CrossMark

Efficient eye typing with 9-direction gaze estimation

Chi Zhang[1] · Rui Yao[1] ⬤ · Jinpeng Cai[2]

© Springer Science+Business Media, LLC, part of Springer Nature 2017

Abstract Vision based text entry systems aim to help disabled people achieve text communication using eye movement. Most previous methods have employed an existing eye tracker to predict gaze direction and designed an input method based upon that. However, these methods can result in eye tracking quality becoming easily affected by various factors and lengthy amounts of time for calibration. Our paper presents a novel efficient gaze based text input method, which has the advantage of low cost and robustness. Users can type in words by looking at an on-screen keyboard and blinking. Rather than estimate gaze angles directly to track eyes, we introduce a method that divides the human gaze into nine directions. This method can effectively improve the accuracy of making a selection by gaze and blinks. We built a Convolutional Neural Network (CNN) model for 9-direction gaze estimation. On the basis of the 9-direction gaze, we used a nine-key T9 input method which is widely used in candy bar phones. Bar phones were very popular in the world decades ago and have cultivated strong user habits and language models. To train a robust gaze estimator, we created a large-scale dataset with images of eyes sourced from 25 people. According to the results from our experiments, our CNN model is able to accurately estimate different people's gaze under various lighting conditions. In considering disable people's needs,

✉ Rui Yao
  ruiyao@cumt.edu.cn

  Chi Zhang
  08133396@cumt.edu.cn

  Jinpeng Cai
  u5715685@anu.edu.au

[1] School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China

[2] College of Engineering & Computer Science, Australian National University, Canberra 2601, Australia

🖄 Springer

we removed the complex calibration process. The input methods can run in screen mode and portable off-screen mode. Moreover, The datasets used in our experiments are made available to the community to allow further research.

1 Introduction

Vision based human-computer interaction (HCI) is a hot topic in the field of computer vision. We designed a low-cost gaze controlled text entry method to help disabled people those who may be unable to carry out the basic activities of daily living but can move their eyes achieve text communication.

A typical video-based eye typing system uses a camera to capture eye movement, from which users can type in words by looking at an on-screen keyboard. This process can be divided into two parts: The first part is an eye tracking system and the other part is a text entry system. The eye tracking system aims to detect eyes from images and then estimate the gaze direction. Before using the system, users must do the calibration to map estimated directions to corresponding positions on the screen. For the text input system, it is based on where people are looking at on the screen to make selections and input text. In previous works, researchers usually design systems on the basis of the existing eye trackers and they only need to design text input systems. However, the following issues widely exist in such systems: First, the quality of the eye tracker constrains the design of the text input system. A high accuracy eye tracking system can estimate human gaze with less error, so it can precisely select the targets even if there are more intensive objects on the screen. However, a high accuracy tracking system demands high precision equipment, with low precision equipment unable to meet the input system accuracy requirement. Accurate eye tracking requires high-resolution human eye images, which demands a high quality camera and optimal lighting conditions. This increases the cost and limits the range of applicable environments. Furthermore, if the device is changed, the camera needs to be recalibrated. So, most previous systems are device-specific.

Currently, there are still many challenges for eye tracking that need to be overcome: low-resolution images, poor lighting conditions and physiological responses (such as people looking down having a natural squint and tired people showing droopy eyelids). Figure 1 illustrates examples of eye images that are hard to use for estimating gaze. Secondly, the accuracy of human gaze angle changes is 0.5° [24], and even when people stare in a fixed direction, there still exists a jittering movement of the eyes and the body also moves subconsciously, both of which can make it difficult to execute accurate and stable eye tracking. Finally, an eye tracker requires a lot of time to calibrate before using. During the calibration process, people are not allowed to move their body, and slight movements may result in an error in the final estimation. As usage time increases, the calibrated system can become skewed and users may need to calibrate the system again. It is easy to foresee how these inconveniences could become an issue for disabled people attempting to use these systems in their daily life.

In order to solve the problems above and consider the needs of disabled people in the application of these systems, instead of employing an existing eye tracker, we attempted to solve the eye tracking problem and input method problem together. We separated human gaze into nine directions: left-up, up, right-up, left, middle, right, left-down, down and right-down.

Fig. 1 Eye images that are difficult to be used for estimating gaze

The division has the following advantages: First, gaze directions can be more easily recognized. The vertical visual field of both eyes is 100° and the horizontal visual field is 200° [3, 5]. If we divide those angles into nine directions, every direction has the range from 30 to 70°. The angle range allows the gaze estimator to have a larger fault tolerant space, which means the nine directions can be recognized with lower error. This can also lower the required standard for the devices employed and the conditions it is used under. In other words, it reduces the cost and increases the range of applicable environments. Second, it can simplify the process of calibration. The calibration process usually contains camera-calibration, geometric-calibration, personal calibration and gaze mapping calibration [10]. The whole process aims to determine a bunch of parameters. A common calibration method is to project points on the screen and have the user follow the points with their eyes accordingly. The calibration process becomes easier if we simplify eye gaze into nine directions. As the nine directions are centrosymmetric, if we want to map nine directions of gaze to nine regions of the screen when using the system (see Fig. 2), we only need to keep the head stationary and eyes level with the center of screen. After adjusting the distance between the eyes and the screen, we can execute the calibration process easily. Moreover, due to the large fault tolerant space, slight body movement does not affect the calibration results. In our work, we introduce a CNN based method to estimate human gaze. The estimation results of the CNN are ten states of eyes which include nine gaze directions and one eye-closed state.
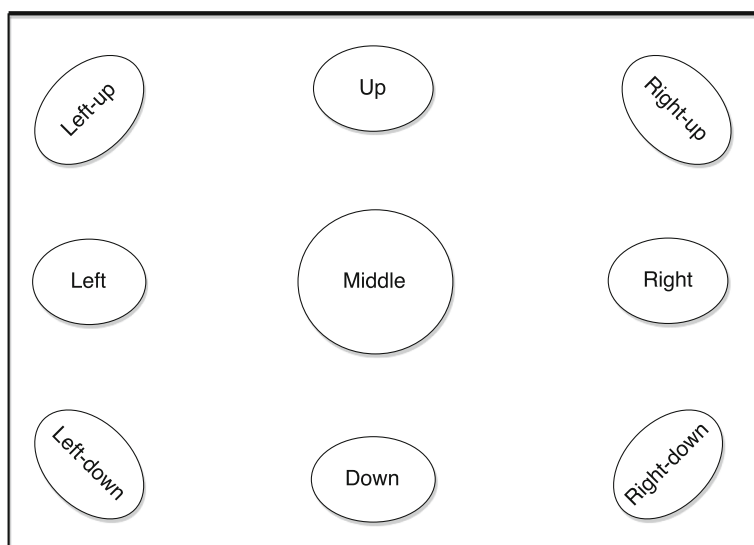

Fig. 2 9 Corresponding regions on the screen of the 9 directions

We trained the CNN model to learn the mapping between the ten categories and corresponding eye images. A big advantage of using CNN to estimate gaze is that CNN methods do not require complex calibration to determine a series of parameters, and in some CNN based methods, calibration is even not needed [14].

