iDec: Real-time eye state classification via web cam

Miguel Picallo, Heerad Farkhoor, Thomas von der Ohe

December 14, 2012

Abstract

This paper presents a real-time eye state classifier via a simple web cam. With the help of 3-means color clustering of detected eyes via OpenCV, features for Softmax Regression are derived and used to classify in real-time four different eye positions: looking straight, looking left, looking upward, looking right. Given that two eyes are detected in a face, the system classifies the eye-states with an accuracy ranging from 95-99%. We portray the systems value for both video content providers and consumers, including the ability to analyze content quality and viewer attention over time.

1 Introduction

Online video education is steadily becoming more important. Reasons are not only geographical differences between the lecturer and the student, but also new teaching concepts such as the so called "flipped classroom". In the U.S. alone already 60% of the 50 million students use online education[3]. Nonetheless, many concerns still exist towards this new teaching technique:

- Parents and teachers do not know whether or not the child really watches the video lectures or if it just lets the video run in the background.
- Content providers, such as the Khan Academy, have little insights into the quality of certain parts of their videos. Overall ratings might not consider that certain sequences are not well explained.
- The student himself gets often tired or distracted while watching a video, which makes him miss certain parts of the video.

In the following paper, we will introduce a new application that might help solve the mentioned concerns.

2 iDec - Idea

By using real-time webcam image processing, we attempt to determine the direction in which a person is looking relative to the screen. Parents and teachers could supervise whether a child really watches a lecture. Content providers could get an attention rate (or content quality graph) over the time of the video, derived by the percentage of viewers focused on the video over time. Lastly, the video could automatically stop once the students gets distracted or tired.

Companies surveyed are excited about the potential of such a product. [1]

> This is the future. Everyone wants to know if their content is working.
> (Khan Academy)

> Assessing the learning value of a video would be extremely valuable.
> (edshelf)

> The potential for automated curation is huge. (SmarterCookie)

> A really powerful idea.
> (TenMarks Education)

3 Previous work

Eye-tracking is an area of computer vision and object detection that has been researched in the past years. Achieving a non-sophisticated, but reliable tool for eye-tracking is one of the challenges of this field. There already exist some commercialized products and guides to build eye-tracking tool, some of them focusing on the pupil, given an accurate enough image of the eye, and relying on some additional equipment such as lighting. One idea used for eye-tracking is differentiating bright and dark zones of the eye[9]. K-means clustering has also been included in some algorithms to detect certain parts of the face or object[4][11], and algorithms based on k-means have even been used for eye-tracking[2].

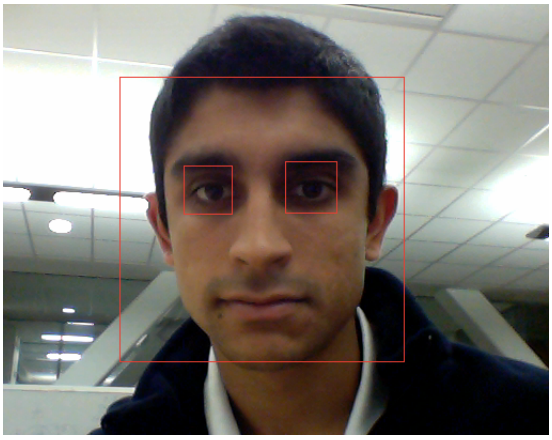

Figure 1: Real-time face and eye detection via a 640x480 webcam

These eye-tracking systems can then be used for several applications. Some of them have already been implemented to conduct psychological studies about attention and interest[6][8][11][2]. Most of them rely not only in the image of the eye, but also in the image that the eye is seeing, so that regions of interest can be identified.

We make use of OpenCV, one of the leading open source computer vision libraries. Its face and eye detection functions rely on the cv function called cvHaarDetectObjects, which uses machine learning tools to detect any object, given a training basis for that object. This particular function uses a method developed by Viola and Jones[10] and improved by Lienhart[5]. The method looks for subregions in the image which are likely to be the object that needs to be detected. First, it reduces the dimension of the features, and thus the computational time. Instead of all the intensities of all pixels, differences among intensities of certain regions are used as features. Then it uses a cascade classification, in which a region is classified as the object only if it goes through all the classifications. These classifications increase in complexity and the first ones are used to discard the regions that are clearly not the object, like the background of the image.

