</td>
<td>63.3%</td>
<td>2 MB</td>
</tr>
<tr>
<td>Inception_v3+Transfer Learning</td>
<td>97.3%</td>
<td>87.4 MB</td>
</tr>
<tr>
<td>MobileNet_1.0_224+Transfer Learning</td>
<td>94.3%</td>
<td>17.1 MB</td>
</tr>
<tr>
<td>MobileNet_0.75_224+Transfer Learning</td>
<td>93.5%</td>
<td>10.5 MB</td>
</tr>
<tr>
<td>MobileNet_0.5_224+Transfer Learning</td>
<td>91.2%</td>
<td>5.5 MB</td>
</tr>
<tr>
<td>MobileNet_0.25_224+Transfer Learning</td>
<td>89.7%</td>
<td>2 MB</td>
</tr>
</tbody>
</table>

As shown in Table 5.9, we also evaluate the classification accuracy for each type of private photo. It can be seen that our deep learning model achieves higher classification accuracy for each type of private photo than that of the baseline model. For instance, the classification accuracy of ‘Photo ID’ and ‘Legal Document’ are as high as 97.8%.
Table 5.9: Classification Accuracy for Each Type of Photo

<table>
<thead>
<tr>
<th>Model</th>
<th>Photo ID</th>
<th>Legal Document</th>
<th>Family</th>
<th>Nude</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNet_1.0_224</td>
<td>97.8%</td>
<td>97.8%</td>
<td>95.6%</td>
<td>86.2%</td>
<td>94.3%</td>
</tr>
</tbody>
</table>

In Table 5.10 we further explore the misclassifications. Even though there are a small number of misclassifications on each type of private photo, none of these are mistakenly classified as ‘Public’. That means although there exist such misclassifications on some photos, this will not prevent the PhotoSafer from prompting alerts to user.

Table 5.10: Confusion Matrix of Private Photo Classification

<table>
<thead>
<tr>
<th>Actual</th>
<th>Photo ID</th>
<th>Legal Document</th>
<th>Family</th>
<th>Nude</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photo ID</td>
<td>265</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Legal Document</td>
<td>2</td>
<td>92</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Family</td>
<td>4</td>
<td>0</td>
<td>130</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Nude</td>
<td>5</td>
<td>0</td>
<td>10</td>
<td>94</td>
<td>0</td>
</tr>
</tbody>
</table>

5.5.3 Photo Content Classification Time

As presented in Section 5.3, in order to avoid affecting user experience, we store the classification results of all stored photos in a photo content database. We measure the time for retrieving one record of a specific photo from the database, compared with the time for making a classification of that photo in real time. We run a total of 10 trials for each of 100 randomly selected photos, and the average time cost is described in Table 5.11. It shows the time cost of the database-based approach is 38 time less than that of running real-time classifications.

Table 5.11: Time Cost for Obtaining Photo Content Classification Result: From Database vs. Real-Time Classification

<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Database</td>
<td>5.2 ms</td>
</tr>
<tr>
<td>Real-Time Classification</td>
<td>190.7 ms</td>
</tr>
</tbody>
</table>
5.6 Discussion

Even though we implemented a prototype system, we did not integrate it into Android kernel as originally designed. This section discusses the limitations of the current design and implementation, and how it can be improved in the next step of research.

**Kernel Interposition.** In our current prototype system, we implemented a function to simulate apps that may access photos under different system status and app-running status in the real world. This is the only way the prototype can interpose photo access and monitor which specific photo is being accessed. Otherwise, the prototype system requires the root privilege in the Android system, which is not safe for user to install such software. However, as described in Section 5.3, the best way is to implement photo access interposition in the Android kernel. Since photos are accessed as regular files in Android, all file access should be interposed. Additionally, the kernel interposition needs to determine if the accessed file is a photo through checking file extension (e.g., .jpg), so that the system can decide in the kernel whether the accessed file needs our proposed access control.

**App Whitelist.** In the current system design, we determine whether a photo access by a specific app is ‘unauthorized’ based the system status and app-running status, which can cover most cases. However, in some cases, users are satisfied with some apps that are running in the background access photos. For instance, some users allow the Google Photos app to backup the stored photos even if it is running in the background. To consider such cases, the current system design can be improved by adding an app whitelist. It allows users to specify which apps can be granted access to all stored photos without the proposed access control.
5.7 Related Work

**Android Permissions** The Android platform offers users two approaches to control permissions. Before Android 6.0 (Marshmallow), apps are required to disclose the full list of resources that an app wants to access at installation. Users must grant all requested permissions, otherwise the installation will be aborted. Some work [2, 23] has shown that few users pay attention to and really understand the meaning of install-time permissions. After Android 6.0, users need to grant permissions only when an app requests a sensitive resource for the first use. This scheme can offer users contextual clues about why the requested resource is necessary for an app. However, it does not account for the fact that the user’s preference for subsequent permission requests might be varied under different contextual circumstances.

