CARDEA: A CONTEXT-AWARE AND INTERACTIVE VISUAL PRIVACY CONTROL FRAMEWORK

by

RUI ZHENG

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in Computer Science and Engineering

October 2016, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

RUI ZHENG

CARDEA: A CONTEXT-AWARE AND INTERACTIVE VISUAL PRIVACY CONTROL FRAMEWORK

by

RUI ZHENG

This is to certify that I have examined the above M.Phil. thesis

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by

the thesis examination committee have been made.

ASSISTANT PROF. PAN. HUI, THESIS SUPERVISOR

PROF. QIANG YANG, HEAD OF DEPARTMENT

Department of Computer Science and Engineering

21 October 2016

ACKNOWLEDGMENTS

First I would like to thank my family for their unconditional support during the past three years. It is their optimistic attitudes as well as patience that helped me walk across all the troubled water among the years.

I would also like to express my deepest gratitude to my advisor Prof. Pan Hui for his support, guidance and encouragement, without which this thesis would not have been possible.

Special thanks to Jiayu Shu, it was a great pleasure to collaborate with her in the past year and I learned a lot from the collaboration. Same thanks gives to Dr. Tongfeng Weng for the enjoyable collaboration on random walks. Other than that, I would like to thank Haris Mughees and Hamza Zia, for countless times we happily talked about everything and I was always surprised by their knowledge, passion as well as determination.

Thanks also goes to all the members in Symlab family. I am fortunate to know so many wonderful people and work as colleague with them. They have helped me a lot in many aspects from daily life to research. Many thanks to Mr. Issac Ma and Mrs. Connie Lau for their administration work.

Last but not least, I would like to thank Prof. Dit-Yan Yeung and Prof. Chi-Keung Tang. It was great fun to take their courses, which guided me to find my interests and thesis topic. I deeply admire the strong sense of responsibility they have on both lecturing as well as supervision of their students. Their research attitudes and hard working will keep motivating me in my future endeavors.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CARDEA: A CONTEXT-AWARE AND INTERACTIVE VISUAL PRIVACY CONTROL FRAMEWORK

by

RUI ZHENG

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

The growing popularity of mobile and wearable devices with builtin cameras, the bright prospect of camera related applications such as augmented reality and lifelogging system, the increased ease of taking and sharing photos, along with advances in computer vision techniques, have greatly facilitated peoples lives in many aspects, but inevitably raised peoples concerns about visual privacy at the same time.

Motivated by the finding that peoples privacy concerns are influenced by the context, in this thesis, we propose Cardea, a context–aware and interactive visual privacy control framework that enforces privacy policies according to peoples privacy preferences. The framework provides people with finegrained visual privacy control using: *i)* personal privacy profiles, with which people can define their context–dependent privacy preferences; *ii)* natural visual indicators: face features, for devices to automatically locate individuals who request privacy protection; *iii)* hand gestures, for people to temporarily update and flexibly inform cameras of their privacy preferences. Benefited

from recent progresses in face and object recognition, Cardea offers a way for context–dependent privacy control in a natural and flexible manner, which differs from tag and marker based systems. We design and implement the framework consisting of Android client app and cloud control server, with convolutional neural networks as core of the image processing module. Our evaluation results confirm such framework is practical and effective, showing promising future for context–aware visual privacy control on mobile and wearable devices.

CHAPTER 1

CARDEA

1.1 System Design

Recalling related works in Chapter 2, what motivates the design of Cardea are the following:

- People's privacy concerns are dependent on context. Although in certain circumstances locations are strong hints of possible privacy intrusion, generally what individuals are doing and with whom are more essential and crucial factors that directly relate to privacy.
- People's privacy preferences vary from each other, thus they should be able to express their personal privacy preferences.
- People's privacy preferences may change from time to time, therefore they need a way to change such preferences easily.

To achieve these objectives, we propose following solution:

- We combine GPS location, grouped scene categories (Table 1.1) and accompanied persons as context that can better represent people's privacy concerns than previous methods. As a result we are able to provide more general as well as finer granularity privacy preference settings for users.
- We use cloud server to host individualized privacy preferences, and user's preference is bound with his facial features. In this way his raw visual information stays locally, preventing the case of visual privacy leakage from hacked servers.
- Other than a simple interface provided to users to update their privacy preferences, hand gestures like  and can be used by user to actively speak out about his preference in the capturing moment, enriching the interaction and adding more flexibilities.

As introduced in chapter 3, breakthroughs made by deep learning community in many computer vision problems such as image classification, face recognition and object detection have guided the proposed solution and shed lights on its practicability. More specifically, given a captured image, Cardea will leverage powerful convolutional neural networks for the recognition of scene context, registered users and gestures. Cardea's design is given in Fig 1.1, it is composed of the client applications and cloud server. It works based on data exchange and collaborative computing involving both client and cloud sides. The major components and interactions include:

