J Electr Eng Technol Vol. 9, No. 2: 707-720, 2014
http://dx.doi.org/10.5370/JEET.2014.9.2.707

ISSN(Print) 1975-0102
ISSN(Online) 2093-7423

A Fast and Accurate Face Tracking Scheme by using Depth Information in Addition to Texture Information

Dong-Wook Kim†, Woo-Youl Kim*, Jisang Yoo and Young-Ho Seo*****

Abstract – This paper proposes a face tracking scheme that is a combination of a face detection algorithm and a face tracking algorithm. The proposed face detection algorithm basically uses the Adaboost algorithm, but the amount of search area is dramatically reduced, by using skin color and motion information in the depth map. Also, we propose a face tracking algorithm that uses a template matching method with depth information only. It also includes an early termination scheme, by a spiral search for template matching, which reduces the operation time with small loss in accuracy. It also incorporates an additional simple refinement process to make the loss in accuracy smaller. When the face tracking scheme fails to track the face, it automatically goes back to the face detection scheme, to find a new face to track. The two schemes are experimented with some home-made test sequences, and some in public. The experimental results are compared to show that they outperform the existing methods in accuracy and speed. Also we show some trade-offs between the tracking accuracy and the execution time for broader application.

Keywords: Face detection, Face tracking, Adaboost algorithm, Template resizing, Early termination

1. Introduction

Detection and/or tracking one or more objects (especially parts of the human body) have been researched for a long time. Their application areas have been diversely expanded from the computer vision area to security or surveillance systems, visual systems for robots, video conferencing, etc. One of the biggest applications is HCI (human-computer interface) by detecting and tracking human hand(s), body, face, or eyes, and is used in various areas, such as smart home systems [1]. In this paper, the target object is restricted to the human face(s).

Many previous works included both face detection and tracking, such that face detection would be a preprocessing to face tracking, as in this paper, but they are reviewed separately here. For face detection, the most frequently used or referred method is the so called Adaboost algorithm [2-4]. This method includes a training process, Haar-like feature extraction and classification, and cascade application of the classifiers, although it only applies to gray images. Many subsequent researches have referenced them [5-10]. [5, 6], and [9] proposed new classifiers to apply to a color image, and [7] designed a classifier that included skin color and eye-mouth features, as well as Haar-like features. [8] focused on the asymmetric features, to make a classifier for them. In [10], the result from local normalization and Gabor wavelet transform to solve the color variation problem was applied to Adaboost. Some also used Haar-like features, but designed different classifiers, and refined them by a support vector machine [11]. Many other methods have used some features on the face, such as nose and eyes [12], skin-color histogram [13], and edges of the components of the face [14]. Also, many works used skin color to detect the face. Most of them used chrominance components. [15] and [16] proposed chrominance distribution models of the face, and [17] proposed a statistical model of skin color. In addition, [18] focused on detecting various poses of face, and [19] proposed a method to detect any angle of the face, with multi-view images.

Most face tracking methods so far also used the factors that have been used in the face detection method, such as the component features in a face [20-23], appearance of the face [24-29], and skin color [30-34]. Among the feature-based tracking methods, [20] used the eyes, mouth, and chins as the landmarks, [21] tracked some features individually characterized by Gabor wavelets, and [22] used an inter-frame motion inference algorithm to track the features. [23] used silhouette features, as well as semantic features to on-line track the features. Appearance-based tracking basically used face shape or appearance [24, 25]. But [26] additionally used the contour of the face to cover large motions of the face, and [27] proposed constraints to temporally match the face. [28] proposed a method to learn the appearance models online, and [29] used a

† Corresponding Author: Dept. of Electronic Materials Engineering, Kwangwoon University, Korea. (dwkim@kw.ac.kr)
* Dept. of Electronic Materials Engineering, Kwangwoon University, Korea. (wykim@kw.ac.kr)
** Dept. of Electronic Engineering, Kwangwoon University, Korea. (jsyoo@kw.ac.kr)
*** College of Liberal Arts, Kwangwoon University, Korea. (yhseo@kw.ac.kr)
Received: April 12, 2013; Accepted: July 22, 2013

condensation method for efficiency. Many face tracking schemes also used skin color [30-34]. [30] proposed a modeling method based on skin color, and [31] constructed a condensation algorithm based on skin color. [32] used color distribution models to overcome the problem caused by varying illumination. Some methods used other facial factors, in addition to the skin color, such as facial shape [33]. Also, some methods used a statistical model of skin color, by adopting a neural network to calculate the probability of skin color, with adaptive mean shift method for condensation [34]. Besides, [35] tracked various poses of face, by using a statistical model. [36] used a template matching method to track a face, by using depth as the template. It first found hands, and then found the face to be tracked, by using the hands as the support information.

This paper proposes a combination of a face detection scheme and a face tracking scheme, to find and track the face(s) seamlessly, even in the case that the human goes out of the image, or the scene changes. In our scheme, the face detection is performed at the beginning of face tracking, or when the tracked face disappears. It is a hybrid scheme that uses features, skin color, and motion. It basically uses the method in [2-4], the Adaboost method or Viola & Jones method. But we reduce the search area to apply the Adaboost method, with motion and skin color. Our face tracking scheme uses a template matching method with only depth information, as in [36]. But it directly tracks the face, without any auxiliary information. Also, it includes a template resizing scheme, to adapt to the change of the distance of the face. In addition, it includes an early termination scheme, to reduce the execution time to run in more than a real time, with minimizing the tracking accuracy. Thus, we will show that it can be used adaptively, by considering the complementation between the execution time and the tracking error.

This paper consists of six chapters. The next chapter explains the overall operation of our scheme. The proposed face detection scheme and face tracking scheme are explained in more detail separately, in Chapter 3 and Chapter 4, respectively. Chapter 5 is devoted to finding the necessary parameters, and experimenting the proposed schemes. Finally, Chapter 6 concludes this paper, based on the experimental results.