When designing the text entry system, many factors need to be considered. Tradeoffs often must be made between accuracy, text entry speed, cost and accessibility. Our goal was to design a low-cost input method which can use off-the-shelf devices to help disabled people type in words with high accuracy and adequate speed. The text entry system can be used in daily life rather than just being limited to laboratory conditions. To achieve this goal, the methods needs to withstand influence from factors including false eye detection, involuntary blinks, saccade, illumination, and image resolution. The basic design is to point letters from an on-screen keyboard by gaze and select by voluntary blinks. Our keyboard layout is close to the classic 9-key T9 input method keypad used on bar phones. The T9 input method has many advantages: First, it requires less buttons in comparison to the QWERTY keyboard. Less buttons support larger on-screen targets which has a larger fault tolerant space on the screen, allowing for higher selection accuracy. Second, as the 9-button keyboard layout is widely used in bar phones, most people possess the experience that would enable them to easily become familiar with the layout. Also, there are many language models of the T9 input method, which can greatly increase the text entry speed. On a bar phone with a word prediction and completion function, users can at most type in 46 words per minute using the T9 input method [28]. Third, the 9 buttons of the T9 input method can be seen in the 9 directions in our system. Buttons 1 to 9 represent left-up, up, right-up, left, middle, right, left-down, down and right-down respectively. Unlike previous designs in which people have to watch the screen to make selections, as people can find the 9 directions by feeling, users can choose the targets without a screen. So, our input method can run both in screen mode and off-screen mode. For off-screen mode, our text entry system can be portable and users can use the system anytime and anywhere. Figure 3 shows the overview of the proposed method.

The contribution of this work is threefold.

1. We made a dataset by collecting a number of double-eye images from 25 participants, which is made available to the community to allow further research on this topic.
2. We built a CNN model for 9-direction gaze estimation based on our new dataset. Further, we designed several data augmentation methods to increase the robustness of our gaze estimator.
3. We presented an input method based on the state of the eyes in each frame of a video sequence. The input method is able to handle false detected eyes and filter noise from involuntary blinks, saccade and momentary false estimation.
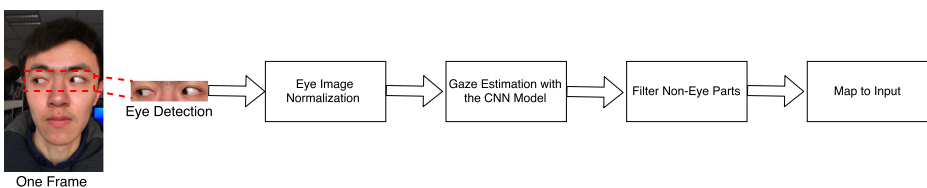

One Frame

Fig. 3 Overview of our vision based text input method. For each frame of the video sequence, we first detect both eyes from the image. Then we normalize the eye images by resizing them to 32×128 pixels. We use the CNN model to estimate the states of the eyes and to eliminate false detected eyes. We input text according to the estimated results of each frame

2 Related work

In order to track the eyes, the first step is to detect the human eyes from the video sequence frame by frame, then to estimate the gaze direction in accordance with the eye detection and calculate the positions on the screen by calibration. There are three popular methods of gaze estimation: model-based gaze estimation, electrooculogram (EOG), and appearance-based gaze estimation [10, 23]. The model-based method mainly focuses on analyzing the physical model of human eyes. We can further divide the model-based method into corneal reflection and shape-based methods on the basis of whether external lights are used to irradiate the eyes [32]. When using corneal reflection to detect eye features, infrared lights illuminate the surface of the cornea and multiple cameras are needed to collect eye images [23, 26]. For the shape-based method, gaze directions are inferred directly on the basis of detected eyes shapes, such as corneal center and iris edges [1]. A disadvantage of the model-based method is that it requires relatively strong lights and high resolution images. The EOG method simulates human eyes as a dipole. The cornea is its positive pole and retina is its negative pole [23]. If cornea-retina has a stable potential difference, we can detect a steady electric potential field. The eye movement can change dipole orientation, which results in variation of electric potential field. The drawback of EOG is that electrodes are attached around the eye region and EOG is easily influenced by surrounding noise. In addition to that, EOG increases the application cost and is constrained by the applicable environments [23]. The appearance-based method is able to directly use human eye images as the input and can effectively handle low-resolution images. The method learns the features for recognition based on lots of training images, which are then used to predict gaze direction. The model-based and EOG methods were widely used previously and saw success in products [23], but the appearance-based method was seldom used in applications. In recent years, learning methods have made a huge success in human activity recognition [2, 16–21, 27], and attracted numbers of research. Despite this, the performance of the appearance-based method in accurate gaze estimation tasks is still not ideal. This method requires a huge amount of eye image data and there is uncertainty as to whether the estimators could apply to unknown users [32]. During the gaze estimation process, the first two methods have an understandable error, because they construct a relationship between extracted features and gaze directions. However, appearance-based gaze estimation builds a relationship between pixel values and gaze directions. The latent learned features used for estimation is unknown. Even though people keep fixed eye direction, the slight variation of illumination may result in different estimation results. Thus, the appearance-based method may produce unstable and discontinuous results during real-time tracking [14].

