Sleep Mode Detection for Smart TV using Face and Motion Detection

Suwon Lee[1], and Yong-Ho Seo[2]
[1] Department of Computer Science, Gyeongsang National University
501 Jinju-daero, Jinju-si, Gyeongsangnam-do - Republic of Korea
[e-mail: leesuwon@gnu.ac.kr]
[2] Department of Intelligent Robot Engineering, Mokwon University
88 Daehak-ro, Seo-gu, Daejeon - Republic of Korea
[e-mail: yhseo@mokwon.ac.kr]
*Corresponding author: Yong-Ho Seo

Received September 28, 2017; revised December 24, 2017; accepted March 10, 2018; published July 31, 2018

Abstract

Sleep mode detection is a significant power management and green computing feature. However, it is difficult for televisions and smart TVs to detect deactivation events because we can use these devices without the assistance of an input device. In this paper, we propose a robust method for smart TVs to detect deactivation events based on a visual combination of face and motion detection. The results of experiments conducted indicate that the proposed method significantly reduces incorrect face detection and human absence by means of motion detection. The results also show that the proposed method is robust and effective for smart TVs to reduce power consumption.

Keywords: Sleep mode detection, smart TV, face detection, motion detection, green computing

http://doi.org/10.3837/tiis.2018.07.018 **ISSN : 1976-7277**

1. Introduction

For a long time, the primary methods for electricity generation have been fossil fuels, nuclear energy, and renewable power. Among these, fossil fuels such as coal and gas are the predominant means for electricity generation.

Most scientists agree that emissions of pollutants and greenhouse gases from fossil fuel-based electricity generation account for a significant portion of the world's greenhouse gas emissions.

For this reason, green computing has received much attention in various research fields during recent years. Green computing technologies utilize human-centric and eco-friendly techniques. In the drive to advance green computing technology, many researchers focus on effective, energy-saving usage of electronic appliances. Reduction of energy consumption is accomplished by lowering electric power usage, or by utilizing effective software algorithms to eliminate unnecessary consumption.

Sleep mode detection is one of several green computing technologies for electronic devices such as personal computers, smartphones, and smart TVs. With personal computers and smartphones, it is easy to detect deactivation events because these devices allow input from peripherals such as keyboards, mice, and touchscreens. However, with respect to televisions and smart TVs, it is difficult to detect a deactivation event because the user can interact with these devices without the assistance of an input device.

In this paper, we propose a robust method for smart TVs to detect deactivation events based on a visual combination of face and motion detection. The human face is the key feature when detecting a deactivation event. However, current face detection technologies are not intended for commercial products due to shortcomings in the detection rate and number of false alarms when used in such applications. To tackle this problem, we adopt skin color filtering and background verification schemes. Additionally, we use motion detection to enhance the ability to find humans missed through face detection.

We organized the remainder of this paper as follows: Section 2 describes the background of our research and Section 3 describes the algorithm used in our proposed technique, as well as aspects of the system design. Sections 4 and Section 5 present the details of color-filtering based face detection and motion detection. Section 6 discusses the experiments conducted to ascertain the effectiveness of our algorithms. Section 7 concludes this paper.

2. Background

Since 1980, the primary methods for generation of electricity have been fossil fuels, nuclear energy, and renewable power, as shown in **Fig. 1**. Among these, power generation using fossil fuels is the predominant means by which energy is obtained. Coal and gas are the primary types of fossil fuels used for the generation of electricity. Most scientists agree that emissions of pollutants and greenhouse gases from fossil fuel-based electricity generation account for a significant portion of the world's greenhouse gas emissions.

As a result, green computing has received much attention in various research fields during recent years. Green computing technology, also referred to as low power technology, utilizes human-centric and eco-friendly techniques. In the drive to advance green computing technology, many researchers focus on effective, energy-saving usage of electronic appliances. Reduction of energy consumption is accomplished by lowering electric power usage, or by

utilizing effective software algorithms to eliminate unnecessary consumption. Efforts to reduce energy consumption in smart TVs, spotlighted as the next generation of televisions, is closely linked to face detection algorithms. The goal of these algorithms is to ascertain the intention of the television users [1-3]. However, this technique has disadvantages related to the effectiveness of the algorithms used. First, the technique requires real-time face detection for continuous verification of a human presence. Additionally, this technique suffers from false alarms that prevent the smart TVs from reacting appropriately to deactivation events. To solve these problems, this paper proposes an effective method for detection of deactivation events. In our proposed technique, face detection is not required in every frame. As a result, relatively few frames are required for image analysis. Furthermore, when there are false alarms, it can remove incorrect information by comparing previously stored background images. In addition, motion detection in front of the smart TV complements the face detection system. If motion is detected in front of the smart TV, no deactivation event occurs.