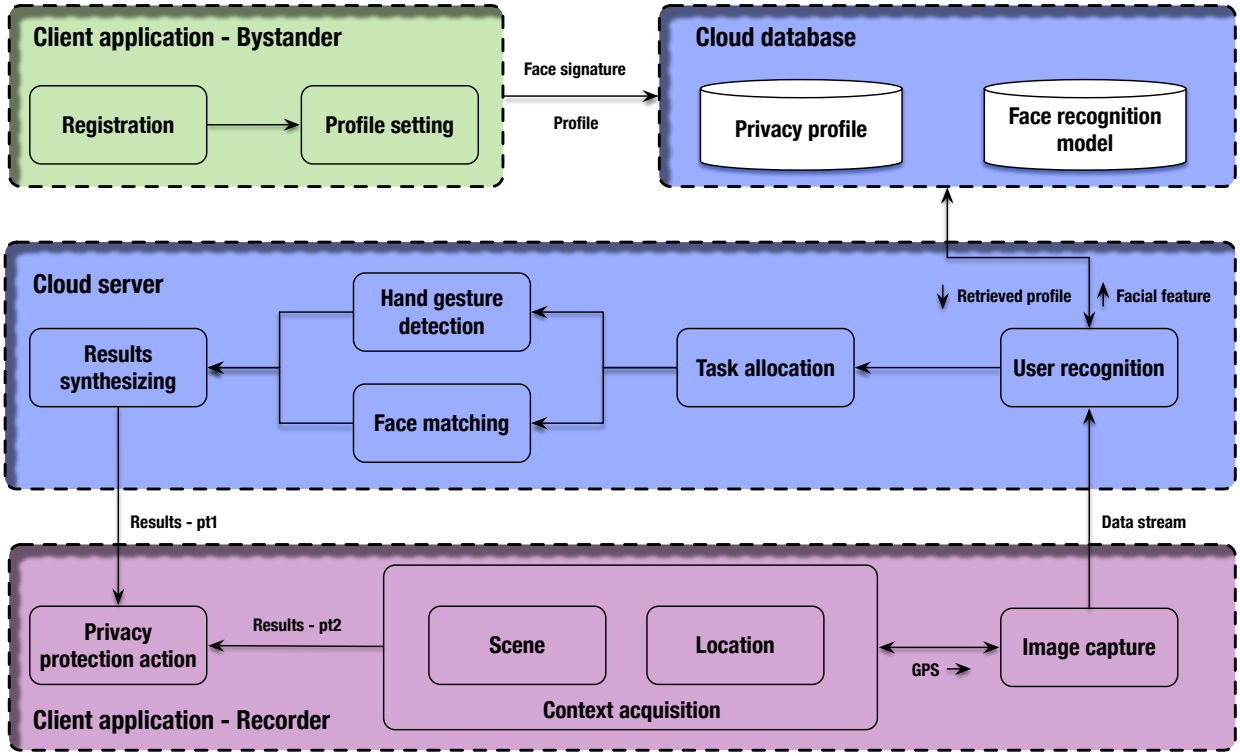

Figure 1.1: System design of Cardea.

Bystander client application: A bystander can use this application to register as Cardea user and define his privacy profile. It will capture a number of face images (about 50–60 images) and extract facial features from these images as his unique face signature. After setting up the context dependant privacy preference, his facial signature and preference will be sent to cloud server for registration and updating of face recognition model.

Recorder client application: A recorder can use this application to take images that will automatically perform privacy protection actions in compliance with all Cardea users' privacy preferences. Given a captured image, it first detects all the faces and extracts the corresponding facial features locally on device, then the features and the captured image (compressed and with all detected faces blurred) are sent to the server for face and gesture recognitions. GPS coordinate is also sent to server in this step to be compared with recognized users' location settings. During the time waiting for response from cloud, it performs scene group prediction task. Finally, predicted scene group and intermediate decision result received from server are combined to decide the actual protection action which will be enforced on the raw captured image.

Cloud server: The cloud server plays two roles: ❶ When receiving requests from Bystander applications, it will store/update users' profiles, and training/updating system's face recognition model automatically; ❷ When receiving requests from Recorder applications, it will initiate face and gesture recognition tasks, as well as partial decision making based on recognition results, and send these intermediate decision results to client for the final step of decision making.

Implementations and evaluation of each module, how Cardea allocates tasks between mobile and cloud, integration and user interactions are discussed in following sections.

1.2 Model Training

1.2.1 Scene Classification

Data Preparing and Preprocessing

For scene classification, we use pre-trained model of Places2 dataset provided by [1]. In the time Cardea project was conducted, Places2 dataset provided by the authors contained 401 categories with more than 8 million training images, and the pre-trained model was based on AlexNet structure [2]. By the time this thesis is writing, the dataset is deprecated and the new Places2 dataset contains 365 categories. And the authors provide more pre-trained models based on different network structures [3].

Table 1.1: Scene categories.

Scene Group	Scene Category
Eating	bistro/indoor, bistro/outdoor, cafeteria, coffee_shop, diner/outdoor, dining_hall, dining_room, fastfood_restaurant, food_court, restaurant, restaurant_patio, sushi_bar
Entertainment	bar, discotheque, pub/indoor
Shopping	bazaar/indoor, bazaar/outdoor, clothing_store, general_store/indoor, jewelry_shop, shoe_shop, shopping_mall/indoor, supermarket
Work	conference_center, conference_room, cubicle/office, library/indoor, office, office_cubicles, reading_room
Public	park, street
Mobility	airplane_cabin, airport_terminal, bus_interior, bus_station/indoor, subway_station/platform, train_interior, train_station/platform
Exhibition	art_gallery, museum/indoor
Religion	cathedral/indoor, cathedral/outdoor, church/indoor, church/outdoor, mosque/outdoor, pulpit, temple/east_asia, temple/south_asia
Illness	hospital, hospital_room, nursing_home
Nudity	bathroom, beach, jacuzzi/indoor, sauna, shower, swimming_pool/indoor, swimming_pool/outdoor

Note that in the dataset we used, there is a non-uniform distribution of images per category for training, ranging from 4,000 to 30,000, mimicking a natural frequency of occurrence of the scene. Among the 401 categories, we choose 59 scene categories that are close to daily life and in such scenes people may have privacy concern. In total this subset composed of 1 million training images and 2950 validation images (50 validation images for each category). We also group these 59 scene categories into 10 groups based on contextual similarity as shown in Table 1.1, such that people have similar reasons for privacy concerns in scenes that are in the same group (e.g. people don't want to be captured in bathroom and beach is both because of nudity concerns). The distribution of training images among categories and groups is shown in Fig 1.2.