2. Overall Operation

The global operation of the face tracking algorithm proposed in this paper is shown as a flow graph in Fig. 1. As mentioned before, the main proposal is for a face tracking algorithm, but we also propose a scheme to reduce the calculation time of face detection. That is, the proposed face tracking method uses a template of the face(s), which consists of the position and depth information of the face, and the first template is extracted by the face detection process. But afterward the template is updated by the tracking scheme itself, except for the case when the face being tracked disappears from the image.

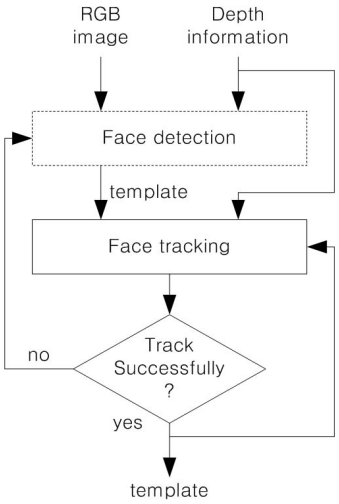


Fig. 1. Process flow of global operation

At the very first, the face detection process is performed. It uses both RGB image and depth information. Basically, it uses an existing method, the Adaboost algorithm [2-4], but the search area is reduced, by finding movement of the human, and skin color. Once a face is detected, its position (x-y coordinate), and the corresponding depth information (segment of depth map), are taken as the template to be used in the tracking process.

The tracking process uses a template matching method that finds a block matching the template. Here, we use only depth information for this process. It also includes a scheme to resize the template, as well as actual calculation for template matching. Also an early termination process is incorporated, to reduce the execution time.

The result from the tracking process, if it is successful, is the template of the current frame, and is used as the template for the next frame. But if it fails, it goes back to the detection process, to find a new face. This case is when the scene change or when the face being tracked disappears from the scope of the image. That is, in most cases, the scene does not change, or the tracked person remains in the scope of the image, so the detection process is performed only once, at the very start.

The image data we need for face detection and tracking is both RGB images and depth images. Here, we assume that the two kinds of images are given with the same resolution, such as the ones by Kinect from Microsoft.

3. Face Detection Algorithm

The processing flow of the proposed face detection scheme is as shown in Fig. 2. Basically it uses the Adaboost algorithm [2-4], but the area to search for the human face(s) is restricted, by using two consecutive depth images ($i\text{-}1^{th}$ and i^{th}), and the current RGB image (i^{th}).

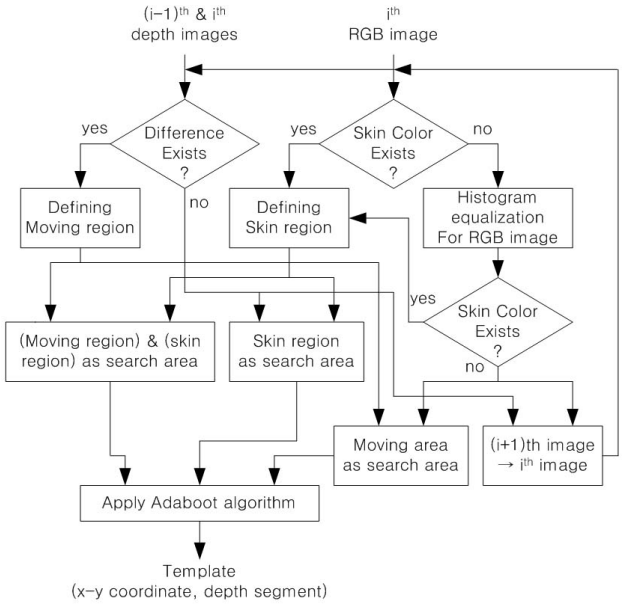


Fig. 2. The proposed face detection procedure

To do this, for the i[th] RGB image, if it contains skin color region is examined with Eq. (1), which was taken from the skin color reference in [37]. Here, only C_b and C_r components are used in the YC_bC_r color format.

$$S_i(x,y)=\begin{cases}1, & (77\le C_b\le 127)\cap(133\le C_r\le 173)\\ 0, & otherwise\end{cases} \quad (1)$$

The result by Eq. (1) is a binary image (we call it a skin image), in which the pixels with '1' value indicate human skin region. But a skin region may not be found, even if it exists (false negative error), which is mostly due to the illumination condition. Thus, in this case, we try once more, after adjusting the color distribution by a histogram equalization method [38]. If the first attempt, or the re-trial of skin color detection, finds any skin color pixel, we define the skin region as in Fig. 3. From the skin image $S_i(x,y)$, the vertical skin region image $SR_i^c(x,y)$ (horizontal skin

region image $SR_i^r(x,y)$) is obtained, such that if any pixel in a column j (row k) in $S_i(x,y)$ has value'1', all the pixels in the column j (in the row k) have '1' if $(j,k)\in S_i(x,y)(j,k)$. Then, the final skin region image $SR_i(x,y)$ is obtained, by taking the common parts of $SR_i^c(x,y)$ and $SR_i^r(x,y)$. Fig. 4 (a) shows the scheme to find the skin region image, where the horizontal and vertical gray regions correspond to $SR_i^r(x,y)$ and $SR_i^c(x,y)$, respectively, and the red-boxed regions are the defined skin region.