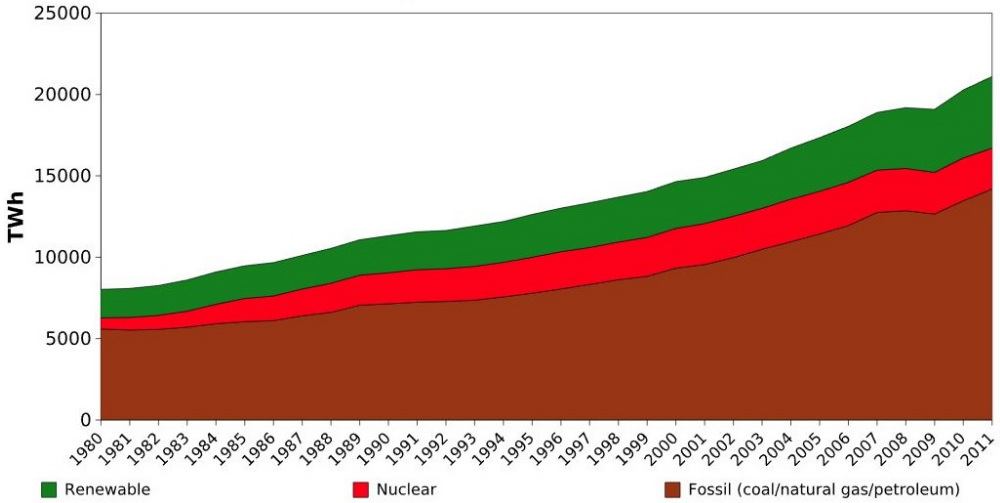

Fig. 1. Annual electricity net generation in the world (from EIA)

3. Overview

In this section, we describe the procedure followed by our proposed method. We present the complete deactivation detection procedure in **Fig. 2**. We check the deactivation event n times during waiting time ω. The sampling number n is a design parameter that determines the power consumption level and the deactivation detection accuracy. At every detection step, the proposed method analyzes the image acquired at time t for face detection, and analyzes image sequences from $t - \delta$ to $t + \delta$ for motion detection. If either face or motion is detected, the deactivation timer (DT) is initialized to zero. If neither a face nor motion is detected, the DT is increased by ω/n. Once the timer becomes larger than ω, a deactivation event is triggered and the system goes into sleep mode.

To reduce the number of false alarms and the computation cost, we adopt skin color filtering and background verification schemes as proposed in [4]. Efficient filtering is accomplished by means of sparse region scanning based on the facial color density in a given region. To handle color variations from the user, we enhance the facial color model by adding faces in difficult lighting conditions to the training dataset. Although we reduce false detections by using facial color filtering, false face detection still occurs in real environments. Therefore, we adopt

background verification, in which detected faces are further verified through comparisons with the same region in the background image; this leads to more robust face detection.