There are three popular ways of eye typing. The first method is typing by point. Commonly, an on-screen keyboard is shown and people operate the keyboard by looking at the keys. Using a QWERTY keyboard [7, 13, 22] can achieve a high text input speed. However, it demands accuracy regarding the eye tracker and the error rate can be higher where there is inaccuracy. A hierarchical keyboard [7, 9, 11] chooses a letter through multiple selections. For example, choosing a letter group then choosing a letter from that letter group. It can decrease the error rate and has a low requirement of devices, but as a single letter is input with multiple selections it has a lower input speed. It is usually used in low-cost systems. Fixation or muscle movement such as blinks are used to make selections. If blinks are used to make selections, involuntary and voluntary blinks must be distinguished. While fixation is used to make selections, the time of fixation is called dwell time. It sets a maximum text input speed and is easily interrupted by blinks and jitter eye movements [23]. The second

method is typing by eye-switches. A common way is that an alphabet is shown on the screen and rows or columns are scanned automatically. When the rows or columns where the target letter is located are scanned, users can blink eyes to make a selection as a binary switch [8, 15]. This method is used for people who can not move eyes but can still blink. Coarse eye movement can also be used as switches. For example, looking left is to start while looking right is to select [25]. The third way is typing by gestures. I4control system [6] used eyes to control a cursor moving over an alphabet and blinks were used to make selections. Eyes worked like a joystick. Wobbrock [31] used letter-like gestures to input letters, which used Edgewrite input method that is used in PDAs, joysticks and trackballs. It was to design an eye gesture for each letter. Dasher system [29] used continuous pointing gestures with a zooming interface. It used dasher language model to predict the next possible letter. Commonly used vocabulary and sentences could be input with a higher speed and it had lower error than common on-screen QWERTY keyboard. However, it only supported 4 languages.

3 Dataset

In this section, we introduce a large-scale dataset for the gaze estimation task in our work. We divided eye states into 10 categories which consist of 9 directions and an eye-closed state. It took us about 1 month to collect and process eye images from 25 participants. It covered various eye appearance, illumination, locations and time. Eye images were taken in everyday life rather than controlled lavatory conditions. Our collecting devices are mobile phones, webcams and digital cameras. We also implemented many data augmentation methods to expand our dataset before and during training to make our gaze estimator more invariant to illumination, skin color, image resolution and slight head rotation. Figure 4 shows some sample images from our dataset.

3.1 Dataset characteristics

Previous appearance based gaze estimation works often take single-eye images as input data, but it may cause problems in some cases. By using the single eye to estimate gaze, a strong assumption is that both eyes are synchronized when looking at a target. However, two eyes are not always synchronized. Figure 5 shows a participant looking at targets locating at the same gaze direction but different distances. Even though eyes are looking at the same angle, synchronization of two eyes varies. As is shown, usually the closer a target is to the eyes, the less likely it is for both eyes to be synchronized. Particularly, eyes become crossed
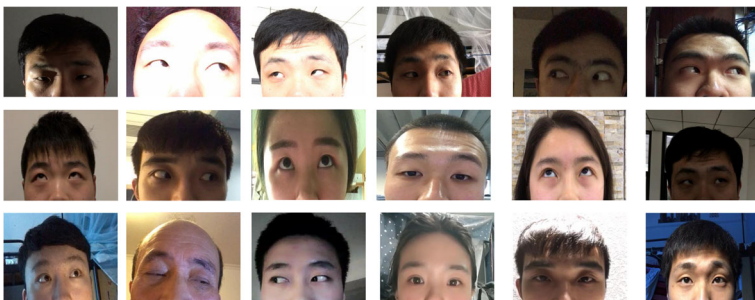

Fig. 4 Sample images from our dataset

Fig. 5 A participant looking at different targets locating in the same angle. From top to down, the targets are located 5 cm, 10 cm, 15 cm, 30 cm and 50 cm from eyes

when the target moves too close to the middle of both eyes. If single eye is used to estimate gaze, different results may be achieved from two eyes. An advantage of using images of both eyes is that even though one eye is unclear and impossible to use to estimate gaze due to poor lighting conditions or low resolution, the estimator can still estimate gaze from double eyes. The reason is that double-eye appearance based gaze estimation learns features from the eye region and the combinational features of both eyes can be learned by a CNN to better estimate gaze. So in our work, we used images of both eyes to train and test. We applied cascade double-eye detector using Viola-Jones algorithm [30] to detect eyes.

3.2 Collection procedure and data augmentation

First, we recorded 162 video sequences from 25 people under different lighting conditions and each video lasts about 10 to 30 seconds. Two different ways were used to record the videos. The first way was that the camera is 20 cm in front of the face and the camera lens are level with eyes. They were asked to look at 9 directions by feeling and close eyes successively. The second way was that the participants looked at 9 targets on a board and closed eyes successively. The size of the board is 30 cm × 50 cm, which is 35 cm far from the face. The camera is put in the center of the board. Camera lens and eyes are on the same level. Each video covers 10 states of the eyes. We used two ways to collect dataset because our input method runs in two modes: screen mode and off-screen mode. In the off-screen mode, the angle of gaze has a wider range while in screen mode, the angle of gaze is limited in the screen region. After all the videos were recorded, we cropped eye images with the eye detector. As we use the eye detector to detect eyes for estimation, the detector knows the best eye region to crop for estimation. Then we manually picked 1 to 3 eye images of a state that looked different in each video and recorded their labels and frame numbers in the video. For the validation set and the test set, we asked participants to take videos at a different time and repeated the same procedures instead of splitting collected data, because this can avoid overfitting caused by similar eye images. We prepared two test sets. One was from people who contributed eye images to the training set. We call them known people. The other one

was from new people and we call them unknown people. Even though test images from known people can achieve better results, the test of unknown people can evaluate whether a estimator can be applied to arbitrary users. So far we collected 832 training images, 728 validation images, 501 test images from known people and 434 test images from unknown people.

Then we implemented three data augmentation methods to expand our dataset and increase robustness. First, according to the frame numbers we recorded before, we applied some image transformation methods. We rotated the face images by 2.5° and −2.5° and detected and cropped the eye region again. We also scaled detected eye regions by 1.2 and 1.5 times and made 5-pixel shifts of the detected regions along 8 directions. Then we saved them as new training images with the same labels. We repeated the transformation to every frame we recorded. This can make the estimator more robust with respect to slight head movement and detected eye regions. Then we manually deleted false detected eye images. We expanded our training set to 74101 eye images. Second, during training, we implemented the method used in [32] which was to flip the eye images horizontally with the probability of 0.5 and changed the label to the opposite direction, namely swapping labels between left-up and right-up, left and right and left-down and right-down while others remaining the same. Third, to make our estimator robust with respect to illumination, in training process we made a small random adjustment to HSV channels of eye images. By changing the values in HSV channels, we changed the hue, saturation and value level of images. Figure 6 shows the results of images after adjustment of HSV channels is made.