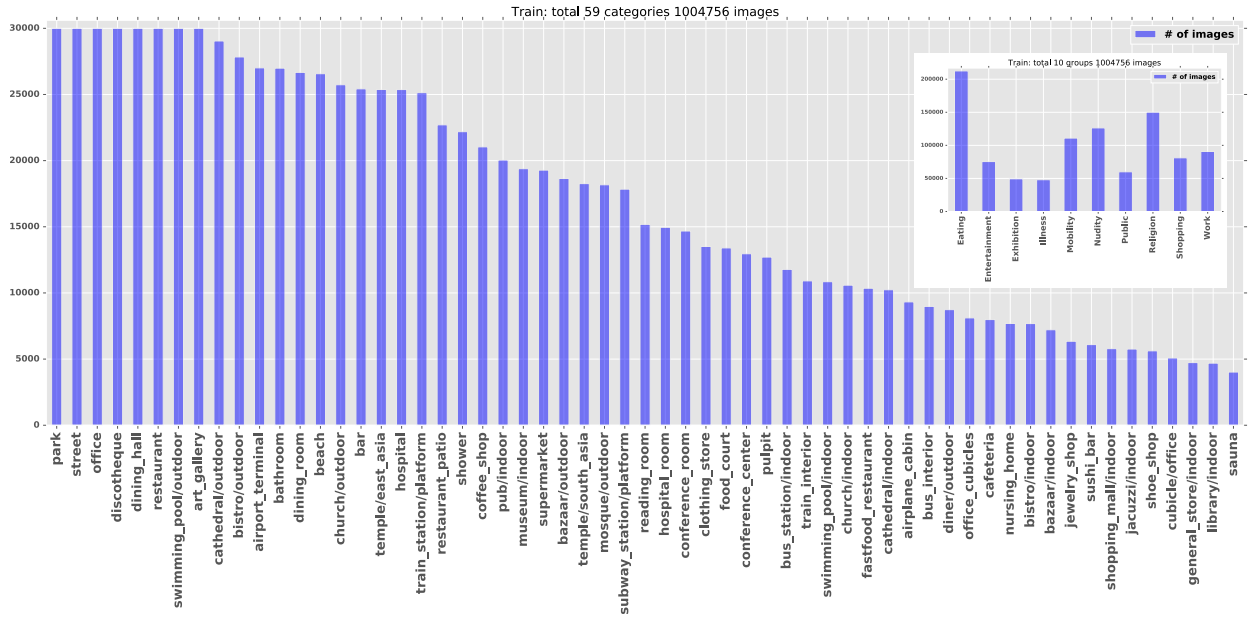

Figure 1.2: Number of images for each category and each group (inset).

Training Procedures

The training step is a standard fine-tuning process, which is extensively used in transfer learning [4, 5]:

❶ Using pre-trained model as feature extractor, we extract the features at *fc7* layer for images belonging to the 59 picked categories. Other than shuffling the features, we also augment the features such that all categories have same amount of features. Though the natural frequencies of occurrence are obviously different among different scenes, we argue that for the purpose of privacy protection, all the scene categories should be equally important, thus categories imbalance is not what we favored. The augmentation step can be implemented using weighted loss layer, but we take simple way of bootstrapping features for categories with less images. After this step, all features are cached and stored in lmdb format.

❷ Train a softmax classifier of the 59 categories using the extracted features. We choose to train a classifier for categories and then add up the output probabilities to predict the group, rather than directly train a group classifier, is because category classifier tells more about the

image, and our desired property is equal weights among scene categories rather than groups.

❸ Both feature extraction and classifier training are implemented using Caffe library [6, 7]. In this step we merge the feature extraction part of pre-trained model and the softmax classifier into a single model by copying weights. Now Caffe has the option of specifying layers with fixed weights, thus simplifying the fine-tuning and deployment process.

Other than improving the validation accuracy from 0.56 to 0.57, shuffling also makes training converges faster. With augmentation to relieve category imbalance issue, the classifier can finally achieve 0.600 validation accuracy on the 59 categories. There is no other benchmarking result specifically on the subset we choose, but recent benchmark gives 53%-56% validation accuracy on the new Places2 dataset with 365 categories [3], suggesting our model is competitive. The higher validation accuracy of our model is due to the smaller scale of classification problem we are dealing with.

Prediction

For prediction, we get probability of a group by summing up the probabilities of all categories belonging to this group, and output the most probable group as prediction of an image. Our model's group prediction accuracy for the validation set is 82.8%. Fig 1.3 shows some prediction examples. As seen from the examples, given an image, the predicted category probabilities are usually distributed to few categories within same group, thus group prediction is resilient to perturbation coming from category prediction. The way we group categories can be seemed as a hard-coded clustering step, which makes prediction more robust to noise. The failure cases are mostly due to natural context ambiguity from a image (e.g. image with object in focus, therefore not enough hints for scene inference). Labeling the 342 non selected categories as an extra group will amplify the ambiguity issue, as doing so will distribute probabilities to the extra group and lead to wrong prediction, even for images with less ambiguity. In other words, a 59 way classifier leads to higher recall for selected scenes and grouping leads to higher accuracy. This is also reflected in confusion matrices shown in Fig 1.4, category confusion matrix shows some clustering structure which is in accordance with the groups we manually assigned. However, only using top 1 category for predic-

Figure 1.3: Scene classification prediction example. Top5 groups (green) and categories (white) are shown with their probabilities.

tion sacrifices prediction accuracy, which can be avoided by grouping as shown in group confusion matrix.