Meanwhile, for the depth information, a depth difference image $DD_i(x,y)$ between the i-1[th] and i[th] depth images is found as Eq. (2), where $D_i(x,y)$ is the depth value at (x,y) in the depth image i. The resulting image $DD_i(x,y)$ is also a binary one, where the region with '1' defines the region with movement.

$$DD_i(x,y)=\begin{cases}0, & D_i(x,y)=D_{i-1}(x,y)\\ 1, & D_i(x,y)\ne D_{i-1}(x,y)\end{cases} \quad (2)$$

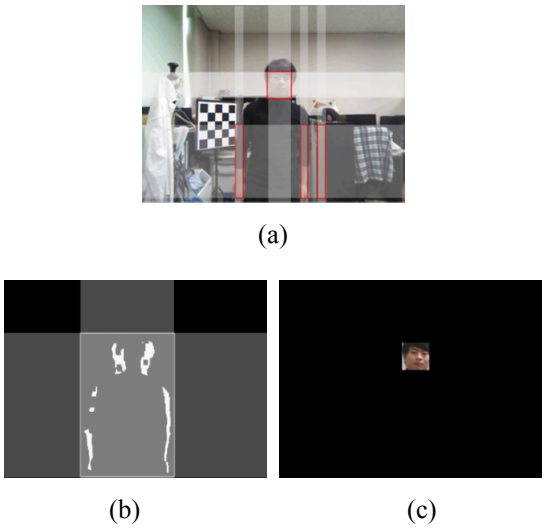
From the extracted depth difference image, the region map $MR_i(x,y)$ can also be found in the same way as Fig. 3 with $DD_i(x,y)$, $DD_i^c(x,y)$, and $DD_i^r(x,y)$, instead of $S_i(x,y)$, $SR_i^c(x,y)$, $SR_i^r(x,y)$, respectively. An example to find the movement region corresponding to Fig. 4 (a) is shown in Fig. 4 (b), where the horizontal and vertical gray regions correspond to $DD_i^r(x,y)$ and $DD_i^c(x,y)$, respectively, and the white-boxed region is the defined movement region.

With existence of the skin color region or movement region, there are four cases to define the area for the Adaboost algorithm to search face(s). When both skin color region and movement region exist, the search area is defined by the common regions of the skin region and the movement region. But when only skin region (movement region) exists, the skin region (the movement region) itself is defined as the search area. But when neither the skin region nor movement region exist, the process takes the next image frame, and performs the whole process again.

Procedure: *Define_SkinRegion*
Input: skin image $S_i(x,y)$ ($m\times n$)
Output: skin region image $SR_i(x,y)$ ($m\times n$)

```
begin {
  for j=0 to m-1 {
    if C_j = Σ_{k=1}^{n-1} S_i(j,k) > 0, then {
      for k=0 to n-1
        SR_i^c(j,k) = 1; }
    else {
      for k=0 to n-1
        SR_i^c(j,k) = 0; }}
  for k=0 to n-1 {
    if R_k = Σ_{j=1}^{m-1} S_i(j,k) > 0, then {
      for j=0 to n-1
        SR_i^r(j,k) = 1; }
    else {
      for j=0 to m-1
        SR_i^r(j,k) = 0; }}
  for j=0 to m-1 {
    for k=0 to n-1
      SR_i(j,k) = SR_i^c(j,k) ∩ SR_i^r(j,k); }}
end}
```

Fig. 3. Procedure to define a skin region



(a)

(b) (c)

Fig. 4. Examples of results from face detection processes: (a) skin region; (b) movement region; (c) detected face

Finally, the Adaboost algorithm is applied to the defined search area. An example of an RGB image segment of the finally detected face is shown in Fig. 4 (c), which corresponds to Figs. 4 (a) and (b).

The data from the face detection process is the coordinate and the size of the detected face, and its corresponding depth image segment. Because our purpose is more for face tracking, rather than face detection itself, the face information nearest to the camera is sent to the face tracking process, when more than one faces is detected in the detection process.

4. Face Tracking Algorithm

By taking the information of the detected face from the face detection process, or the previous face tracking process as template, the face tracking process is performed, as in Fig. 5. Because the proposed tracking process uses only depth information, it takes the next frame of the depth image as another input. Each step is explained in the following.

4.1 Template and search area re-sizing

The first step in the proposed face tracking scheme is to resize the template, and the search area, which is the area to find the face being tracked. Among the two, the template size is processed first, because the size of the search area depends on the resized template.

4.1.1 Template Re-sizing

If a human moves horizontally or vertically, the size of the face is nearly the same, but for back-and-forth

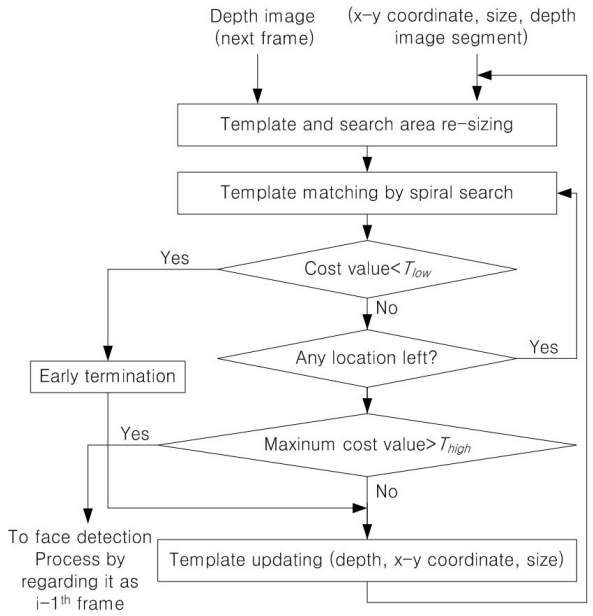


Fig. 5. The proposed face tracking procedure

movement it changes. So, the template that has been detected or updated in the detection process or the previous tracking process needs to be resized to fit to the face in the current frame.