4 Implementation

4.1 Face and eye detection

We developed using the C++ OpenCV library an application which accesses the computer's webcam in real-time. Frame-by-frame, the program scans for faces via cvHaarDetectObjects. The upper-third of each face is then scanned using the same function, but with an eye classifier input. Cascading the eye detection into these two stages rather than scanning a full frame for eyes reduces computation time and the probability of a false positive – the face detection mechanism is more reliable and robust against backgrounds. Figure 1 demonstrates the program tracking both the face and eyes of a user.

4.2 Relevant eye image

Before extracting features from the cropped-out eyes, the image is further trimmed both horizontally and vertically, while being constrained to show the entirety of the sclera that is exposed directly around the iris. Vertical cropping removes portions of the eyebrows and shadows from the image, while the horizontal cropping corrects for a slight asymmetry in the eyes. Additionally, the relative variation between eye positions when looking in different directions is maximized, and the subsequently extracted features become more well-defined. The constraint ensures a clear distinction between each eye direction – especially between looking upward versus straight.

Accounting for facial tilt variation, the optimal cropping profiles were found to be: two-thirds of the horizontal span, offset by one-fourth from the left, and five-twelfths of the vertical span, offset by one-third from the top.

4.3 Collection of training data

With the same webcam used for classification, we collected for each of the four labels (looking straight, looking left, looking upward, looking right) approx-

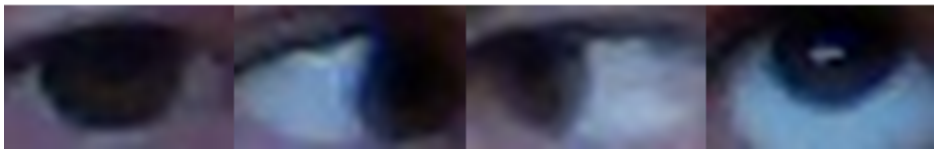

Figure 2: One example of each labeling taken from the training set

Figure 3: Eye images having undergone 3-means clustering with type-1 features marked

imately 400 relevant training images. These images were taken in various lighting conditions and for different eye and skin colors. We chose to omit a "looking downward" label because human eyes naturally close when looking down, and the Haar classifier does not reliably detect closed eyes.

Figure 2 demonstrates a training example for each labeling, cropped according to the specifications in the previous section.

4.4 Feature extraction via clustering

Two types of features were extracted from these cropped eye images:

1. Normalized Cartesian coordinates of the centers of mass of the iris and the sclera
2. Percentage of darker versus lighter pixels in each cell of a 3x3 grid of the eye

To extract these features, a given image's pixels were represented in three-dimensional (RGB) color space, and the K-means clustering algorithm with $K = 3$ was used to filter the image into three distinct colors. Given the nature of the eye, these colors approximately represent the iris, the skin around the eye, and the sclera in order of darkest to lightest. To remove the skin from consideration, the second cluster is ignored. Each filtered pixel is then placed in its original location within the image.

Features of type-1 are calculated by finding the normalized center of mass of every pixel belonging to clusters 1 and 3. After dividing the image into a 3x3 grid, the percentages of cluster 1 out of cluster 1 and 3 pixels are calculated in each region, forming the second set of features.

Figure 3 demonstrates the images from Figure 2 having undergone 3-means clustering, with the dark and light cluster centers of mass labeled in red and blue, respectively. The separation between these two points gives a clear distinction between each of the four classes.