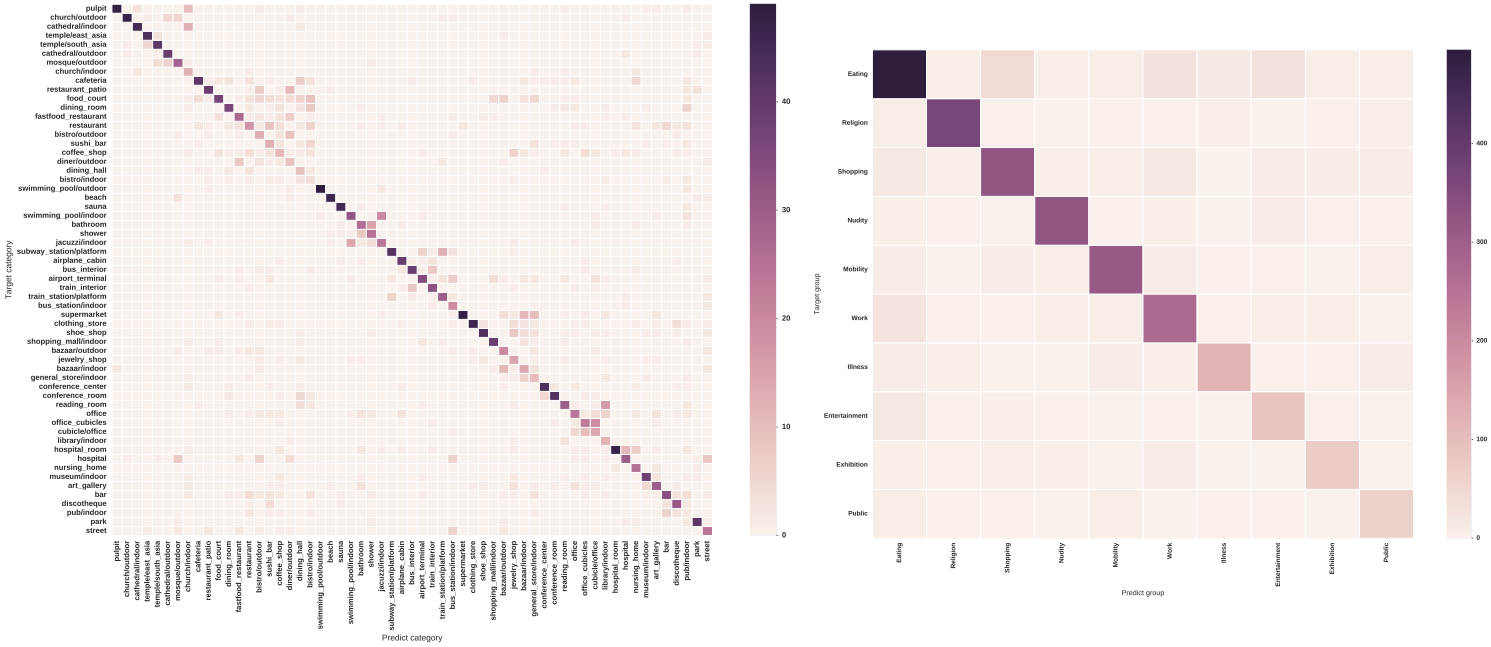

Figure 1.4: Confusion matrices for category prediction (left) and group prediction (right).

1.2.2 Face Recognition

Like scene classification, we select a pre-trained model for face recognition task. More specifically, the pre-trained model will serve as face feature extractor, and we will update the face classifier whenever new users register in Cardea and upload their face features. There are already many deep neural networks deployed in commercial products, like Megvii's Face++ [8], Facebook's Deepface [9], Google's Facenet [10], Sensetime's Deepid [11]. OpenFace [12] is an open source project that is gaining attentions in recent months, it is based on Torch [13]. Because Cardea's other modules are under Caffe framework, we limit our options on open sourced Caffe models. The models in our consideration are VGG face recognition model [14] and Lightened CNN face recognition model [15].

To compare performance of features extracted from the two models, we run t-SNE visualization [16] on the features of a small dataset we previously collected for emotion sensing. Fig 1.5 shows the t-SNE visualization result. It seems VGG feature and Lightened CNN feature have similar performance, at least on this small dataset. Though it is found that comparing to Lightened CNN model, VGG model is more robust to variations and its features show better transferability [17], the

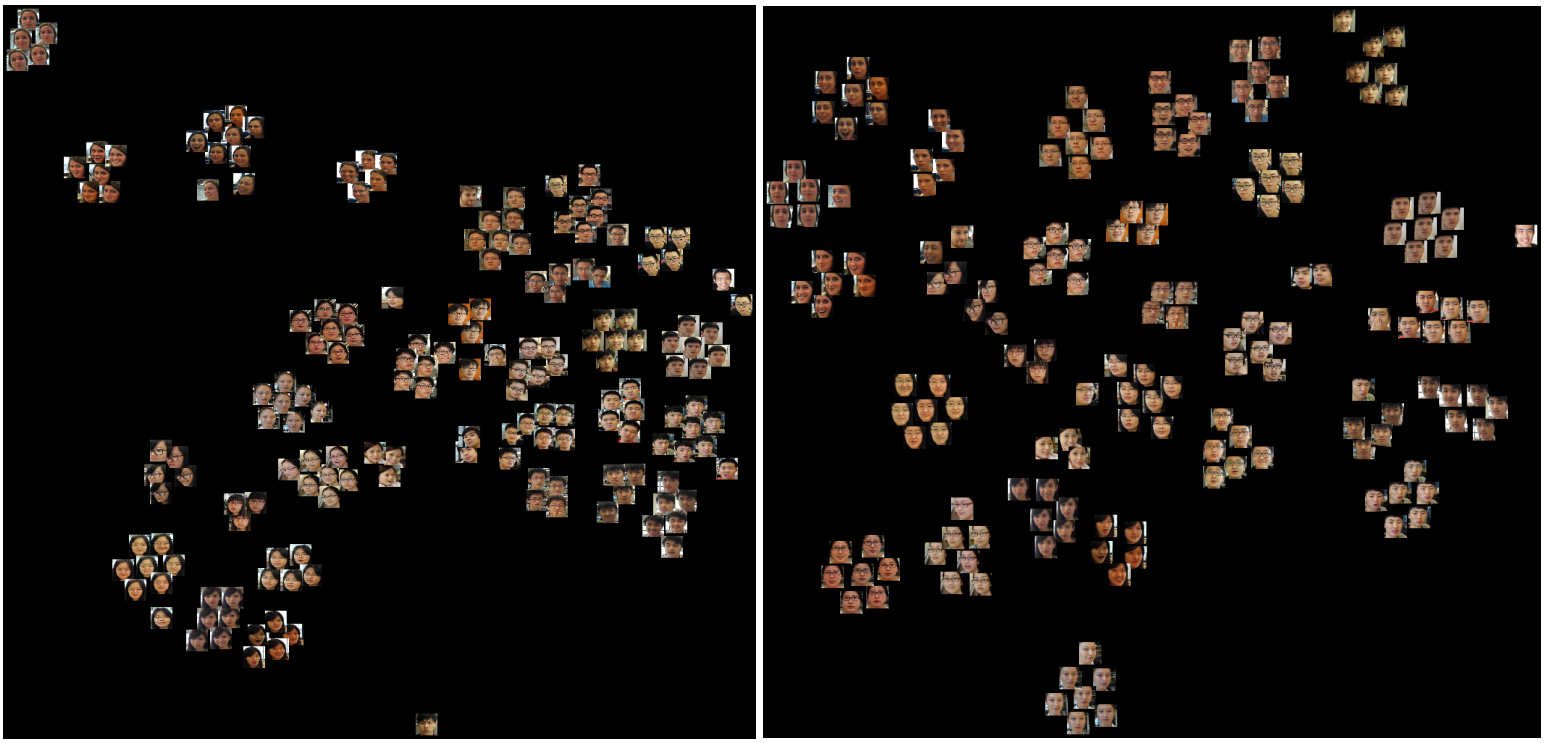

Figure 1.5: t-SNE visualization of VGG *fc8* layer features and Lightened CNN *fc1* layer features.

model size is more than 500MB, 10 times bigger than Lightened CNN model. And the released VGG model has a feature dimension of 4096, while Lightened CNN model has a feature dimension of 256. Our experiment on different Android smartphones shows it takes 10 times longer to extract VGG features. Table 1.2 shows the forwarding time we tested on different smartphones. It can be seen VGG model consumes much more memory that it can only run on phones with memory larger than 3GB. Due to above concerns, we use Lightened CNN model in our implementation.

Table 1.2: Time of single facial feature extraction and batch facial feature extraction (10 faces).