(1) Relationship between depth and size

Because the size of an object in an image depends totally on its depth, change of the size of an object according to its depth is explained first. To do this it is necessary to define the way to express a depth. In general, a depth camera provides a real depth value in a floating-point form. Also, it usually has a distance range, within which the estimated value is reliable. Let's define the range as ($z_{R,min}$, $z_{R,max}$), and the depth value is expressed digitally by an n-bit word. Then, a real depth value z corresponds to a digital word depth is explained first. To do this it is necessary to define the way to express a depth. In general, a depth camera provides a real depth value in a floating-point form. Also, it usually has a distance range, within which the estimated value is reliable. Let's define the range as ($z_{R,min}$, $z_{R,max}$), and the depth value is expressed digitally by an n-bit word. Then, a real depth value z corresponds to a digital word Z', as Eq. (3)

$$Z' = \left(2^n - 1\right)\frac{z - z_{R,\min}}{z_{R,\max} - z_{R,\min}} \tag{3}$$

But in a typical depth map, a closer point has a larger value, by converting it as Eq. (4) or (5), and this paper also uses this value, Z.

$$Z = \left(2^n - 1\right) - Z' = \left(2^n - 1\right)\frac{z_{R,\max} - z}{z_{R,\max} - z_{R,\min}} \tag{4}$$

$$z = z_{R,\max} - \frac{z_{R,\max} - z_{R,\min}}{2^n - 1} Z \tag{5}$$

Now, when an object, whose real size is s and depth is z, has the size S_{sensor} in the image sensor of a camera whose focal length is f, the relationship between S_{sensor} and z or Z is as Eq. (6).

$$S_{sensor} = \frac{sf}{z} = \frac{\left(2^n - 1\right) f}{\left(2^n - 1\right) z_{R,\max} - \left(z_{R,\max} - z_{R,\min}\right) Z} s \tag{6}$$

If we assume that the pixel pitch in the image sensor is P_{sensor} , and number of pixels corresponding to S_{sensor} is N, the number of pixels in the real image is the same as N, and is found as Eq. (7).

$$N = \frac{S_{sensor}}{P_{sensor}} = \frac{\left(2^n - 1\right) f}{\left(2^n - 1\right) z_{R,\max} - \left(z_{R,\max} - z_{R,\min}\right) Z} \frac{s}{P_{sensor}} \tag{7}$$

Fig. 6 shows an example plots for the relationship in Eq. (7), and its measured result. Here, the dashed line is the plot from Eq. (7), the dots are the measured values, and the un-dashed line is the trend line. The error between the dashed line, and the measured value or the trend line, is because of the digitizing error (from the above equations, a function to delete the fractions should be applied), and the measuring errors.

(2) Face depth estimation and template re-sizing

To resize the template according to Fig. 6 or Eq. (7), the depth of a face in the current frame must be re-determined. For this, we define the depth template area as all the pixels in the i^{th} frame $D^i(x,y)$, corresponding to the ones in the previous template Ti-1 , as Eq. (8).

$$DT^i = \bigcup_{(x,y) \in T^{i-1}} D^i(x,y) \tag{8}$$

The scheme is shown in Fig. 7. The template and are

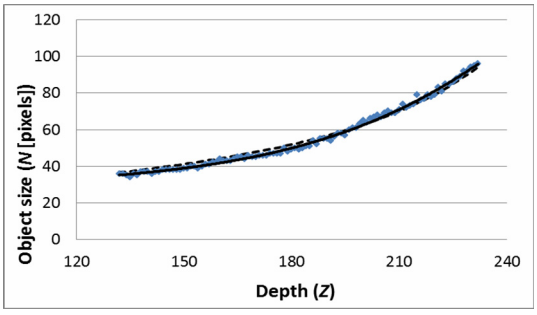


Fig. 6. Plotting for the relationship between size and depth

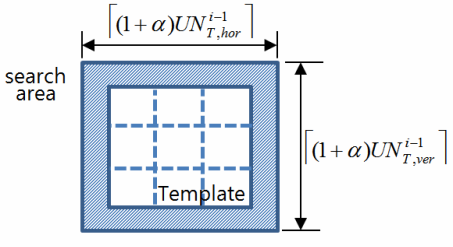


Fig. 7. Defining the search area

divided into $p \times q$ blocks (each block has $a \times b$ resolution), and average values $TA^{i-1}_{j,k}$ and $DTA^{i}_{j,k}$ for each block (block (j, k)) in the template and the depth template area, respectively, are calculated, to find the maximum values TA^{i-1}_{max} and DTA^{i}_{max}, respectively, as Eqs. (9) and (10). In this paper, *p* and *q* are determined empirically.

$$TA^{i-1}_{\max} = \max_{j,k=1}^{p,q} \left(TA^{i-1}_{j,k}\right) \tag{9}$$

$$DTA^{i}_{\max} = \max_{j,k=1}^{p,q} \left(DTA^{i}_{j,k}\right) \tag{10}$$

Then, the size of the updated template $UN^{i-1}_{T,X}$ is calculated with the size of the previous template $N^{i-1}_{T,X}$ as Eq. (11), and the template is resized accordingly (*X* is *hor* or *ver*, representing horizontal and vertical, respectively).

$$UN^{i-1}_{T,X} - N^{i-1}_{T,X} = \frac{\left(2^n - 1\right) f}{\left(2^n - 1\right) z_{R,\max} - \left(z_{R,\max} - z_{R,\min}\right)} \frac{s}{P_{sensor}} \left(\frac{1}{DTA_{\max}} - \frac{1}{TA^{i-1}_{\max}}\right) \tag{11}$$

4.1.2 Search area re-sizing

Although the template size has been updated, it is necessary to find the exact location of the face in the current frame, by searching the appropriate area, which we call the search area. The search area must be determined by considering the depth value of $UN^{i-1}_{T,X}$, and the maximum amount of face movement. The first has just been considered above. For the second, we have measured the maximum movement empirically with proper test sequences, and this will be explained in the experiment chapter. By considering both factors, we determine some extension of the template size $\lceil (1+\alpha) UN^{i-1}_{T,X} \rceil$ as the size of the search area, as shown in Fig. 7, where $\lceil x \rceil$ means the smallest integer not less than *x*, and X is *hor* (horizontal) or *ver* (vertical).