Work has been done on permission models [2, 24, 25, 26, 27], which found users usually do not know how apps access sensitive resources and how such access is managed. Shih et al. [28] showed that privacy information is more likely to be leaked when users are unaware of the purpose for requesting a specific sensitive resource.

Almuhimedi et al. analyzed AppOps, which is a permission manager introduced in Android 4.3 but removed in Android 4.4.2 [29]. AppOps allowed users to review and modify app permissions after apps were installed. They provided both qualitative and quantitative evidence that the permission manager can increase users’ awareness of privacy risks. Even though AppOps was removed from Android, a new permission management system was introduced in Android 6.0, which allows users to review all permissions that an app has been granted. However, since it is hidden in the deep level of the **Settings** app, it is not easy for average users to discover it. There exist several third-party permission management apps, such as XPrivacy [30], DonkeyGuard [31], Permission Manager [32] and Privacy Guard [33].
However, these apps require additional privileges to support their functionalities, since there is no official approach offered to third-party apps to modify the permission system. For example, XPrivacy requires an unlocked bootloader and a custom recovery partition. Such restrictions are needed to protect the permission system against interfering by malicious apps.

**Photo Privacy**  Ra et al. [34] designed a system P3 to protect privacy of photos when they are shared on online social networks. He et al. [35] proposed an approach to protect users’ privacy for photo sharing. Jana et al. [36] proposed a system Darkly based on OpenCV library, which protect users’ private information from continuously-sensing applications. Templeman et al. [37] implemented a system PlaceAvoider to protect visual privacy by identifying sensitive places in video streams. Tan et al. [11] designed an access control scheme to protect private photos on mobile phones, but it only depends on pre-stored faces on mobile phones, which can only provide limited protection. However, PhotoSafer can protect broad types of private photos. Zerr et al. [20] collected a photo dataset from Flickr with labeled as public, private or undecided. Then, they extracted low-level features and trained a SVM model to identify private photos. Squicciarini et al. [38] et al. conducted an extensive study based on Flickr dataset collected by Zerr et al., and developed learning models to estimate adequate privacy settings for shared photos in online social networks. Similar to work by Zerr et al., Liu et al. [39] recruited workers to label photos collected from Facebook as share with “only me”, “some friends”, “all friends”, “friends of friends” and “everyone”. They found that there is a big difference between the actual labels on Facebook and labels obtained from workers. The difference is due to Facebook users usually sharing photos using the default privacy setting. Consequently, the default privacy setting on Facebook is much lower than the privacy protection that users desire.
5.8 Summary

Motivated by a user survey and analysis of 200 apps’ permission requests, both of which showed the potential risk of private photos being leaked without being known, we proposed a system PhotoSafer to protect private photos that are stored on mobile phones from being accessed without users’ awareness. The access control on those private photos is enforced by checking the system status and photo content. A mobile-compatible private photo classifier was designed with transfer learning. We implemented a prototype system, and evaluated its performance and cost through experiments.

Bibliography


6 Conclusions and Future Work

6.1 Conclusions

In this dissertation, we proposed a set of systems to protect people’s privacy in photo taking and accessing.

In Chapter 2, we proposed a mobile cooperative privacy protection system, called PrivacyCamera, to protect the privacy of a stranger who is accidentally included in a photo taken by mobile phones. The system can notify nearby strangers of the possible inclusion in a photo and allow them to decide if to blur their faces in the photo. We designed techniques to detect if a stranger requesting face blurring in the photo or not based on GPS locations. We implemented a prototype system on Nexus 5 smartphones, and evaluated the system’s performance and cost through experiments as well as field tests. Evaluations show that the system can accurately detect the stranger and blur his face to protect his privacy with low computation cost and power consumption.