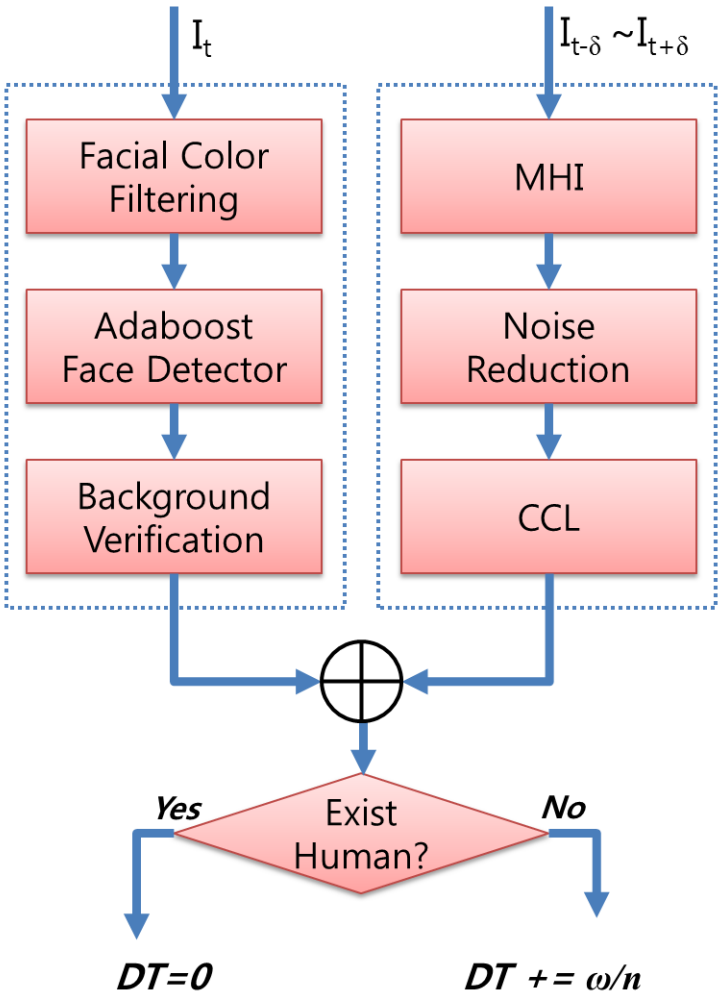

Fig. 2. Overall procedure followed by our proposed algorithm

To detect motion from image sequences, we adopt the representation of the motion history image (MHI) with the image set comprising $t - \delta$ to $t + \delta$. The MHI collapses an image sequence into a 2D image that captures spatial and temporal information pertaining to motion [5]. The MHI is known for its fast processing speed and its ability to represent short-duration motion. To reduce noise arising from a vision sensor or an illumination condition, we perform a morphology operation [6]. Next, we utilize connected component labeling (CCL) to find motion blobs that exceed a certain size.

4. Color-Filtering Based Face Detection

In this section, we review color-filtering based face detection [4]. To enhance the conventional face detector, we adopt a color-filtering based region scanning approach with a face/non-face

classifier. We base the classifier on facial color density at the face detection preprocessor. To use facial color density, we obtain a facial color filtered image and its integral image [7] to quickly obtain the density of the sub-window. We obtain a facial color filtered image using a facial color membership function that can reflect facial colors under various illumination environments:

$$M(color) = \frac{max_{I_i \in face}(p_i(color))}{p(color)}, \tag{1}$$

where $p(color)$ is the color probability distribution of all images, and $p_i(color)$ refers to the color probability distribution of an image I_i. If we use the traditional facial color likelihood model obtained from a Bayesian rule, some exceptional face colors taken in difficult lighting conditions will have a relatively low likelihood of matching. This condition manifests because these colors occur sparsely in the sample space. By taking the maximum value of the merged color probability distributions of facial images, rare facial colors in the sample space may have a high likelihood of matching. Thus, by inserting some sample face images containing difficult lighting conditions, we can easily include that color into the space of the facial color. To obtain $M(color)$, all facial and non-facial images are represented in the HSV color coordinate and a histogram over hue. Saturation is calculated to obtain $p(color)$ and $p_i(color)$. $M(color)$ is stored in a lookup table indexed by hue and saturation, and is convolved with a 2D Gaussian function for generalization. Using $M(color)$, the facial color membership value of each pixel can be obtained.

Using this membership information, we obtain a facial color filter image, whose pixel value is 1 when its color belongs to a facial color. To calculate facial color information efficiently, we adopt the integral image to this facial color filtered image. Using this facial color integral image, we can calculate the facial color density of a region with a relatively light computational load. From this density information, we determine the scan interval, which denotes the position of the next face candidate sub-window:

$$si_h = \begin{cases} \mu & \text{if } density(r) = 1 \\ \tau(1 - density(r)/\emptyset) & \text{if } density(r) < 1' \end{cases} \tag{2}$$

where si_h is the horizontal scan interval, μ is the minimum scan interval, τ is the width of the rectangular region r, and $\emptyset$ is the minimum facial color density. We determine the color density value as 0.55 through experimentation. The proposed detector skips regions that have a lower density than the lower bound of the facial color density by changing the scan interval. We can apply a similar method to the vertical direction. By passing the scan interval, the detector can avoid assessing non-candidate regions. We then adopt a face/non-face classifier using facial color density that quickly rejects non-faces with a high rejection rate at the initial stage of the AdaBoost face detector. If the region density is lower than a certain threshold, we reject that region. The proposed method reduces the overall face detection computation time and eliminates false alarms.

Although we reduce false alarms, false detection of faces still occurs in real environments. To avoid deactivation of the TV when a false detection occurs, we adopt background verification, in which all detected regions are further verified by means of comparison with the same region in the background image. We verify the last background image against other periodically archived background images taken during sleep mode or when the power is off. To judge that the region is a real face or background we adopt face verification protocol.