4.2 Template matching

Once the resized template (we call it just 'template', from now on) and the corresponding search area are determined, the exact face location is found by a template

matching. For this, we use the SAD (sum of absolute differences) value per pixel (PSAD) as the cost value, as Eq. (12), where K is the number of pixels in the template, (c_x, c_y) is the current position to be examined in the search area, and $D_T(i,j)$ ($D_{SA}(i,j)$) is the pixel value at (i, j) in the template (search area).

$$PSAD(c_x, c_y) = \frac{1}{K} \sum_{(i,j) \in template} \left| D_T(i,j) - D_{SA}(i + c_x, j + c_y) \right| \tag{12}$$

The final location of the matched face template SP_{opt} is determined as the pixel location satisfying Eq. (13)

$$SP_{opt} = (x, y) = \arg \min_{(i,j) \in template} \{ PSAD(i,j) \} \tag{13}$$

4.2.1 Early termination

The process of Eq. (13) needs to repeat the calculation of Eq. (12) as many times as the number of pixels in the search area, which might take too much time. The means to reduce this search time in our method is an early termination scheme, whereby the search process is terminated when a certain criterion is satisfied. Because the amount of the face movement in the assumed circumstances is usually small or none, search from smaller movement to larger is more proper. So we take a spiral search scheme, as in Fig. 8, which is an example with a search area of 5×5 [pixel^2]. The dashed arrows show the direction of search, and the numbers in the blocks (pixels) are the search order. Thus, if the early termination scheme is not applied, each of the full search sequence SS_{FS} is examined to find the pixel SP_{opt} , as Eq. (14), where the size of the search area is assumed as $m \times n$ [pixel^2].

$$SP_{opt} = \arg \min_{PSAD} \{ SS_{FS} \} = \arg \min_{PSAD} \{ SP_0, SP_1, SP_2, \cdots, SP_{m \times n - 1} \} \tag{14}$$

To terminate the examination earlier than the last pixel, the PSAD value of a pixel satisfies Eq. (14),

$$SP_{ET} = \arg \underset{SP_k}{first} \{ PSAD(SP_k) < T_{ET} \} \tag{15}$$

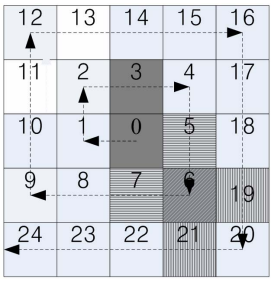


Fig. 8. Spiral search and early termination scheme

where, T_{ET} is the threshold value of PSAD that we assume for when the pixel SP_{ET} satisfying Eq. (14) is close enough to SP_{opt} , and $first\{x\}$ means the first position satisfying x. T_{ET} is determined empirically.

4.2.2 Sparse search and refinement

Even though the early termination scheme reduces the search time, it could be reduced more, if we might sacrifice accuracy a little more. This can be done by hopping several pixels, from the currently examined pixel, to the next one to be examined. In this case, the search sequence is as in Eq. (16), when the number of intervals between the current and the next is p, which is called the 'hopping distance'.

$$SS_p = \{ SP_0, SP_p, SP_{2p}, \cdots, SP_{qp}, \cdots \} \tag{16}$$

If the early termination scheme is not applied, and p=1, it is the same as SS_{FS}. If the early termination scheme is applied, and p=1, it is the same as when this sparse search scheme is not applied. Fig. 8 shows a case of p=3, where the dark pixels are the ones in the SS_3.

One more step is included in this sparse search. When p > 1, and a pixel (SP_{qp}) in the sequence satisfies Eq. (15) (early terminated), the immediate neighbor pixels are additionally examined. We call it a refinement process, and two cases are considered. The first is called 2-pixel refinement, and additionally examines two pixels (SP_{qp-1} , SP_{qp+1}) just before and just after SP_{qp} (horizontally striped),); while the second is called 4-pixel refinement, and two more pixels just outside of SP_{qp} (vertically striped) are examined. In either case, the pixel with the lowest PSAD value would be the final pixel, SP_{opt}. If a sparse search does not find a pixel satisfying Eq. (14), the one with minimum PSAD value is selected as the final result.

4.2.3 Feedback to face detection

If the face can no longer be tracked, in such a case that the human goes out of the screen, or is hidden by another object, our algorithm goes back to the face detection algorithm, to find another face. This is decided by a PSAD threshold value T_{FB}, as in Eq. (17)

$$\min_{sequecne} PSAD(SS_p) > T_{FB} \tag{17}$$

Eq. (17) is applied to any search sequence, no matter whether the early termination scheme, or sparse search, is applied, or not.

5. Experiments and Results

We have implemented the proposed face detection

algorithm and the face tracking algorithm, and conducted experiments with various test sequences. In these experiments, we used Microsoft Visual Studio 2010, and OpenCV Library 2.4.3 in the Windows operating system. The computer used in the experiments has an Intel Core i7 3.4GHz CPU, with 16GB RAM.

5.1 Determining parameters for face tracking

First, we have empirically determined the parameters defined in the previous chapter. For this, we used three home-made contents, whose names indicate the directions of movements. The information is in Table 1, and two representative images for each sequence are shown in Fig. 9. The speed of movement in each content was conducted maximally, to cover more than enough movement speed, compared with the assumed circumstances. All three contents are captured by Kinect® from Microsoft, so the resolution of both RGB image and depth image is 640×480.

5.1.1 Template segmentation for template resizing

In Fig. 7, we have segmented the template and the corresponding region of the current frame into $p \times q$ blocks. Because there can be so many combinations of p and q, we only take the cases of $p = q$ that are an odd number. The purpose of this segmentation is to resize the template to match the current face size. So, we have estimated the relative error of the resized template. In this experiment, we have used the faces extracted by applying the face detection algorithm, as the reference templates. The relative error, named as template resizing error, was calculated as Eq. (18).

$$\textit{Template resizing error} = \frac{\textit{size of resized template} - \textit{size of reference template}}{\textit{size of reference template}} \times 100 \quad (18)$$

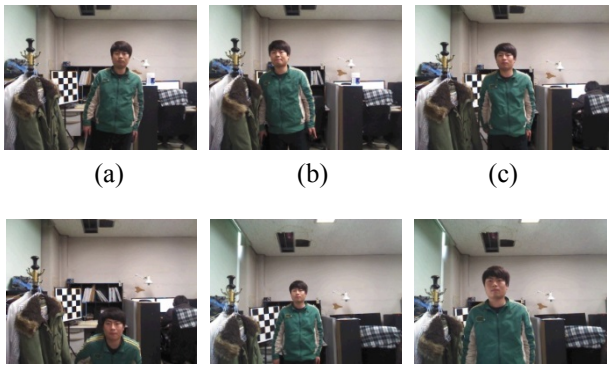
Fig. 10 shows the experimental results, where (a) shows the average values of the template resizing errors for various segmentations, and for the three test sequences. As can be seen by considering all the three test sequences, 3×3 segmentation is the best. Fig. 10 (b) shows the change of the template resizing error throughout the sequences, for 3×3 segmentation. Any frame of any sequence does not exceed 3% in error. Other experimental results showed that all the segmentations have very similar execution time. So, we decided on 3×3 segmentation as our segmentation scheme.