	Xiaomi Mi 3W Snapdragon 800 2GB RAM	Galaxy Note 4 Snapdragon 805 3GB RAM	Xiaomi Mi 5 Snapdragon 820 4GB RAM
1 VGG CNN	N/A	N/A	$\sim$ 2780 ms
10 VGG CNN	N/A	N/A	$\sim$ 26740 ms
1 Lightened CNN	$\sim$ 508 ms	$\sim$ 330 ms	$\sim$ 303 ms
10 Lightened CNN	$\sim$ 6602 ms	$\sim$ 3971 ms	$\sim$ 2031 ms

Detection and Alignment

Lightened CNN model takes aligned face as input, requiring that the distance between midpoint of eyes and midpoint of mouth is 48, and y value of midpoint of eyes is 40, as shown in Fig 1.6. We use OpenCV's haar cascade [18, 19] frontal face detector. The limitation it brings to Cardea is only frontal faces will be detected and recognized. We set *minNeighbors* (the parameter specifying how many neighbors each candidate rectangle should have to retain it) to be 3 to ensure a relative high recall for face detection. To remove false positive, we further apply skin color filter (range $[0, 48, 60] - [30, 255, 255]$ in HSV color space) on retained rectangles. Following that, we use Dlib library's HOG [20, 21] based face detector as a second stage filter. Note that Dlib's face detector has higher accuracy comparing to OpenCV's face detector, but is much slower if applied directly on a high resolution image, therefore it is used as a filter on small rectangular areas. Dlib's facial landmarks detector [22] is also used in later face alignment stage, it can detect 68 facial landmarks [23, 24]. With the detected landmarks and required alignment condition about inputs to the CNN model, we can calculate the homography matrix that is finally used to align faces. The steps for detection and alignment is shown in Fig 1.6, and we implemented it as a JNI library for Android platform [25].

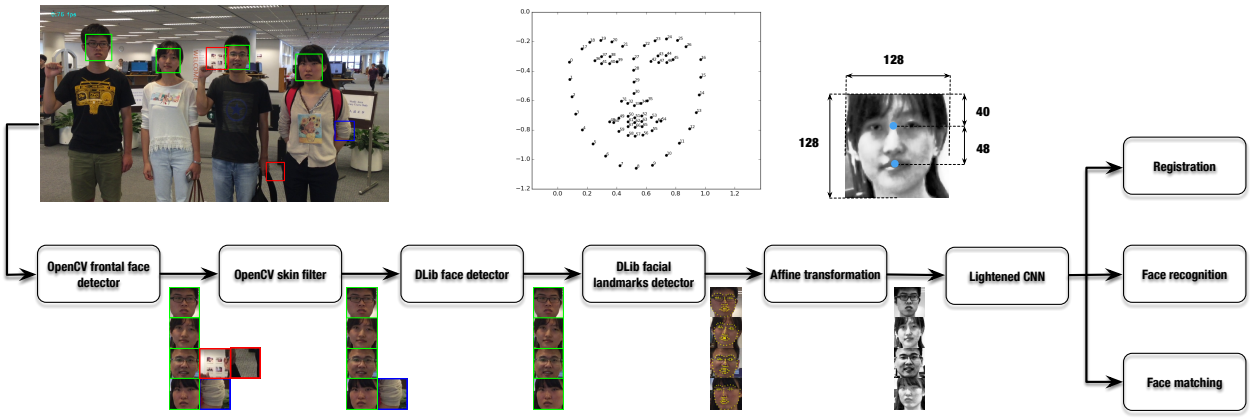

Figure 1.6: Face detection and alignment workflow.

Recognition

All the facial features uploaded by registered users are used to train a classifier in the cloud server, using LIBSVM library [26]. During training, we set the parameter to enable probability estimations of classes $p_i, i \in 1, \cdots, N$. In prediction time for each facial feature, if $\max_i p_i \leq 0.3$, then we treat it as from an unknown person who hasn't registered in Cardea, otherwise it is from the user who has the highest probability and his privacy preference will be fetched for further processing.

Matching

Face matching occurs when a recognized user A has also specified and uploaded features of person B with whom he doesn't want to be captured, it is to determine whether B also appears in this captured image. Note that B is not necessarily a registered user of Cardea. It is required that N_B, the number of B's facial features uploaded by A should be more than 10. Then feature f_l from other faces $l \neq A$ in the image will be compared with B's features. `Cosine` similarity is used as distance metric. Among N_B distances between f_l and B's features, we can calculate the ratio r of distances which are shorter than a threshold $d_{threshold}$, if the ratio r is higher than a threshold $r_{threshold}$, then l and B are the same person, thus B appears with A in the same image. By tuning, we find $d_{threshold} \in (0.3, 0.5)$ and $r_{threshold} \in (0.5, 0.8)$ shows good enough performance. In Fig 1.7, we plot the distribution of distance between same person's Lightened CNN features and different person's Lightened CNN features. The features are extracted from all the faces in ORL face database [27], which consists of 400 images from 40 distinct subjects, 10 images per subject. Each subject has different photos, such as: with/without glasses, open/closed eyes, and different facial expressions. It is obviously seen that distances between same person and different persons are well seprated, especially for the case of `cosine` similarity.

1.2.3 Gesture Recognition

In Cardea, "yes" (✌) and "no" (✋) gestures have the highest priorities and are used to temporarily overwrite privacy preferences. To recognize gestures in daily life images, the first step is detection of hands, and it turns out to be the most challenging part in this sub task. Skin color based hand