In Chapter 3, we addressed the same privacy issue of protecting strangers’ privacy in photo taking as in Chapter 2. Different from PrivacyCamera, we use facial attributes to determine whether a stranger is in a photo or not instead of using GPS locations. Since the selected facial attributes are relatively stable, if a person is in a photo, by comparing the faces in the photo with his recent profile photo in facial attributes, the person can be correctly matched to his face in the photo. Also, in photographing scenarios, it is not very likely that the facial attributes of two nearby strangers are exactly same, since the number of persons in a limited geographic area around the photographer is usually not large. Based on facial attributes, we proposed a cooperative scheme PoliteCamera to protect the privacy of strangers who are unintentionally included in photos taken by mobile phones. We designed
a novel ABCNN structure to predict facial attributes. We implemented a prototype system on Nexus 5 smartphones, and evaluated its performance through experiments. The accuracy of the facial attributes prediction is better than the state of the art, and experimental evaluations demonstrate that the system can effectively protect strangers’ privacy.

In Chapter 4, we proposed a feature-based model to distinguish the target from the strangers in a photo. Based on this model, we can protect strangers’ privacy by automatically detecting them and blurring their faces in the photo, and hence there is no need for the cooperation between the photographer and the strangers compared with the solutions in Chapter 2 and Chapter 3. We implemented the model based on different supervised learning algorithms and compared their performance. Evaluations show that the feature-based model can effectively tell the target from the stranger in a photo, and its performance is better than that of the heuristic rule base approach. In addition, we explored how feature selections affect the performance.

In Chapter 5, motivated by a user survey and analysis of 200 apps’ permission requests, we designed a novel content-based context-aware private photo protection system PhotoSafer, which provides real-time access control over private photos based on the photo contents and context status of mobile phone. The specific sensitive information that a private photo contains will be disclosed to users before the access is granted. A resource-friendly private photo classifier was trained with transfer learning. The evaluations demonstrated that PhotoSafer can provide reliable protection for private photos against unauthorized access.

6.2 Future Work

This dissertation proposed several systems to provide privacy protection for photo taking and accessing over mobile phones. Besides collecting more datasets to better validate the proposed systems, there are still many other issues that deserve further exploration. In
the following, we discuss two future research directions.

- **Privacy-Preserving Online Photo Sharing**: Although we proposed several systems to protect privacy in photo taking and accessing, we have not addressed privacy issues in online photo sharing. Currently, photo sharing-supported online social networks (Instagram, Flicker, Facebook or WeChat, etc.) are very popular, and many people share their photos on them. However, when a user shares a photo that includes his friends and strangers, the included friends and strangers may not want to be published online due to their own privacy concern. Some works have been done to address this issue [1, 2, 3, 4], but they require users to set a privacy policy for each photo, or do not consider the privacy of strangers. Therefore, it is necessary to design a privacy-preserving online photo sharing system, which has less burden of policy settings and can protect the privacy of strangers in a photo.

- **Personalized Privacy-Preserving Photo Accessing**: Even though we have designed the PhotoSafer to protect private photos from unauthorized access, the current design can only protect several common types of private photos. Since different users have different categorization of private photos, one research direction is to design personalized systems to protect user-specific private photos.

**Bibliography**


A Appendix

A.1 IRB Approval

MEMORANDUM

TO: Ang Li  
Ryan Graham  
Qinghua Li

FROM: Ro Windwalker  
IRB Coordinator

RE: New Protocol Approval

IRB Protocol #: 16-10-159

Protocol Title: Access Control to Protect Sensitive Photos on Smartphones

Review Type: ☑ EXEMPT ☐ EXPEDITED ☐ FULL IRB

Approved Project Period: Start Date: 10/27/2016 Expiration Date: 10/26/2017

Your protocol has been approved by the IRB. Protocols are approved for a maximum period of one year. If you wish to continue the project past the approved project period (see above), you must submit a request, using the form Continuing Review for IRB Approved Projects, prior to the expiration date. This form is available from the IRB Coordinator or on the Research Compliance website (https://vpred.uark.edu/units/rscp/index.php). As a courtesy, you will be sent a reminder two months in advance of that date. However, failure to receive a reminder does not negate your obligation to make the request in sufficient time for review and approval. Federal regulations prohibit retroactive approval of continuation. Failure to receive approval to continue the project prior to the expiration date will result in Termination of the protocol approval. The IRB Coordinator can give you guidance on submission times.

This protocol has been approved for 50 participants. If you wish to make any modifications in the approved protocol, including enrolling more than this number, you must seek approval prior to implementing those changes. All modifications should be requested in writing (email is acceptable) and must provide sufficient detail to assess the impact of the change.

If you have questions or need any assistance from the IRB, please contact me at 109 MLKG Building, 5-2208, or irb@uark.edu.