5.1.2 Size of the search area

The next parameter is the extension ratio α in Fig. 7, to determine the size of search area. To do this, we performed two experiments. The first was to measure the actual maximum amount of movement between two consecutive frames. In this experiment, we have 25.8%, 23.9%, and 18.2% of the size of the template as the maximum movement in for the UD, LR, and BF sequence, respectively. From this experiment, the value of α should be 0.258.

Table 1. Contents used for parameter determination

Name	RGB image resolution	Depth map resolution	Number of frames
Up-down movement (UD)	640×480	640×480	200
Left-right movement (LR)	640×480	640×480	200
Back-and-forth movement (BF)	640×480	640×480	200



(a) (b) (c)

(d) (e) (f)

Fig. 4. Representative images in each sequence: in LR: (a) 18th frame; (b) 119th frame; in UD; (c) 36th; (d) 166th, in BF; (e) 96th frame; (f) 130th frame

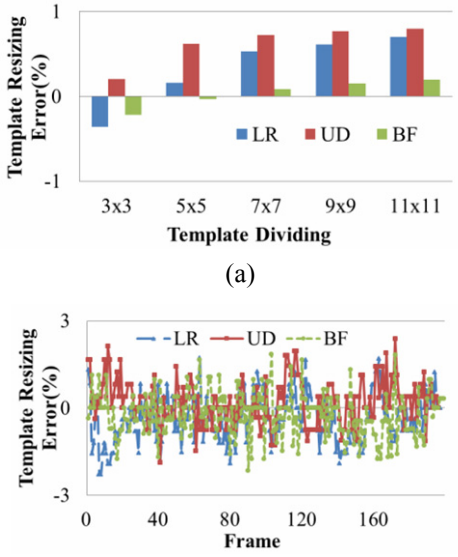


Fig. 5. Template resizing error for template segmentation: (a) average values for various segmentations, and (b) values for 3×3 segmentation

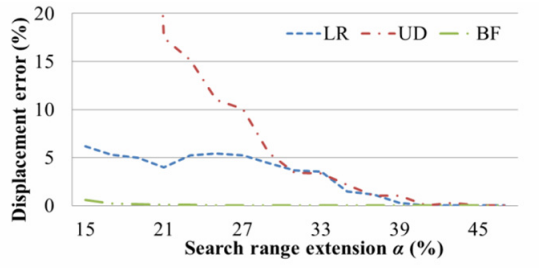


Fig. 11. Displacement errors for the three test sequences to the value of α

The second experiment was to measure the amount of displacement between the found template T_{found} within the given search area, and the best solution of template T_{best} found in the whole image. Here, both templates were found by a full search, without early termination, or sparse search. We converted it to displacement error, which is calculated by Eq. (19). *size*(T) and *position*(T) means the size and the position of T, respectively.

$$\text{Displacement error} = \frac{\left|position\left(T_{found}\right) - position\left(T_{best}\right)\right|}{size\left(T_{best}\right)} \times 100 \quad (19)$$

The result is shown in Fig. 11. The values in α<21% of sequence UD were more than 20%, because the face could not be tracked properly. From this experiment, it is clear that the result by estimating maximum movement between two consecutive frames (α =0.258) is not enough. To make the displacement error almost 0 (less than 0.1%), α≥0.41 should be maintained. Therefore, we decided on α as 0.41.

5.1.3 Threshold value for early termination

The PSAD threshold value T_{ET} for early termination was also determined by an empirical method. Fig. 12 is the result, where both displacement errors (D-Error) by Eq. (19), and execution times (Time), are shown with average values. The BF sequence shows the lowest displacement errors on almost all the threshold values, but for the execution time, the UD sequence shows the lowest. Because our aim was to track the face in real time, with appropriately low displacement error, we have chosen T_{ET} as 2, which makes the displacement error lower than 4%, with execution time lower than 40ms.

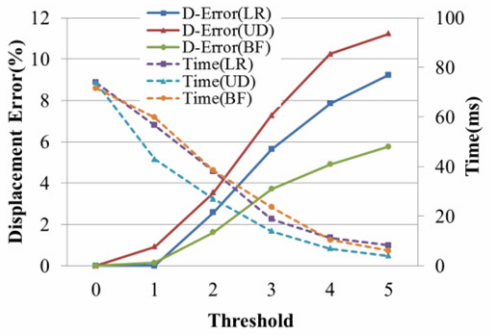


Fig. 12. Experimental results to determine T_{ET}

5.1.4 Hopping distance and refinement search for sparse search

Another scheme to reduce the execution time is to hop from the current search pixel to the next, in the spiral sequence. The hopping distance was determined empirically, as well. We have measured the displacement error and execution time by increasing the hopping distance, fixing on T_{ET}=2. The result is shown in Fig. 13 (a), where all the displacement errors and execution times are the average values. As the hopping distance increases, the displacement error increases, and the execution time decreases, as expected. Because of our aim, we took 3 as the hopping distance, which makes the displacement error less than 5%, and the execution time less than 30ms.