4 Proposed method

In this section, we introduce how we type in words by gaze directions and voluntary blinks. Our general method is pointing by gaze directions and making selections by voluntary blinks. For a video sequence, we need to process each frame. We first detect both eyes from an image and then crop the eye region. After normalization, the eye images are used to estimate states of the eyes. If no eyes are detected in the image, the frame is skipped. If multiple eyes are detected, we eliminate false eyes by the gaze estimator. We choose the detected double eyes with the highest score in the CNN output layer as the real eyes. Then we input text according to the states of the eyes. To make the input system work properly, we need to handle false estimated states, and filter estimated states of saccade and involuntary blinks.

4.1 Gaze estimation with CNN

CNN is a powerful kind of neural networks that has a great performance in object recognition. Normally CNN is composed of four parts, including convolutional layers, pooling layers, non-linearity layers, and fully-connected layers. In the following, we give a brief introduction to them respectively and present our network structure.
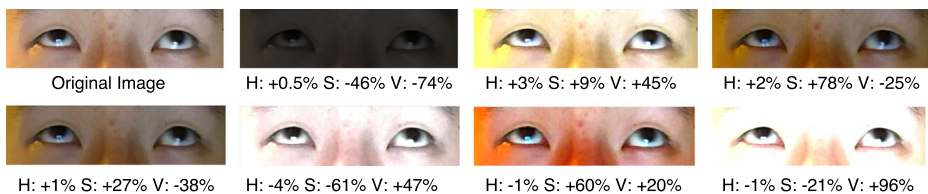

| Original Image | H: +0.5% S: -46% V: -74% | H: +3% S: +9% V: +45% | H: +2% S: +78% V: -25% |
| H: +1% S: +27% V: -38% | H: -4% S: -61% V: +47% | H: -1% S: +60% V: +20% | H: -1% S: -21% V: +96% |

Fig. 6 Effects on images after the adjustment is made on HSV channels of images

In order to apply convolution, we apply convolutional filtering over the image. After sliding a learnable filter over the RGB image and computing the dot product, we get a feature map, which is also called an activation map. The primary purpose of convolution is to extract features from the input image. It can also preserve the spatial relationship between pixels by learning features from input data. Spatial Pooling, also called subsampling or downsampling, reduces the dimensionality of each feature map but retains the most important information. It can be of different types: Max pooling, Average pooling, Sum pooling etc. Max pooling is used in our architecture. We use a 2×2 window sliding over the feature map and keep the largest element within the window. One typical non-linearity operation is called ReLU (Rectified Linear Unit), which is employed in our network. This operation replaces all negative pixel values in the feature map with zero. The reason for using non-linearity operations is that most real-world data is non-linear. Thus this operation allows our network to better recognize objects from the real world. The fully-connected layers consist of fully connected Multi-Layer Perceptrons that use ReLU as the activation functions in our model. The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron in the next layer. Generally, in CNN, convolutional layers, pooling layers, and non-linearity layers are mainly used for feature extraction. The classification is done by the trainable fully-connected layers. We also employ batch normalization layers [12] in our model. Batch normalization layers perform normalization for mini-batch in the training process. It can act as a regularizer and works effectively in accelerating training, reducing reliance on weight initialization and avoiding overfitting.

The architecture of our CNN model is shown in Fig. 7. We use RGB images with a size of 32×128 pixels as the input to the network. For the first convolutional layer, the feature size is 3×3 and we pad zeros around the input image. The number of filters is 64. After the convolutional layer, we add a batch normalization layer, followed by ReLU and the max pooling layer. We repeat the same operation on second and third convolutional layers. The fully connected layer has 300 hidden units and each unit has connections to all feature maps of the third convolutional layer. The output of our network is ten categories. We use softmax loss function to calculate the loss for optimization.

4.2 Noise filtering

In the process of continuous gaze estimation, noise mainly comes from natural blinks, saccade and false estimation. Figure 8a shows estimated states of eyes in each frame of a video sequence. As is shown, the signal is stable and accurate during fixation and is unsure during saccade and blinks. Figure 9 shows a natural blink process and its estimated states of each
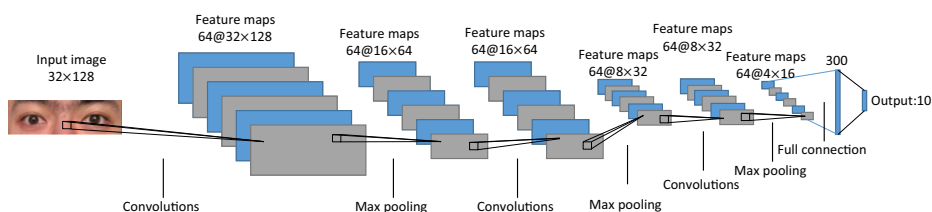

Fig. 7 The architecture of our CNN model. The CNN consists of 4 blocks. The first three blocks have the same operations which include convolution with 64 3×3 filters, batch normalization, ReLU and max pooling. The fourth block is a fully-connected layer with ReLU as the activation function. The input is an RGB image with the size of 32×128 pixels. The output is the scores of 10 categories
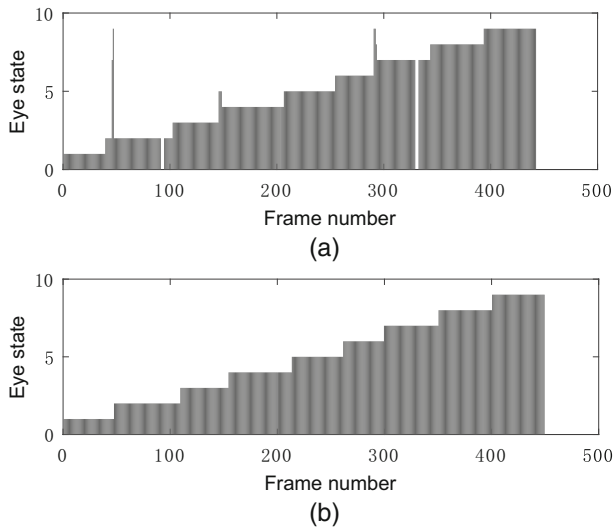
Fig. 8 The states of the eyes in each frame of the video sequence (29 fps). Numbers 1 to 9 represent the 9 directions and 0 means the state of closed eyes. In the video, the participant moves their eyes from numbers 1 to 9 and then closes their eyes. Natural blinks have occurred during the process. The diagrams **a** and **b** show the results before and after the filter respectively