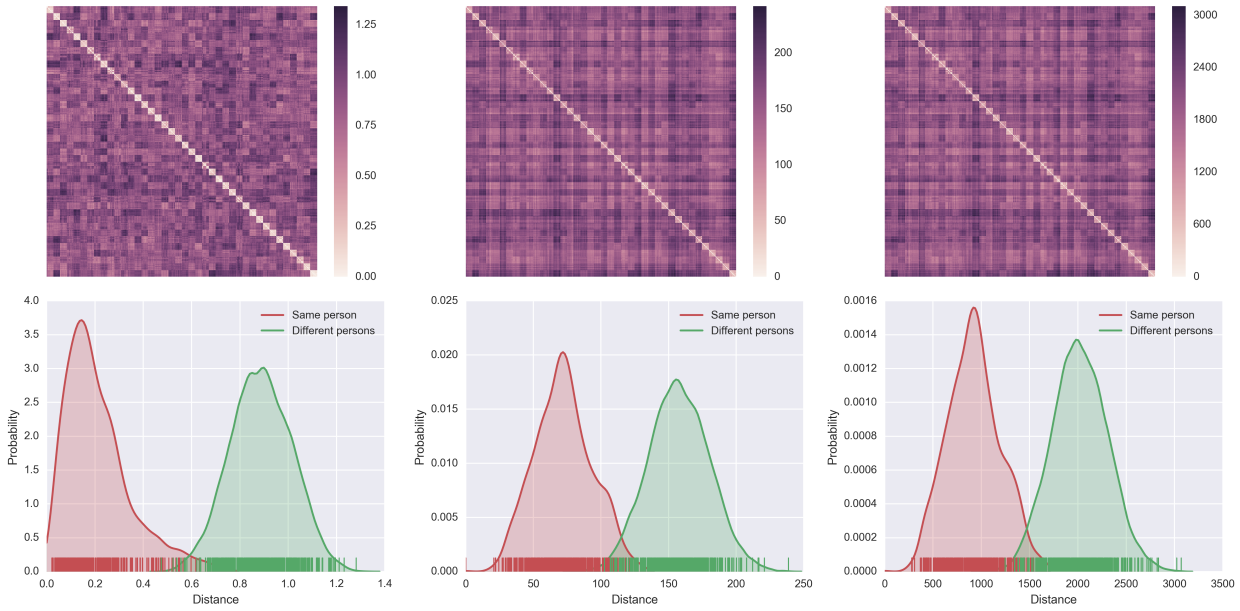

Figure 1.7: Distance matrix (top) and distance distribution (bottom) of Lightened CNN features using `cosine` similarity (left), l_1 norm (center) and l_2 norm.

detector will fail dramatically in images with cluttered background. A much more robust method will be using multiple proposals [28] based on hand shape, context, and skin color. However, it takes an extremely long time to detect hands in one image. Finally, we choose to leverage state-of-the-art detection framework faster R-CNN [29] to train a gesture detector in an end-to-end manner.

Data Preparing and Preprocessing

VGG group has shared a comprehensive dataset of hand images collected from various different public image data set sources in [28, 30]. It contains 5628 images, which is composed of 4069 training images, 738 validation images, 821 testing images respectively, each image is with annotations of hand bounding boxes. However, this dataset can only let us train a hand detector. To achieve the goal of recognizing gestures, there are two solutions in our consideration:

- First train a hand detector using this dataset, then train another hand gesture classifier using other commonly used gesture datasets and pipe them together.
- Take this dataset as a subset of images with ”natural” gestures, then prepare extra images

with ”yes” and ”no” gestures including annotations by ourselves, and train a ”natural/yes/no” gesture detector end-to-end.

The first solution is not an end-to-end solution, and the specific ”yes” and ”no” gestures may not be included in those standard gesture datasets, then we will still need to prepare our specific gesture dataset like in second solution. Therefore, we choose the second solution, based on the observation and also assumption that annotated hands in VGG's hand dataset are in natural relaxing modes, thus will not be treated as ”yes” or ”no” gestures. The annotations of VGG dataset are tilted rectangles shown as yellow ones in Fig 1.8, we re-annotate the dataset using bounding boxes of the original annotations shown as blue rectangles.

We crawled 527 images with ”yes” hand gestures, and 363 images with ”no” hand gestures. Note that in a crawled image, it may contain different types of hand gestures as shown in Fig 1.8, which is not a problem so long as gesture types are annotated correctly (*remind* that all gestures in VGG dataset are treated as ”natural” class). These images are crawled from Google and Flickr image search with keywords such as ”victory sign”, ”stop gesture”, ”palm gesture” and so on, many of them are focused on the hands thus don't contain many background pixels. We rescale these images in different scales and then pad zeros on rescaled images. In Faster-RCNN python implementation [31], an input image is rescaled to around 1000×1000 before fed to region proposal network. If without padding data augmentation step, the bounding boxes of hand gestures will be huge in many crawled images that are focused on hands, which makes the learned model not able to detect small hand gestures and also not perform well on the regression of large hand gestures. Another reason for the padding step is to counter data imbalance of three classes. After augmentation, we have a dataset of 13843 images, including 5628 images from VGG dataset, 4712 augmented images mostly with ”yes” gestures and 3503 augmented images mostly with ”no” gestures. Fig 1.8 shows some sample images with annotations from this composed dataset. We wrote a tool [32] to annotate the crawled images.

Figure 1.8: Training hand gesture dataset composed of VGG hand dataset (blue) and augmented crawled dataset (red and green).

Training Procedures

Using the composed dataset, we fine-tune the *conv3_1* and up layers of VGG16 pre-trained model provided by Faster-RCNN library, jointly with region proposal layers and detection layers that are not part of VGG16 pre-trained model. Features from *conv5_3* layer of VGG16 network are shared by region proposal network (RPN) and Fast-RCNN [33] detection network, RPN uses them to generate proposals, and region of interest (ROI) pooling layer in detecting network uses them for bounding box regression and classification. There are two methods to train a shared feature extraction network. One is an alternating optimization method with following steps: ❶ Train an RPN M_1 initialized from VGG16 pre-trained model M_0, ❷ Generate training proposals P_1 using RPN M_1, ❸ Train Fast R-CNN model M_2 on proposals P_1 initialized from M_0, ❹ Train RPN M_3 from M_2 without changing convolutional layers, ❺ Generating proposals P_2 using RPN M_3, ❻ Train Fast R-CNN model M_4 on proposals P_2 initialized from M_3 without changing convolutional layers, ❼ Add M_3's RPN layers to Fast R-CNN model M_4. Another method is an approximate joint optimization method by training with stochastic gradient descent as usual, which is easier, faster and achieves similar performance [31], so we use the second training procedures.

Prediction

During prediction, we set Non-Maximum Suppression (NMS) threshold as 0.4 and confidence level threshold as 0.7. Figure 1.9 shows some examples of gesture detection and recognition results in natural environment. It can be seen that the trained model can handle cluttered background such as in shopping environment, and indoor dark lighting condition. It is interesting to notice that bounding boxes for "natural" hands are bigger, reflecting the fact that we select re-annotated the VGG dataset for "natural" class using bounding boxes of the original annotations. The model has a good recall in terms of hand detection, however, its gesture recognition is sensitive to motion blur, palm angles and gesture size. We think the good recall of hands comes from the comprehensive VGG dataset, and the not so good recognition result is because the "yes/no" dataset we composed does not have a good quality because gestures are focused in many images, especially for "no" gestures, which is reflected by the observation that "yes" gesture recognition performs better than

”no” gesture.