One more thing we decided was the refining process that examines the neighboring pixels to the early-terminated one (refer to Fig. 8). This also has two options, 2-pixel refinement and 4-pixel refinement. Because we have chosen T_{ET}=2 and hopping distance=3, we used them in this experiment. The result is shown in Fig. 13 (b). The displacement error of UD dramatically decreased in 2-pixel refinement and 4-pixel refinement, although other sequences did not change much for any refinement. Also, the increase in execution time is negligible for all the

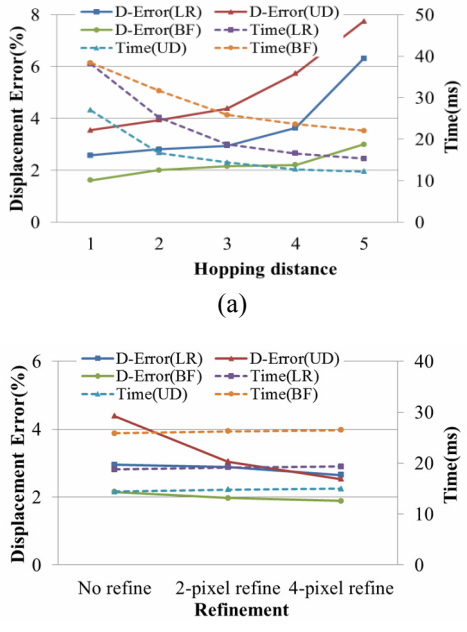
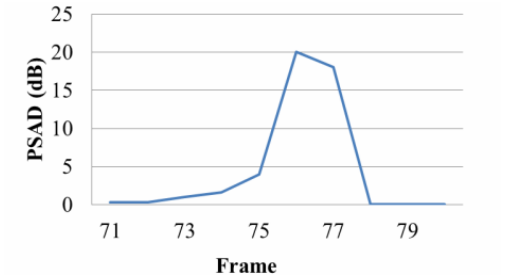
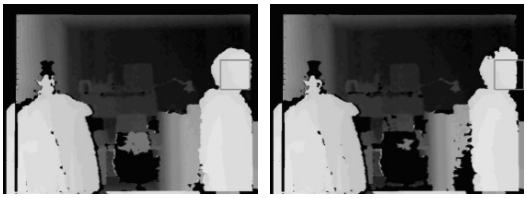


Fig. 13. Experimental results to determine the hopping distance and refinement process: (a) hopping distance; (b) refinement process.

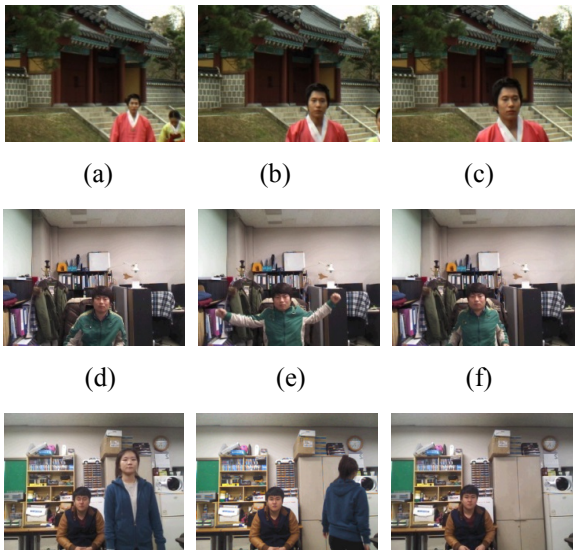


(a)



(b) (c)

Fig. 14. Experimental result to determine TFB: (a) PSAD value change; (b) 75th frame image; (c) 76th frame image.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Fig. 15. The representative images for Lovebird1: (a) 18th frame; (b) 90th frame; (c) 119th frame; WL: (d) 116th frame; (e) 303th frame; (f) 416th frame; and S&J: (g) 144th frame; (h) 182th frame; (i) 328th frame.

sequences, and all the refinement processes. So, we will apply 2-pixel refinement or 4-pixel refinement.

5.1.5 Threshold value to return to the face detection process

The final parameter to be decided is the PSAD threshold for the tracking process to return to the face detection process, by assuming that the tracking process could not track the face correctly any more. For this we prepared a few special sequences, where a human walks out of the screen, or behind an object, etc. Fig. 14 (a) shows an example of the experimental result of PSAD values when the human being tracked walks out of the screen. As you can see in the graph, the PCAD value changes dramatically from the 75th frame to the 76th frame. Images for the two frames are shown in Figs. 14 (b) and (c), respectively. Other sequences showed similar results, so, it is quite reasonable to choose T_{FB} =15[dB].

5.2 Experimental results and comparison

We have experimented using the proposed face detection scheme and the face tracking scheme with some test sequences. They are listed in Table 2, where the first three sequences were tested only for the detection scheme, because they were used to extract the parameters for the tracking scheme. Lovebird1 is a test sequence from MPEG for multi-view, and its resolution is 1,920×1,080. The last two were home-made with Kinect®. In Lovebird1, two persons walk side-by-side, from far away, to very near the camera. The WL sequence has only one person sitting on the screen, and moving various ways. In the S&J sequence, two persons appear at first, but the one near the camera walks out of the screen. Then, the other is moving this and that way, while sitting on a chair. Three representative images for each sequence are shown in Fig. 15.

5.2.1 Face detection

The experimental results for the proposed face detection scheme are summarized in Table 3, which includes the true positive rate (TP), false positive rate (FP), false negative rate (FN), and execution time per frame for each test sequence. In this table, our results are compared with the

Table 2. Applying test sequences

Sequence name	Number of frame	Resolution	Application	
			Face detection	Face tracking
Left-right movement (LR)	200	640×480	○	×
Up-down movement (UD)	200	640×480	○	×
Back-and-forth movement (BF)	200	640×480	○	×
Lovebird1	120	1,024×768	○	○
Wooyoul (WL)	450	640×480	○	○
Soojin and Jinhyuk (S&J)	330	640×480	○	○

Adaboost method in [3] (V&J). TP, FP, and FN were calculated as Eqs. (20-1), (20-2), and (20-3), respectively, where, a true positive face is a face that was detected and truly resides in the image, a false positive face is one that was detected but is not truly a face truly, and a false negative face is a face that resides in the image, but was not detected.