frame. The noise lasts 3 to 5 frames which are 0.1–0.17 seconds. So it is important to distinguish voluntary and involuntary blinks and filter noise caused by involuntary blinks, saccade and momentary false estimation to avoid false pointing and selection. Our method is that we set an array recording estimated states in the last 16 frames and the array is updated every time when a new frame comes. The earliest state in the array is popped. The most frequent state in the array is set as the current state. So normally only when eyes fixate at a direction for more than 8 frames, the direction can be recognized as the current state, while momentary noise (less than 8 frames) does not influence correct pointing and selections. Figure 8 compares the estimated states before and after the filter. As natural blinks usually last less than 5 frames and saccade lasts less than 4 frames, noise caused by them can be filtered. The length of the array is set according to the frame rate and the noise duration. Usually, it is set as more than twice the length of the longest noise duration.
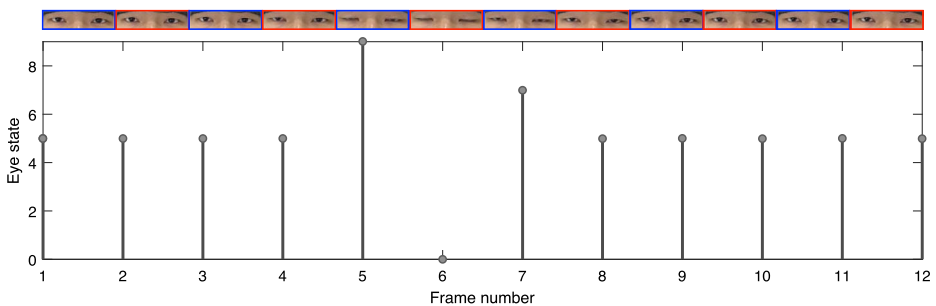

Fig. 9 A blink process and estimated states of each frame (29 fps)

4.3 Input method

Our input method runs in two modes: screen mode and off-screen mode. For screen mode, the camera is table mounted. A keyboard is shown on the screen. For off-screen mode, the camera is head mounted. The camera is right in front of the eyes and users can use our system in any position. In our interface, letters are distributed in buttons 2 to 9 and functional keys are in button 1, which accords with most layouts on mobile phones. After a button is selected by a blink in the main interface, the interface jumps to the secondary interface and then users select the letters or numbers. After a number or a letter is entered, the interface jumps back to the main interface (See Fig. 10).

In our text entry system, when gaze moves to a new direction two types of feedbacks will be given to users. In the off-screen mode, the feedback is sound. The system reads out the letters or numbers that the users are looking at. When users close their eyes to make a selection, a 'click' sound is played, which means a voluntary blink has been recognized and users can then open their eyes and continue to use. Users do not have to close eyes for a long time to distinguish voluntary and involuntary blinks, which can increase input speed. In screen mode, the feedbacks include sound and real-time visual effects on the screen. The buttons that the users are fixating change color which is synchronized with the sound. With feedbacks, users can know whether the correct direction is estimated and wrong selections can be avoided.
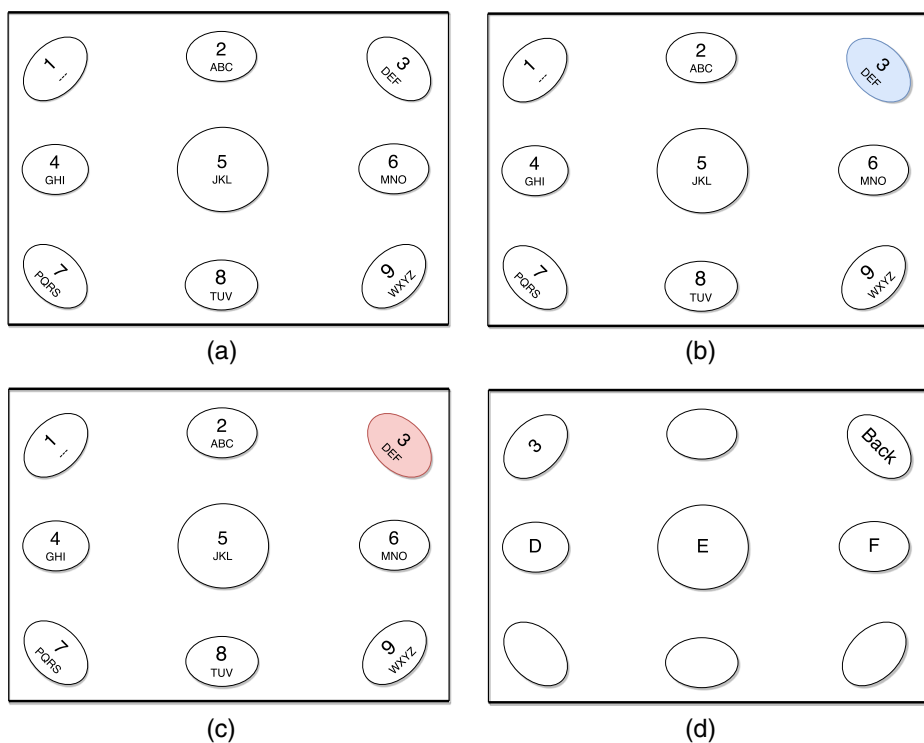

Fig. 10 Main interface (**a**), main interface when users look at number 3 (**b**), main interface when users blink to select number 3 (**c**) and secondary interface of button 3 (**d**)

5 Experiments and discussion

In this section, we made two test sets from known users and unknown users to evaluate the performance of our CNN model and compared that with other models. We also tested the effectiveness of our data augmentation methods. Comparing the performance of gaze estimation using images of single eye and both eyes is also included in this part. We then invited five users to operate our input method and recorded their mean error rate and text input speed.