The most common process of face verification includes face representation as a main step. Each face image is represented as a vector format; a similarity score is then computed between vectors to make a decision. To represent a face image in vector format, the intensity magnitude of each pixel is not directly used because its value is very sensitive to environment changes, such as illumination, shadowing, and noise. Instead, the value is encoded into the other one by considering the relations between each pixel and its neighboring pixels. As encoding method, local binary patterns (LBP) [8], which encode micro-structures of images as invariant to monotonic photometric changes have been often used. The encoded images are then divided into a grid of patches to preserve certain spatial information between grids. A histogram vector is then computed on each patch, and all histogram vectors are concatenated into a high-dimensional vector to form the representation of the whole face image. Because the face images are represented in a single vector format, it is intuitive and efficient to compare two face images (e.g., face verification) or train classifiers such as support vector machines (e.g., face identification). However, the current histogram vector completely ignores the importance of each code in the face image because the histogram vector is computed by only counting each of the codes in the corresponding bin of the histogram. This degrades the discriminative power of the final representation. To address this problem, we applied the inverse document frequency (idf) weighting scheme to the face representation. In information retrieval and text mining, idf is used as a weighting factor to reflect a word's degree of importance to the corpus. It measures how much information the word provides, which helps to control for the fact that some words are generally more common than others. Idf weighting is typically used with term frequency (tf), which reflects the raw frequency of a term in a document, or so-called tf–idf weighting. With tf–idf weighting, documents are compared with consideration of both the frequency and importance of each word. The tf–idf weighting scheme is often used by search engines and has been recently proven effective in computer vision tasks, such as image retrieval [9].

A simplified outline of our method is illustrated in **Fig. 3**. First, the idf vector is trained on training samples. N face images are given for training; then, through the basic face representation process, they are represented as a D -dimensional vector (called the base vector) while forming an N-by-D matrix, where D is defined by the code number of the local encoder and the grid density (e.g., D = 256 codes × 120 patches = 30,720). From the matrix, the D -dimensional idf vector is calculated as follows:

$$\mathrm{idf}_i = \log\left(\alpha \frac{N}{1+n_i}\right),\ i = 0, \cdots, D-1, \tag{3}$$

where n_i is the number of face images (in our context) in which the term i appears (i.e., i -th element is not zero), and α is a constant smoothing factor, which is introduced to control the relative impact of a particular term compared with the other terms. The base of the log function mathematically constitutes a constant multiplicative factor, which means it does not affect the overall result. Once the idf vector is calculated, any base vector can simply be weighted by the idf vector in a post-processing step only if the base vector is represented in the same way it was used in the training step (i.e., the same local encoder, code number, and grid density). Because each value of D elements in the idf vector connotes the importance of the corresponding code in relation to the other codes, base vectors are weighted by a per-element scaled product with the idf vector. By the weighting, each value of D elements in the weighted vector reflects the importance of the corresponding code, and the weighted vector becomes a final representation of our method.

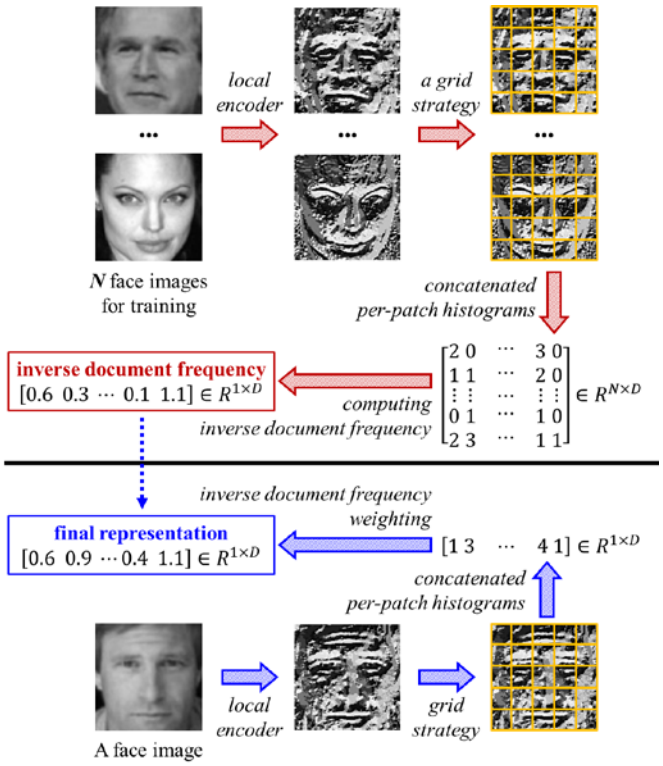

Fig. 3. Simplified idf weighting scheme for face representation
Top: training the idf vector; bottom: applying the idf vector to face representation

Fig. 4 shows some examples of face representations. **Fig. 4 (a)** shows the facial region in the real environment and its histogram representation. **Fig. 4 (b)**, **Fig. 4 (c)**, and **Fig. 4 (d)** show the sample background regions in the background image and their respective histogram representations. This example shows that the false alarm region in the background image is distinguishable from a real face. Hence, we can eliminate false alarms by means of background verification.