Figure 1.9: Examples of gesture detection and recognition.

1.3 System Integration

1.3.1 Deployment on Android

Ideally, for a privacy control framework, we prefer a design that does not require cloud server and all the algorithms run locally on mobile devices. In the current design and implementation, cloud server exists mainly for two reasons: ❶ Storage center for profiles and hosting face recognition model; ❷ RPN in gesture recognition task is written in Python language, thus gesture recognition can not run on Android smartphones easily. However, these are not hard restrictions, possible improvements are discussed in Chapter 5.

The deployment of Caffe models for scene classification and facial feature extraction is based on Caffe-android-library [34], we modified its code [35] to support loading multiple Caffe models, batch feature extraction, and only forwarding to a specified layer during feature extraction, which

saves useless computations from fully connected layers. There are other libraries for deploying neural networks on mobile, such as Torch-android [36] and MXNet [37]. The deployed scene classification model (based on AlexNet structure) has a size of 230MB, which is not small. However, its prediction is very fast, and can be easily fitted into the time slot of waiting response from cloud server. The facial feature extraction model (based on Lightened CNN structure) has a relatively bearable size of 33MB. Comparing to AlexNet, Lightened CNN model has smaller filter sizes, but with many more feature maps, therefore in run time, Lightened CNN model consumes about 1GB memory, which makes it not able to run on smartphones with less than 2GB memory 1.2. Possible ways to decrease model size and optimization of resource consumptions are also discussed in next chapter. Other lighter models we deployed on android include OpenCV face cascading model (less than 1MB), and shape predictor (90MB) in Dlib facial landmark detection.

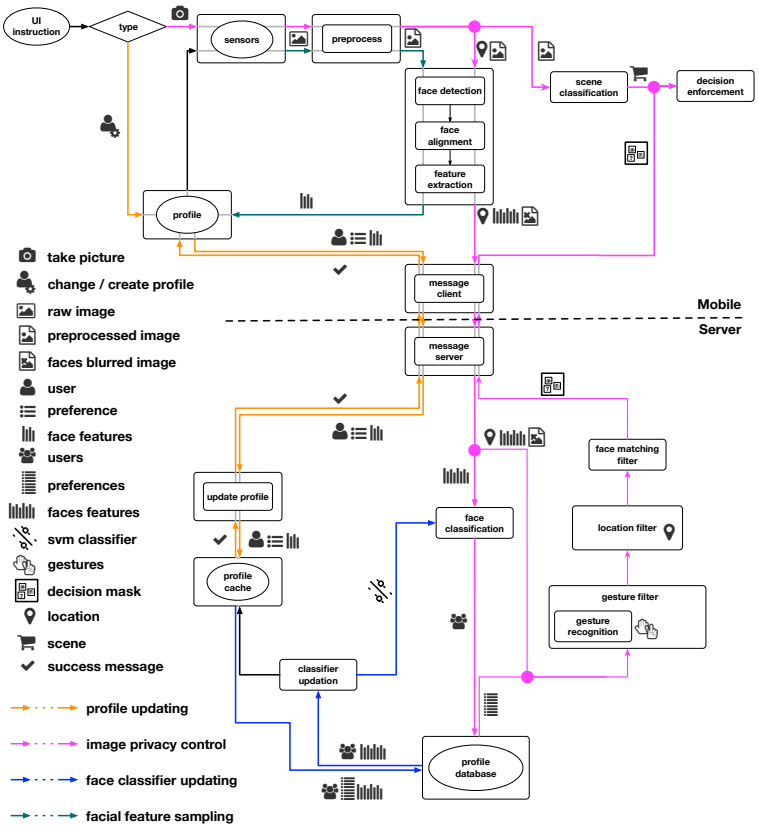

Figure 1.10: Dataflow of Cardea

1.3.2 Dataflow and Integration

Fig 1.10 shows the detailed structure and dataflow of Cardea, which is mainly composed of the following steps (a demo video about the usage can be found in [38]):

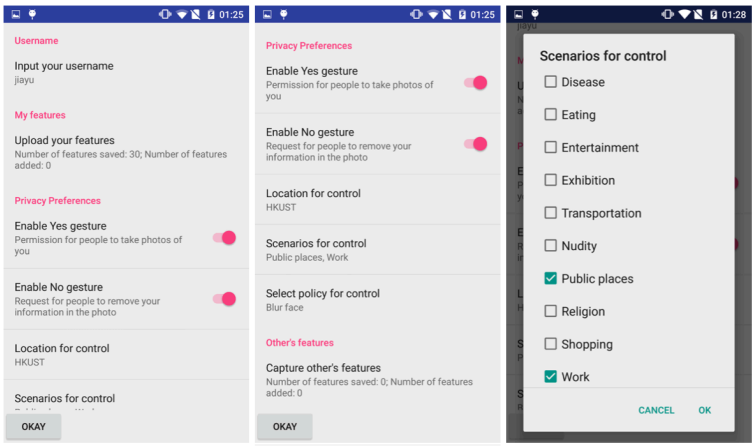

Figure 1.11: Registration and profile updating interface

Registration and Profile updating: The interface of registration for bystanders and updating of profiles is shown in Fig 1.11. He is able to select one or more scene categories, one location for control, as well as enable gestures or not. In two cases his facial features will be packaged with his privacy preferences: one is in registration time and the other is when he wants to update his facial features in the cloud. This registration and updating message will be send to server, if it is an updating message without feature updating, then his user profile in cloud will be updated immediately and he will receive a "success" notification, otherwise this message will be first buffered in a profile cache.

Face classifier updating: In the server, the face classifier will be updated intermittently. For every time interval ΔT, if there is cached messages that brings new facial features, it will ❶ block the queue of prediction messages from doing face recognition, ❷ merge profile cache with profile database, ❸ retrain a face classifer and ❹ unblock queued prediction messages.

Image capturing: When a recorder uses Cardea to take a image, extracted facial features, GPS coordinate as well as face-blurred image will be packaged as prediction message and sent to server

for processing. In the server side, after faces recognition and profiles retrieval, it will start the cloud part of decision making process for every bounding box, returned to the client side is a decision mask that specifies among all the detected faces, which should be blurred, which should be kept and which should be further determined based on the scene results calculated on the client side. In client side, while waiting for response from server it will calculate the scene result. With received decision mask, it makes final decisions on every face and enforces the privacy protection actions complied with everyone's preference.