5.1 Gaze estimation evaluation

We compared the accuracy of our model with LeNet-5 model and Support Vector Machine (SVM). LeNet-5 is a CNN model that has a great success in recognizing handwritten characters. We trained a SVM with a polynomial kernel using HOG features [4] of the same training images. We also experimented to test the performance of our model with input images of single eye. The only difference is that the size of input single-eye image is 32 × 64 pixels. We cropped images of single eye from our dataset and the estimation of both left and right eyes used the same model. We trained our CNN model for 200 epochs with the batch size of 200. Initial learning rate was 0.0433 and decreased by 3% in each epoch. The initial learning rate was chosen by random search which had the best performance in training the network in the first 20 epochs. The test result is shown in Table 1.

As is shown in the table, our model and data augmentation methods are effective for gaze estimation of known and unknown people. We observed the incorrect estimation images and found that the incorrect estimated direction is usually near the correct direction. Thus, we infer that it is likely to confuse the CNN model to make an incorrect estimation when users watch the edge of two adjacent directions. When we calculated the top 2 error, the error rate decreased to 0.2%. This kind of error is acceptable because when using the system to input words, users always watch the middle part of every direction rather than look at the edge of the direction. Moreover, we can watch the results on the screen and listen to the feedback sound at real time. Based on feedbacks, we can make a quick adjustment if the recognized direction is not desired. For example, when the system misunderstands left-up as left, we can adjust gaze closer to up direction.

According to observing on the test results, we find our model with single-eye data has a higher error rate. We examined all false estimated human eye data and found the following

Table 1 The top 1 and top 2 accuracy of different models using different test sets and using or not using our data augmentation methods

Model	Data augmentation	Test set of known users		Test set of unknown users	
		Top1 Acc.	Top2 Acc.	Top1 Acc.	Top2 Acc.
Our model	No	82.44	95.61	78.34	95.85
Our model	Yes	95.01	99.80	91.71	98.39
Lenet-5	No	68.46	90.02	67.28	88.94
Lenet-5	Yes	80.44	94.81	81.57	93.32
SVM	No	70.66	87.62	70.51	89.63
Our model (SE)	Yes	76.27	93.31	73.15	92.84

Our model (SE) denotes our model with single-eye data

Fig. 11 Images of both eyes where left and right eyes have different estimated results. The estimated directions of left and right eyes are indicated by arrows on the images.

three reasons. First, single eye data is more sensitive to slight head movement. Head tilting significantly influences the estimation result. Second, under low resolution and weak lighting conditions, images of single eyes may be too blurry for estimation so that even human cannot recognize the direction. Third, left and right eyes are not always synchronized. The estimated results of two eyes are different but both are reasonable. Figure 11 shows images of eyes in which left and right eyes have different estimation results.

5.2 Gaze typing experiments

We invited 5 participants to test our text entry system. They all had experiences with the T9 input method on bar phones. Before they used our system, we made a calibration for every user. The calibration method was very simple. On screen mode, the screen size is 30 cm × 40 cm. We used a common webcam. The camera was table mounted. We only required the participant to keep eye level with the center of the screen, then adjusted the distance between the participant and the screen to make sure gaze can be correctly estimated when they looked at the nine regions on the screen. We recorded the distance for each user for later test. The calibration took 10 to 20 seconds for each user and the distance between the head and the screen was around 25 to 40 cm. On off-screen mode, we just needed to adjust the distance between the camera and eyes to make sure the camera captures the whole eye region. The distance was around 20 to 30 cm.

Our test consisted of three sessions. First, we introduced our system to participants and then they had 10 minutes to familiarize with it before the first test session. There was one hour of training time before the second session where they trained half an hour for each mode. There was a one-day interval between the second and third session. They were required to keep our keyboard layout in mind before they took part in the third test. They were free to use our system for training during the one-day interval. The reason for this arrangement was that we intended to simulate three kinds of users: novices, regular users and experienced users. The length of the filter array (FAL) used for our final system was 16 which was set by observing the duration of natural blinks. It must be long enough to distinguish natural blinks and voluntary blinks but it can not be too long otherwise it will decrease the input speed. So we experimented three filter array lengths which were 6, 16, 26. We tested screen mode at first and off-screen mode afterwards in each session, and in each mode, our system was test for three times with the three filter array lengths receptively. They were asked to type in sample words they knew the spelling of. When participants forgot how to spell the words, we gave them support. They were required to type in as many words as possible during 2 minutes. When they made a mistake, they just continued on without a need to delete the incorrect typing. Participants did not need to type a space and spaces were not counted in total letters. The error rate is the ratio of incorrect letters to the total number of input letters. Figures 12 and 13 illustrate the mean input speed and error rate of two modes respectively. We can observe that in both modes there was a great increase
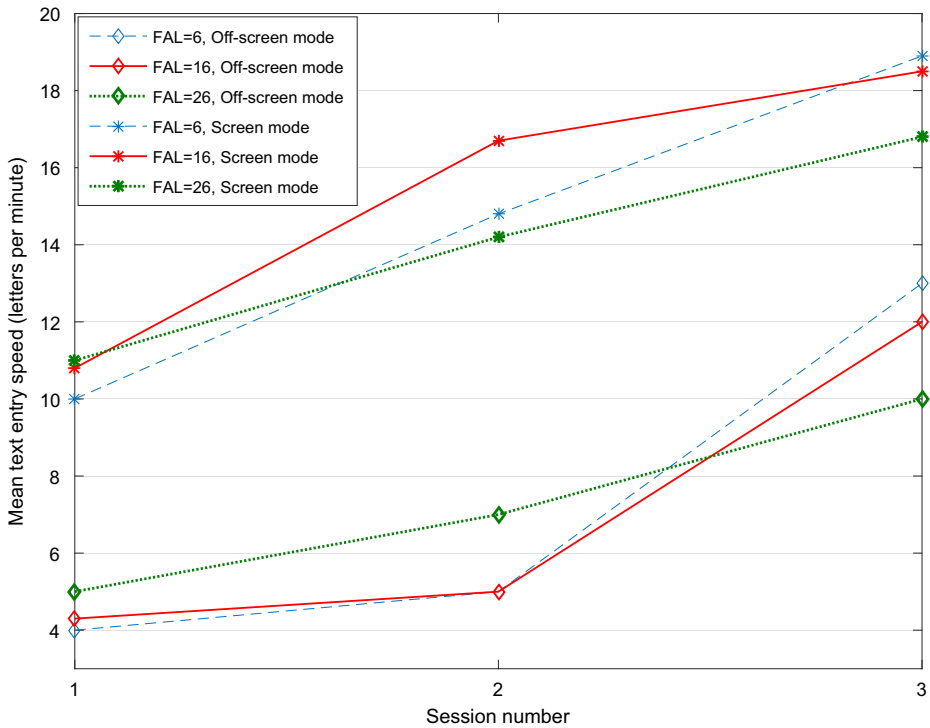
Fig. 12 Mean text input speed of 5 participants in three sessions