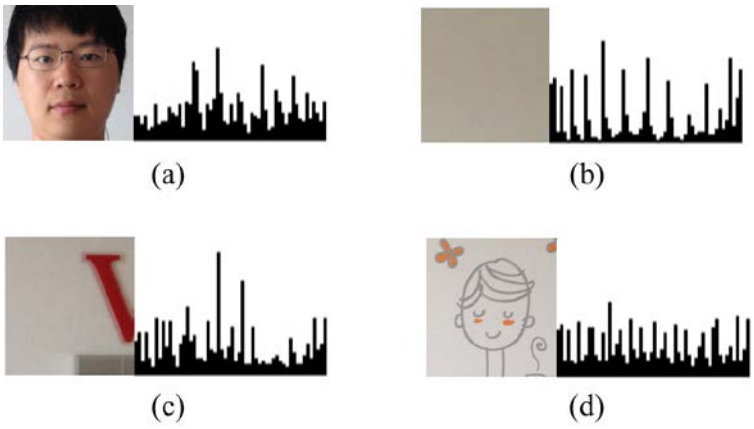

Fig. 4. LBP histogram samples

5. Motion Detection

To compensate for the probability of humans being missed by the face detector, we need to detect human motion in front of the TV. To detect motion from image sequences, we adopt a representation of the MHI [5] with difference images from $t-\delta$ to $t+\delta$. First, we use the difference image to ascertain the changes between the current frame and the previous frame. We calculate the difference between the current and previous frames by finding the intensity difference between each pixel in each image. The difference image is a binary image whose pixel value is 1 when its intensity difference is greater than the difference threshold value, as follows:

$$I_d^t = \begin{cases} 1, & \text{if } |I^t(x,y) - I^{t-1}(x,y)| > \theta \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

where $I^t(x,y)$ is the input image at time t, and θ denotes the intensity threshold. The proposed system then adopts a representation of the MHI to the difference image for archiving motion datum from $t-\delta$ to $t+\delta$. The MHI collapses an image sequence into a 2D image that captures spatial and temporal information about motion. The MHI is known for its fast processing speed and ability to represent short duration movement. The MHI at time t is updated as follows:

$$MHI_\gamma^t = \begin{cases} t/\gamma & \text{if } I_d^t(x,y) \neq 0 \\ MHI_\gamma^{t-1}(x,y) & \text{otherwise} \end{cases}, \tag{5}$$

where γ is the number of images used for the collapse, and $I_d^t(x,y)$ is the difference image at time t.

Noise from a vision sensor or an illumination condition in the MHI may exist. To reduce this noise, we perform the morphology operation, opening [6]. Opening is the composite operation of erosion and dilation. In other words, dilation follows erosion. Opening is a basic workhorse of morphological noise removal. Opening removes small noise from the foreground of an image, placing them in the background. The basic effect of the erosion operator is to erode away the boundaries of foreground pixels. Thus, the areas of the foreground pixels shrink in size, resulting in efficient elimination of small noise from MHIs. The basic effect of dilation is to enlarge the areas of foreground pixels at their borders. The areas of foreground pixels thus grow in size, so the object shrunk by the erosion operator can be similarly restored. Thus, the opening operator can effectively remove small or thin noise from the image. The equations of the erosion and dilation operator are as follows:

$$dst(x,y) = \max_{(x',y'):element(x',y')\neq 0} src(x+x', y+y'), \tag{6}$$

$$dst(x,y) = \min_{(x',y'):element(x',y')\neq 0} src(x+x', y+y'), \tag{7}$$

where *src* and *dst* refer to the source and destination images, respectively, and *element* refers to the structuring element, which is the shape used to interact with the given source image. In this paper, we use a 3×3 rectangular shape as the structuring element.

After noise removal, CCL is performed to find motion blobs. CCL scans an image and groups its pixels into components based on pixel connectivity. All pixels in a connected component share similar pixel intensity values and are in some way connected to one other. Once all groups have been determined, we label each pixel according to its component assignment.

Fig. 5. Sample images from the Caltech dataset

6. Experimental Results

6.1 Face Detection Experiment

In this experiment, we evaluate our proposed face detection approach. To verify the facial-color-filtering based face detector with a large dataset, we used the Caltech 10,000 web faces dataset [10] as the test set. The Caltech dataset contains 7,092 color images and has 10,524 human faces of various resolutions and complexions in different settings as shown in **Fig. 5**. In addition, the coordinates of the eyes, nose, and the center of the mouth for each frontal face are provided to define ground truth. The test set consists of only color images from the database. This test set includes 5,525 color images and 8,382 faces. We applied our proposed method to an AdaBoost face detector, which is state-of-the-art in terms of detection rate and computational time. OpenCV supports the AdaBoost face detector used in this experiment. To detect both frontal and profile faces, we combined one frontal classifier and two profile classifiers sequentially.