4.5 Softmax regression

With these features extracted from our training data, we calculated the log-likelihood-optimal parameter values for softmax regression[7], which enables classification of a eye's feature set into one of four directions. Maximizing the log-likelihood function, the optimal parameters values θ^* can be derived using gradient ascent:

$$\theta_r^{(t+1)} := \theta_r^{(t)} + \alpha \nabla_{\theta_r} l(\theta^{(t)})$$

with

$$\nabla_{\theta_r} l(\theta^{(t)}) = \sum_{i=1}^{m} (1\{y^{(i)} = r\} - h_{\theta^{(t)}}(x^i)_r) x^{(i)}$$

$$h_\theta(x)_r = e^{\theta_r^T} / \sum_{i=1}^{k} e^{\theta_i^T x}$$

$$h_\theta(x) = [h_\theta(x)_1 \dots h_\theta(x)_k]$$

We used $\alpha = 0.05$. With our training set size of 400, batch gradient ascent was implemented without incurring a computational penalty, and parameters were derived for every (label,feature) pair.

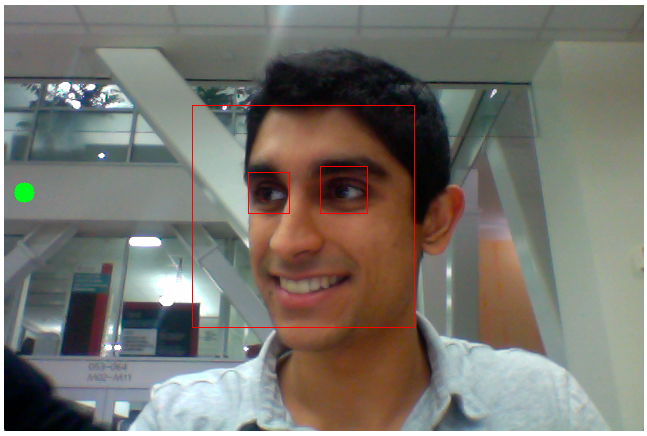

Figure 4: Real-time classification

4.6 Real-time classification

The application accesses the webcam in real-time, processing each frame by detecting and cropping face then eyes, extracting features from the eyes, and inputting the features for each eye into the softmax function with our derived parameters. Depending on the application's success in detecting a face, two eyes, and classifying the eyes in the same direction, the following algorithm is followed:

Data: Frame from webcam
Result: Frame's classification
if *no face detected* **then return** *no face*;
if *no eyes detected* **then return** *down*;
if *two eyes detected* **then**
  if *same label* **then return** *label*;
else
  return *previous label*;
end

Algorithm 1: Real-time classification

As mentioned above, because human eyes tend to shut when looking downward and the Haar classifier does not easily find closed eyes, we chose to interpret a frame with a detected face but no detected eyes as a "looking down" frame. To smooth the output, classifications are only updated when exactly two eyes with the same classification are detected. Otherwise, the classifier could have interpreted non-eye objects as eyes, and we would have no means of determining which detected "eyes" to use.

Figure 4 shows this algorithm in action, with a green dot correctly marking the direction in which the user is looking.

5 Error analysis

We split our error analysis into three stages in order to identify the major bottleneck in the application's performance:

1. Percentage of frames with a face detected
2. Percentage of frames with two eyes detected, given a face is detected
3. Percentage of frames with correct classification, given two eyes detected

We tested each stage in various lighting conditions, and tested the third stage using **a)** only type-1 features, and **b)** both type-1 and 2 features. The results are in Figure 5. Additionally, we performed leave-one-out cross-validation (LOOCV) with 400 training examples for both cases. Adding in type-2 features into the feature set improves the LOOCV error rate from 3.5% to 1%.

Clearly eye-detection is the lowest-performing and least robust stage of the classification algorithm, with only 80% of frames with a detected face having two detected eyes in bad lighting conditions. It is also apparent that by adding type-2 features into the feature set, we significantly improve straight and upward classification.