in the speed of text input and a decrease in error rates with the increasing of session numbers. The input speed of screen mode was faster than that of off-screen mode for all the participants. Experienced users had a comparable speed of two modes. The fastest speed of screen mode was 20 letters per minute and 18 letters per minute for off-screen mode among all participants. We noticed that although shorter filter array length (6) increased the input speed, it also increased the error rate meanwhile. It made the system sensitive to saccade and could not effectively filter natural blinks. As a consequence, participants would easily be distracted by frequent sound and visual feedbacks and made wrong selections, particularly in off-screen mode. While longer array length (26) resulted in that users had to fixate and closed eyes for a longer time to point and make selections.

5.3 Discussion

Participants stated that it was easier to choose letters on screen mode as they were able to find the positions of desired letters on screen easily. On the other hand, they had to spend time recalling the letters' position when they used the off-screen mode. When they forgot the positions of letters, they had to try directions one by one and rely on the sound feedback to select the correct letter. After they remembered the letters' position, the speed of input increased greatly. Incorrect text input was all due to users moving their eyes before they blink to choose the correct letter. We can also see that by session three users were able to accurately type in words letter by letter with very low error rate. The input speed of 20
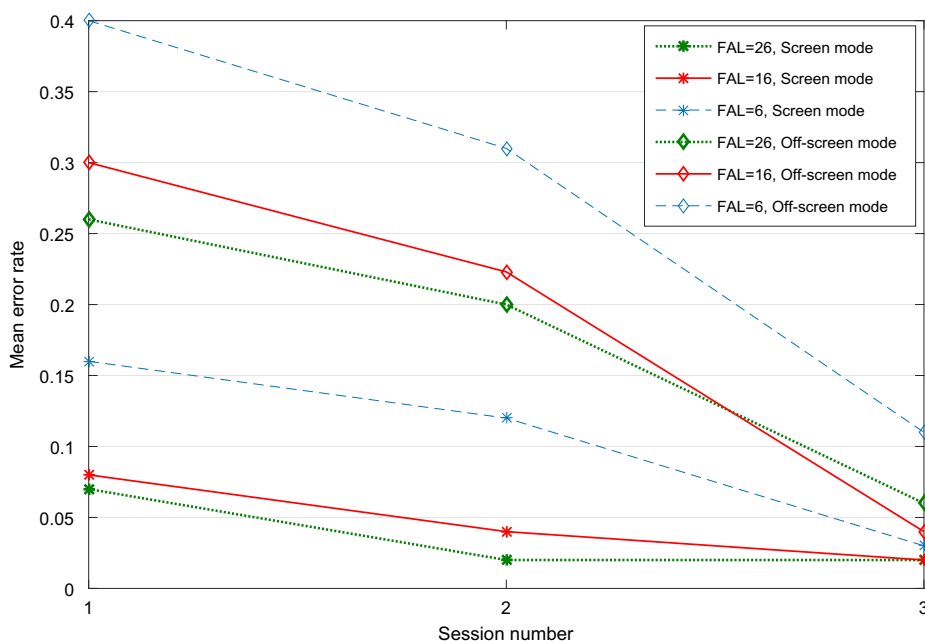
Fig. 13 Mean error rate of 5 participants in three sessions

letters per minute is quite ideal for disabled people in daily life. They can express words, like 'eat', 'toilet' and 'headache' in seconds, which is very convenient for communication.

When people use T9 input method on bar phones, they usually do not need to type each letter of a word because on bar phones, language models are often used to help users type in words efficiently. Users only need to choose the letter groups. For example, to type in 'hello', they only need to select buttons 4, 3, 5, 5 and 6 successively where letters 'h', 'e', 'l', 'l' and 'o' are located respectively and the language model can predict the desired words. So only one selection is needed to type in a letter on average. As shown in our experiments, our input method can reach up to 20 letters per minute which means we make 40 selections. So if language model is introduced in our input method, text entry speed can double and can reach up to 40 letters per minute theoretically. Also, there are many language models based on 9-key T9 text input keyboard on bar phones so our vision based text input method is expected to support efficient text input of more languages.

6 Conclusion

We have designed the vision based text entry method that users can input text by eye movement and blinks. We designed the system by solving both eye tracking problem and text input problem. We selected the popular bar phones' 9-key T9 keyboard layout as the interface. A large scale dataset was made which was collected by using several capturing devices and covered different lighting conditions, locations and time. We used images of both eyes to avoid single eye images' synchronization problems, which increased the estimation accuracy. For applications purposes, we defined nine gaze directions and trained a CNN model for gaze estimation which can precisely estimate known people's and unknown people's

gaze. Data augmentation methods we used provided robustness and increased the estimation accuracy. Our text input method utilizes low cost devices, simplifies the calibration steps and effectively filters noise caused by saccade, natural blinks and momentary false estimation during the operation process. Our input method is able to run in two modes: off-screen mode and screen mode. On off-screen mode, users can arbitrarily move and the device is portable. The text input speed of screen mode can reach 20 letters per minute and off-screen mode can reach 18 letters per minute with low error rate. Our future work is to introduce language model and add word prediction and completion functions to our current system and to train a head pose invariant gaze estimator.