Fig. 6. Face detection examples: AdaBoost (left), Facial color filtered image (middle), Proposed method (right)

Fig. 6 shows face detection examples. Blue rectangles signify the ground truth, red rectangles signify detected faces (true positives), and green rectangles signify false alarms (false positives). As shown in **Fig. 6**, we eliminated numerous false alarms using our proposed method.

We compared our proposed method with the AdaBoost face detector using detection ratio (DR), false alarm (FA) per image, and computation time. We conducted the experiment on a Pentium 4 2.4 GHz single-core PC. As shown in **Table 1**, a 61% reduction in the overall false alarm rate is achieved. The detection rate of our proposed approach was slightly higher than that of the AdaBoost face detector, as some false alarms led to missed detections. Moreover, the computational time reduced significantly, which may affect power consumption.

Table 1. Face detection results using the Caltech dataset

	AdaBoost	**Proposed method**
DR (%)	86.77	86.95
FA per image	2.13	1.31
Time (s)	2.473	1.684

Fig. 7. Four versions of images posted on the LFW website according to different alignment methods

6.2 Face Verification Experiment

We evaluated our face verification method through a face verification test on the labeled faces in the wild (LFW) dataset [11, 12]. The LFW database contains a collection of annotated faces gathered from news articles on the Internet using Viola-Jones face detection [7]. Each face in the database is labeled with the name of the person pictured; it is the only label information the database provides. Therefore, because researchers cannot perform experiments on specific conditions, they must simultaneously address all conditions mentioned above. This makes use of the database extremely challenging. The LFW dataset provides four different versions of face images, including the original and three different types of aligned images according to different alignment methods, as shown in **Fig. 7**. Of these methods, we used the aligned version of the LFW dataset [13] in all experiments. For face representations, face images were encoded by LBP; histogram vectors were then computed on 120 non-overlapping patches (i.e., 8×15) that were concatenated, which resulted in a 30,720-dimensional vector.

We evaluated our method by comparing it to the baseline features. By following the standard training and evaluation protocols specified in [11], the mean classification accuracy was calculated using ten-fold cross-validation. For the cross-validation, five sets were used to compute the idf vector, and four sets were used to compute the global threshold of cosine similarity for the two-class problem (i.e., determining whether it was the same person in two face images). The remaining one was used for the evaluation. As shown in **Fig. 8**, compared with the baseline features, the idf weighting scheme produced significant gain in accuracy. The final performance was heavily influenced by the smoothing factor. This is because the smoothing factor controls the relative impact of a particular term compared to the other terms. The best-performing parameter, which differed according to the encoding method, was 0.66 and we used that parameter configuration for our system.

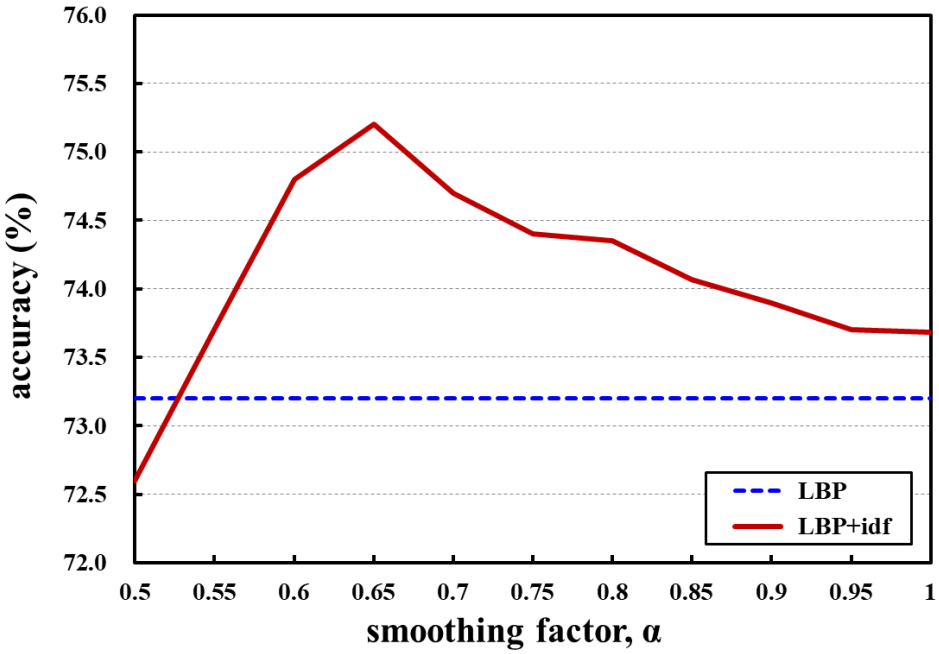

Fig. 8. Face verification experiment results

6.3 Power Consumption Experiment

To measure power consumption, we conducted simulation tests. We used an LG 27-inch LCD TV, and a Logitech webcam. To simulate a smart TV and process our proposed algorithm, we used a 2.4 GHz single-core PC. To measure the power consumption, we attached a watt-hour meter to the both TV and the PC. **Fig. 9** shows the equipment used in the experiment.