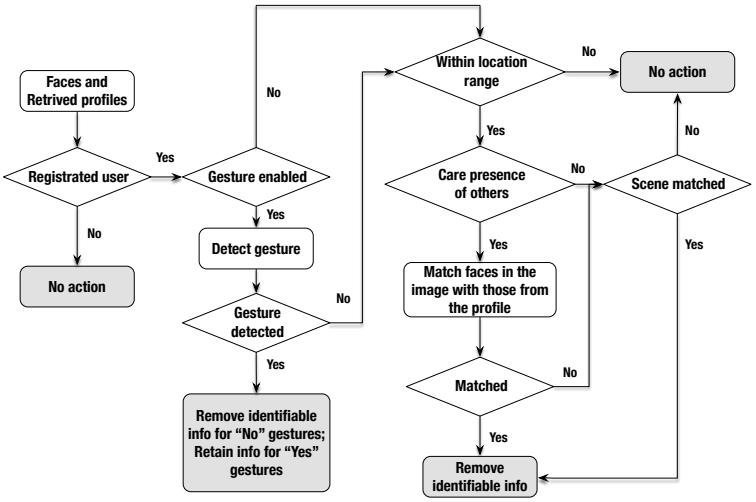

Figure 1.12: Steps of decisions about actions to be applied on detected faces.

Decision making: Fig 1.12 gives the detailed decision steps. Note that in this process, we need to match a detected gesture with the right face or people who issued the gesture, currently we simply take the nearest face of a detected gesture as the person who issued this gesture, thus it requires the user to put his hand near his face when sending "yes/no" gestures. As seen from the figure, when the detected face is recognized as a registered user, and context including location, scene, presence of other people and gesture matched with recognized user's profile, this detected face will be removed. For other cases, the face will be kept but it may cause removal of other faces in the face matching step.

Concurrent requests: To enable concurrent requests from different client apps, we implement multiple "recognition" workers to process the queued messages from all the clients, recognition results from different workers will be collected and put into a result queue, followed by multiple

”mailing” workers to make decision masks and mail the decision masks back to corresponding blocked client threads. We tested with a server configuration - ”Intel i7-5820K CPU, 16GB RAM, GeForce 980Ti Graphic Card(6GB RAM)” and the campus’s Wi-Fi network, the time from start sending message to receive server’s response is $1 - 2s$, depending on the message type. The total time from capturing moment to the enforcement of protection actions is $2 - 4s$, depending on how many faces detected. Most time is spent on facial feature extraction and message data transmission. Note that processing in the server side is relatively fast, gesture recognition of one image takes $200 - 300ms$, while the time spent on SVM face classifier and face matching is negligible. With 6GB GPU RAM, the server can serve 3 gesture recognition workers at the same time, supporting 5-10 concurrent requests. The implementation of Cardea is hosted in [39].

Bibliography

[1] *Places2 Dataset Project*. URL: http://places2.csail.mit.edu/index.html.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[3] *Release of Places365-CNN*. URL: http://github.com/metalbubble/places365.

[4] Ali Sharif Razavian et al. "CNN features off-the-shelf: an astounding baseline for recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014, pp. 806–813.

[5] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.

[6] *Caffe*. URL: http://caffe.berkeleyvision.org/.

[7] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *arXiv preprint arXiv:1408.5093* (2014).

[8] Erjin Zhou et al. "Extensive facial landmark localization with coarse-to-fine convolutional network cascade". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2013, pp. 386–391.

[9] Yaniv Taigman et al. "Deepface: Closing the gap to human-level performance in face verification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1701–1708.

[10] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 815–823.

[11] Yi Sun et al. "Deepid3: Face recognition with very deep neural networks". In: *arXiv preprint arXiv:1502.00873* (2015).

[12] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. *OpenFace: A general-purpose face recognition library with mobile applications*. Tech. rep. CMU-CS-16-118, CMU School of Computer Science, 2016.

[13] *Torch7*. URL: http://torch.ch.

[14] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. "Deep face recognition". In:

[15] Xiang Wu, Ran He, and Zhenan Sun. "A Lightened CNN for Deep Face Representation". In: *arXiv preprint arXiv:1511.02683* (2015).

[16] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605.

[17] Mostafa Mehdipour Ghazi and Hazim Kemal Ekenel. "A Comprehensive Analysis of Deep Learning Based Representation for Face Recognition". In: *arXiv preprint arXiv:1606.02894* (2016).

[18] *OpenCV*. URL: http://opencv.org.

[19] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2001, pp. I–511.

[20] *Dlib C++ Library*. URL: http://dlib.net.

[21] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 886–893.

[22] *Real-Time Face Pose Estimation*. URL: http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html.

[23] *Dlib facial landmarks*. URL: http://openface-api.readthedocs.io/en/latest/openface.html.

[24] *Dlib facial landmark coordinates*. URL: http://openface-api.readthedocs.io/en/latest/_modules/openface/align_dlib.html.

[25] *Face alignment JNI library*. URL: http://github.com/ZhengRui/FaceAlignmentJNI.

[26] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: a library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2.3 (2011), p. 27.

[27] *ORL face database*. URL: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

[28] Arpit Mittal, Andrew Zisserman, and Philip HS Torr. "Hand detection using multiple proposals." In: Citeseer.

[29] Shaoqing Ren et al. “Faster R-CNN: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[30] *VGG: Hand dataset*. URL: http://www.robots.ox.ac.uk/~vgg/data/hands/index.html.

[31] *Faster-RCNN (Python Implementation)*. URL: http://github.com/rbgirshick/py-faster-rcnn.

[32] *Image annotation tool*. URL: http://github.com/ZhengRui/ImgAnnotaPyQt4.

[33] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1440–1448.

[34] *Caffe Android Library*. URL: http://github.com/sh1r0/caffe-android-lib.

[35] *Caffe Android Library (Clone)*. URL: http://github.com/ZhengRui/caffe-android-lib.

[36] *Torch-7 for Android*. URL: https://github.com/soumith/torch-android.

[37] *MXNet on Mobile Device*. URL: http://mxnet.readthedocs.io/en/latest/how_to/smart_device.html.

[38] *Cardea demo video*. URL: http://drive.google.com/file/d/0B4z8qjK8O_uUc0o2RjZWYktiMTg/view.

[39] *Cardea project*. URL: https://github.com/ZhengRui/cardea.