References

1. Chen J, Ji Q (2008) 3D gaze estimation with a single camera without ir illumination. In: 19th international conference on pattern recognition, 2008. ICPR 2008. IEEE, pp 1–4
2. Cui J, Liu Y, Xu Y, Zhao H, Zha H (2013) Tracking generic human motion via fusion of low-and high-dimensional approaches. IEEE Trans Syst Man Cybern Syst 43(4):996–1002
3. Dagnelie G (2011) Visual prosthetics: physiology, bioengineering. Rehabilitation
4. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: IEEE computer society conference on computer vision and pattern recognition, 2005. CVPR 2005, vol 1. IEEE, pp 886–893
5. Dohse KKC (2007) Effects of field of view and stereo graphics on memory in immersive command and control. Iowa State University
6. Fejtova M, Fejt J, Lhotska L (2004) Controlling a PC by eye movements: the MEMREC project. In: Proceedings of the 9th international conference on computers helping people with special needs, ICCHP 2004, Paris, France. Springer Berlin, Heidelberg, pp 770–773
7. Gips J, Olivieri P (1996) Eagleeyes: an eye control system for persons with disabilities. In: The eleventh international conference on technology and persons with disabilities, pp 1–15
8. Grauman K, Betke M, Lombardi J, Gips J, Bradski GR (2003) Communication via eye blinks and eyebrow raises: Video-based human-computer interfaces. Univ Access Inf Soc 2(4):359–373
9. Hansen DW, Hansen JP, Nielsen M, Johansen AS, Stegmann MB (2002) Eye typing using markov and active appearance models. In: Proceedings of the sixth IEEE workshop on applications of computer vision, 2002 (WACV 2002). IEEE, pp 132–136
10. Hansen DW, Ji Q (2010) In the eye of the beholder: a survey of models for eyes and gaze. IEEE Trans Pattern Anal Mach Intell 32(3):478–500
11. Hutchinson TE, White KP, Martin WN, Reichert KC, Frey LA (1989) Human-computer interaction using eye-gaze input. IEEE Trans Syst Man Cybern 19(6):1527–1534
12. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167
13. Istance HO, Spinner C, Howarth PA (1996) Providing motor impaired users with access to standard graphical user interface (gui) software via eye-based interaction. In: Proceedings of the 1st european conference on disability, virtual reality and associated technologies (ECDVRAT'96)
14. Krafka K, Khosla A, Kellnhofer P, Kannan H, Bhandarkar S, Matusik W, Torralba A (2016) Eye tracking for everyone. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2176–2184
15. Królak A, Strumiłło P (2012) Eye-blink detection system for human–computer interaction. Univ Access Inf Soc 11(4):409–419
16. Liu L, Cheng L, Liu Y, Jia Y, Rosenblum DS (2016) Recognizing complex activities by a probabilistic interval-based model. In: AAAI, vol 30, pp 1266–1272
17. Liu Y, Cui J, Zhao H, Zha H (2012) Fusion of low-and high-dimensional approaches by trackers sampling for generic human motion tracking. In: 21st international conference on pattern recognition (ICPR), 2012. IEEE, pp 898–901

18. Liu Y, Nie L, Han L, Zhang L, Rosenblum DS (2015) Action2activity: recognizing complex activities from sensor data. In: IJCAI, pp 1617–1623
19. Liu Y, Nie L, Liu L, Rosenblum DS (2016) From action to activity: sensor-based activity recognition. Neurocomputing 181:108–115
20. Liu Y, Zhang X, Cui J, Wu C, Aghajan H, Zha H (2010) Visual analysis of child-adult interactive behaviors in video sequences. In: 16th international conference on virtual systems and multimedia (VSMM), 2010. IEEE, pp 26–33
21. Lu Y, Wei Y, Liu L, Zhong J, Sun L, Liu Y (2017) Towards unsupervised physical activity recognition using smartphone accelerometers. Multimedia Tools and Applications 76(8):10,701–10,719
22. Majaranta P, Ahola UK, Špakov O (2009) Fast gaze typing with an adjustable dwell time. In: Proceedings of the SIGCHI conference on human factors in computing systems. ACM, pp 357–360
23. Majaranta P, Bulling A (2014) Eye tracking and eye-based human–computer interaction. In: Advances in physiological computing. Springer, pp 39–65
24. Majaranta P, Räihä KJ (2002) Twenty years of eye typing: systems and design issues. In: Proceedings of the 2002 symposium on eye tracking research & applications. ACM, pp 15–22
25. Majaranta P, Räihä KJ (2007) Text entry by gaze: utilizing eye-tracking. Text entry systems: mobility, accessibility, universality, pp 175–187
26. Morimoto CH, Amir A, Flickner M (2002) Detecting eye position and gaze from a single camera and 2 light sources. In: Proceedings of the 16th international conference on pattern recognition, 2002, vol 4. IEEE, pp 314–317
27. Preoţiuc-Pietro D, Liu Y, Hopkins D, Ungar L (2017) Beyond binary labels: political ideology prediction of twitter users. In: Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers), vol 1, pp 729–740
28. Silfverberg M, MacKenzie IS, Korhonen P (2000) Predicting text entry speed on mobile phones. In: Proceedings of the SIGCHI conference on human factors in computing systems. ACM, pp 9–16
29. Tuisku O, Majaranta P, Isokoski P, Räihä KJ (2008) Now dasher! dash away!: longitudinal study of fast text entry by eye gaze. In: Proceedings of the 2008 symposium on eye tracking research & applications. ACM, pp 19–26
30. Viola P, Jones M (2001) Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition, 2001. CVPR 2001, vol 1. IEEE, pp I–I
31. Wobbrock JO, Rubinstein J, Sawyer M, Duchowski AT (2007) Not typing but writing: eye-based text entry using letter-like gestures. In: Proceedings of the conference on communications by gaze interaction (COGAIN), pp 61–64
32. Zhang X, Sugano Y, Fritz M, Bulling A (2015) Appearance-based gaze estimation in the wild. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4511–4520

Chi Zhang received his Bachelor's degree in School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China. His current interests include pattern recognition and intelligent information processing.
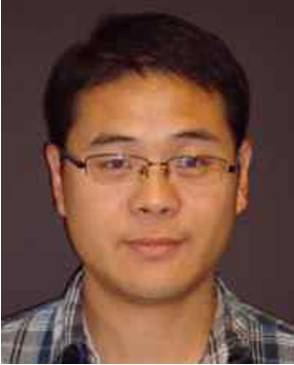
Rui Yao is an associate professor in School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China. His current research interests include computer vision and machine learning.
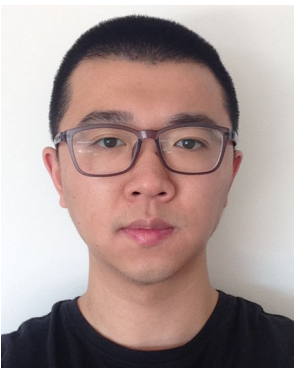
Jinpeng Cai is currently pursuing the M.S. degree from College of Engineering & Computer Science, Australian National University, Canberra, Australia. His current interests include pattern recognition and machine learning.