Fig. 9. Equipment used in the experiment: TV, webcam, watt-hour meter

We measured the power consumed by the TV before the experiment to determine whether our proposed system can reduce power consumption. We subsequently measured the power consumed over a two-hour period. We also measured the power consumption of the TV with our proposed sleep mode detection system active over another two-hour period. The waiting time ω was set at 10 min, and with the sampling number n set at five. In our scenario, two users watched the TV for 50 minutes in a room, and then went outside. Ten minutes after the users left the room, the TV turned off automatically. **Table 2** shows the results of this experiment.

Table 2. TV power consumption results

	Without sleep mode detection	**With sleep mode detection**
Watts (W)	74	35
Watt-hours (Wh)	146	70

As shown in **Table 2**, our proposed system reduced power consumption by half, compared to the standard case. Although we used a small LCD TV in this experiment, if we had used a PDP TV or a larger TV, the proposed system would have reduced power consumption by much more. In this experiment, we used a relatively short waiting time for convenience. By changing the waiting time and sampling number, the provider can adjust the parameters of the proposed method between efficiency and robustness.

We also measured the power consumed by the PC while running our proposed algorithm to account for any increase in power consumption. First, we measured the power consumed by the PC over a two-hour period, without the proposed algorithm. We then measured power consumption over a subsequent two-hour period, while running the proposed algorithm.

Finally, we computed power consumption with the proposed algorithm by subtracting the former from the latter. **Table 3** shows the results obtained.

Table 3. Power consumption of the proposed algorighm

	PC without algorithm	PC with algorithm
Watts (W)	140	142
Watt-hours (Wh)	280	284

As shown in **Table 3**, the power consumption due to the proposed algorithm was 2W per hour, which is negligible when compared with the power saved.

Fig. 10 shows sample face and motion detection results. In this figure, the red rectangles signify detected faces and green rectangle signify detected motion.

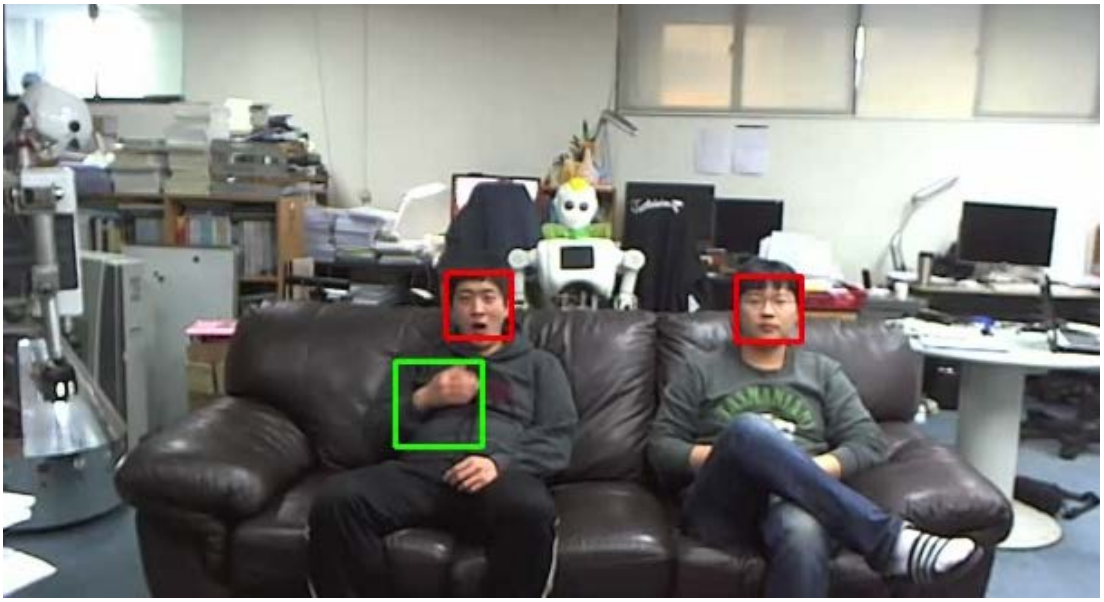

Fig. 10. Sample face (red) and motion (green) detection result

7. Conclusion

In this paper, we proposed a vision-based method to detect deactivation events for sleep mode detection for smart TVs. To detect deactivation events, we use a vision-based approach that combines facial and motion detection. To reduce the false alarm rate of the face detector, we adopted facial color filtering and background verification schemes. To enhance discovery of human faces missed through face detection, we adopted a motion detection scheme.