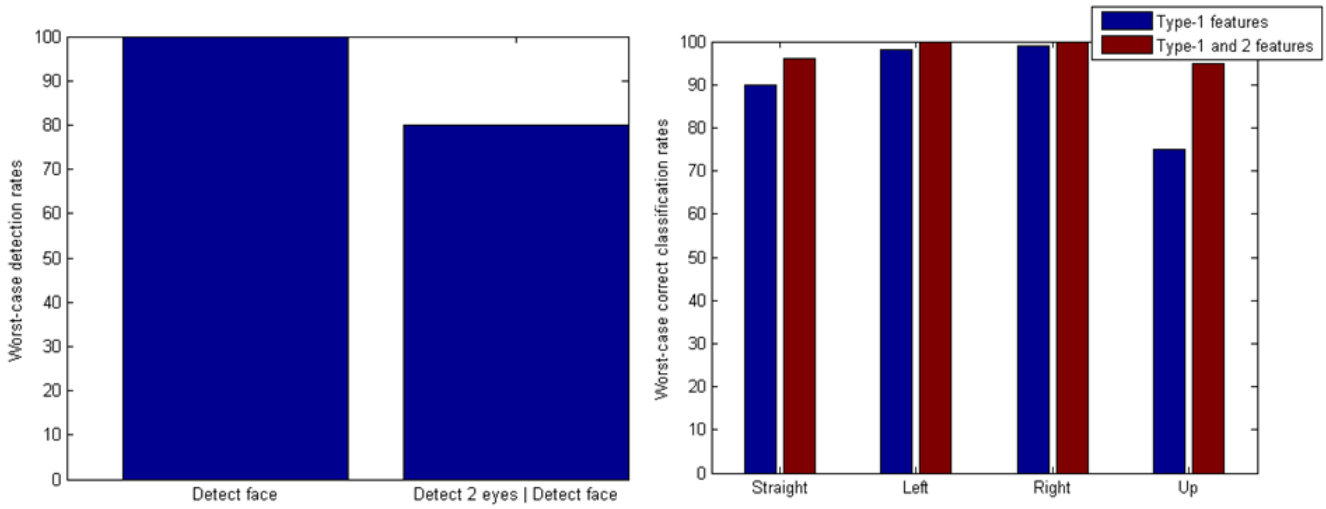

Figure 5: Worst-case detection and classification rates for face, eyes, and directions

6 Next steps

Based upon our error analysis results and algorithmic compromises, we identify three topics of interest for future work.

Because eye detection is more of a bottleneck than is classification, an important step would be to generate a custom eye classifier cascade for use in cvHaarDetectObjects, which is robust against lighting conditions. To solve the issue with the "looking downward" classification, a second, nearly-closed eye classifier cascade can also be developed. Finally, we note that while left/right classification works well both when the head is turned and when the eyes are angled, the same is not true for upward classification. Our classifier only notices upward-angled eyes. Adding a binary classifier to the entire detected face that determines whether the face is angled would solve this problem.

In general, our goal is to develop a product in close feedback loops with the different stakeholders in order to ensure product-market fit.

References

[1] Eduflip team of the Stanford StartUpWeekend 10/2012. Not all members are authors.

[2] S. Amershi C. Conati, C. Merten and K. Muldner. Using eye-tracking data for high-level user modeling in adaptive interfaces, 2007.

[3] National Center for Educational Statistics, 2011.

[4] R. Kaucic and A.Blake. Accurate, real-time, unadorned lip tracking. In *in ICCV*, pages 370–375, 1998.

[5] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *IEEE ICIP*, pages 900–903, 2002.

[6] M. Couper M. Galesic, R. Tourangeau and F. Conrad. Eye-tracking data : New insights on response order effects and other signs of cognitive shortcuts in survey responding. *Public Opinion Quarterly*, 72(5), 2008.

[7] A. Ng. Supervised learning. CS229 lecture notes.

[8] O. Oyekoya and F. Stentiford. Exploring human eye behaviour using a model of visual attention. In *International Conference on Pattern Recognition*, pages 23–26, 2004.

[9] J.Doyle R. Argue, M.Boardman and G.Hickey. Building a low-cost device to track eye movement, 2004.

[10] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE CVPR*, pages 511–518, 2001.

[11] Q. Wang. Eye location in face images for driver fatigue monitoring. In *in ITS*, pages 322–325, 2006.