We presented face detection and power consumption experiments. We conducted a face detection experiment using a large dataset with real environments. The proposed approach significantly reduces the false alarm rate of face detection and compensates for the probability of humans missed by means of motion detection. The results of the power consumption experiment show that our proposed approach can significantly reduce the power consumed by smart TVs.

Our proposed method can only detect upright faces; therefore, we are currently expanding the proposed method to cover faces that appear at an angle.

References

[1] Ryo Ariizumi, Shigeo Kaneda and Hirohide Haga, "Energy saving of TV by face detection," in *Proc. of 1st Int. Conf. on Pervasive Technologies Related to Assistive Environments*, pp. 1-8, July 16-18, 2008. Article (CrossRefLink).

[2] Christoph Czepa, Shelley Buchinger, Helmut Hlavacs, Ewald Hotop and Yohann Pitrey, "Towards an energy-efficient attention-aware mobile video player with sensor and face detection support," in *Proc. of 13th IEEE Int. Symp. on a World of Wireless, Mobile and Multimedia Networks*, pp. 1-6, June 25-28, 2012. Article (CrossRefLink).

[3] Shruti Patil, Yu Chen and Tajana Simunic Rosing, "GazeTube: Gaze-Based Adaptive Video Playback for Bandwidth and Power Optimizations," in *Proc. of IEEE Global Communications Conference*, pp. 1-6, December 6-10, 2015. Article (CrossRefLink).

[4] Yeong Nam Chae, Ji-nyun Chung and Hyun Seung Yang, "Color filtering-based efficient face detection," in *Proc. of 19th Int. Conf. on Pattern Recognition*, pp. 1-4, December 8-11, 2008. Article (CrossRefLink).

[5] Aaron F. Bobick and James W. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257-267, March, 2001. Article (CrossRefLink).

[6] Robert M. Haralick, Stanley R. Sternberg and Xinhua Zhuang, "Image Analysis Using Mathematical Morphology," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp. 532-550, July, 1987. Article (CrossRefLink).

[7] Paul Viola and Michael Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 511-518, December 8-14, 2001. Article (CrossRefLink).

[8] Timo Ahonen, Abdenour Hadid and Matti Pietikainen, "Face Description with Local Binary Patterns: Application to Face Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2037-2041, December, 2006. Article (CrossRefLink).

[9] Josef Sivic and Andrew Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. of IEEE Int. Conf. on Computer Vision*, pp. 1470-1477, October 13-16, 2003. Article (CrossRefLink).

[10] Anelia Angelova, Yaser Abu-Mostafa and Pietro Perona, "Pruning Training Sets for Learning of Object Categories," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 494-501, June 20-25, 2005. Article (CrossRefLink).

[11] Gary B. Huang, Manu Ramesh, Tamara Berg and Erik Learned-Miller, "Labeled faces in the wild: a database for studying face recognition in unconstrained environments," *Technical Report 07-49*, University of Massachusetts, Amherst, 2007.

[12] Gary B. Huang and Erik Learned-Miller, "Labeled faces in the wild: updates and new reporting procedures," *Technical Report 14-003*, University of Massachusetts, Amherst, 2014.

[13] Lior Wolf, Tal Hassner and Yaniv Taigman, "Effective Unconstrained Face Recognition by Combining Multiple Descriptors and Learned Background Statistics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 10, pp. 1978-1990, October, 2011. Article (CrossRefLink).

Suwon Lee is an assistant professor of Dept. of Computer Science at Gyeongsang National University. He received his B.S. degree in Computer Engineering at Kyungpook National University, and his M.S. and Ph.D. degrees in Computer Science at KAIST. His research interests include computer vision, machine learning, augmented reality and human computer interaction.

Yong-Ho Seo received his B.S. and M.S. degrees from the Department of Electrical Engineering and Computer Science, KAIST, in 1999 and 2001, respectively. He also received a PhD degree at the Artificial Intelligence and Media Laboratory, KAIST, in 2007. He was an Intern Researcher at the Robotics Group, Microsoft Research, Redmond, WA in 2007. He was a consultant at Qualcomm CDMA Technologies, San Diego, CA in 2008. He is currently a Professor of the Department of Intelligent Robot Engineering, Mokwon University. His research interests include humanoid robot, human-robot interaction, robot vision and wearable computing.