As can be seen in the table, V&J is a little better in FN, but ours are much better in TP. In particular, ours showed 0% of FP. Also the execution time of ours was about 1/10 that of V&J. This means that our scheme reduces the search area to about 1/10 of the whole image, with a little sacrifice in the FP.

Table 4 compares some previous methods with ours. Note that the test sequences for each method are different. Because ours uses depth information, and the others do not contain their depth information, those test sequences could not be used. Also, we failed to get the implemented results for the existing methods. Thus, it seems unclear to compare them; but we still think it is worthy, because it makes some indirect comparison possible. Also note that the '-' in a cell indicates that there is no information in the corresponding paper. As can be seen in the table, ours outperforms any existing method in TP, FP, FN, and even in the execution time.

5.2.2 Face tracking

The experimental results of the proposed face tracking scheme for the last three sequences in Table 2 are shown in Fig. 16, where displacement error rate and execution time per frame are graphed to the frames for each sequence. Also, the average values are shown in Table 5. Fig. 16 and Table 5 include two refinement schemes, 2-pixel refinement and 4-pixel refinement. In this experiment, we let the scheme track the nearest person to the camera, when more than one person resided in the screen (Lovebird1 sequence, and the front part of the S&J sequence).

Because the man in Lovebird1 walks toward the camera, his face gets larger with moving left and right repeatedly, and whenever the direction changes, he remains still for a couple of frames. It is projected exactly to the execution time, as shown in Fig. 16 (a). Also, the execution time becomes larger, as the sequence progresses, because the size of face becomes larger. That is, as the size of face increases, the execution time increases. This can be also found, by comparing the times between before and after, about the 190th frame of S&J. The woman is nearer than

$$TP = \frac{\#(\text{true positive faces})}{\#(\text{true positive faces}) + \#(\text{false positive faces}) + \#(\text{false negative faces})} \times 100 \quad (20\text{-}1)$$

$$FP = \frac{\#(\text{false positive faces})}{\#(\text{true positive faces}) + \#(\text{false positive faces}) + \#(\text{false negative faces})} \times 100 \quad (20\text{-}2)$$

$$FN = \frac{\#(\text{false negative faces})}{\#(\text{true positive faces}) + \#(\text{false positive faces}) + \#(\text{false negative faces})} \times 100 \quad (20\text{-}3)$$

Table 3. The experimental results and comparison of the face detection methods

Sequence name	V&J				Ours			
	TP(%)	FP(%)	FN(%)	Execution time [ms]	TP(%)	FP(%)	FN(%)	Execution time [ms]
LR	88.0	21.5	0.5	387.97	99.1	0	0.8	36.13
UD	94.0	5.5	0.5	988.99	98.4	0	1.6	37.65
BF	91.0	1.0	8.0	366.31	94.5	0	5.5	35.03
Lovebird1	98.3	0	1.7	885.49	98.6	0	1.4	84.08
WL	98.9	0.9	0.2	371.86	98.8	0	1.2	43.65
S&J	91.5	8.2	0.3	380.44	98.2	0	1.8	65.44
Average	93.6	6.2	1.9	463.51	97.9	0	2.1	50.33

Table 4. Comparison with existing methods for face detection

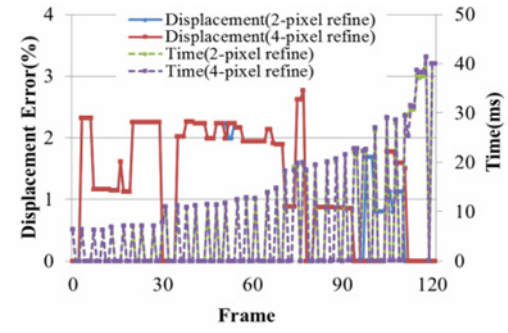
Method	Test sequences			Detection statistics			Execution time per frame [ms]
	Kind	Properties	Resolution	TP(%)	FP(%)	FN(%)	
V&J	Ours Loverbirds1	Gray	640×480 1,024×768	92.7 98.3	7.42 0	1.9 1.7	379.11 885.49
[17]	GTAV	Color	240×320	77.5	2.2	19.2	10024.00
[5]	Bao database	Color	-	90.6	5.4	4.0	-
[6]	Caltech & home-made	Color & gray	306×434	83.7	8.9	7.4	370.00
[7]	-	-	640×480 1024×768	85.5	2.8	-	129.79 286.45
Ours	Ours & Loverbirds1	Depth+ color	640×480 1,024×768	97.8 98.6	0 0	2.2 1.4	43.58 84.08

Table 5. Experimental results for the proposed face tracking scheme

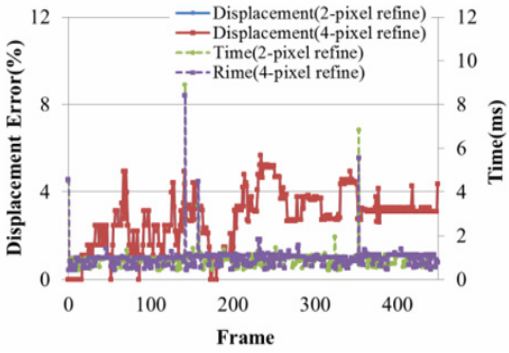
Sequence name	Displacement error (%)				Execution time per frame (ms)			
	Full search	Early term.	2-pixel refinement	4-pixel refinement	Full search	Early term.	2-pixel refinement	4-pixel refinement
Lovebirds	0	1.510	1.303	1.280	164.039	7.374	7.462	7.561
WL	0	3.188	2.797	2.658	71.145	0.960	0.978	1.003
S&J	0	3.348	3.337	3.209	74.461	3.786	3.911	3.944
Average	0	2.682	2.479	2.382	103.215	4.040	4.117	4.169

Table 6. Comparison with existing methods for face tracking

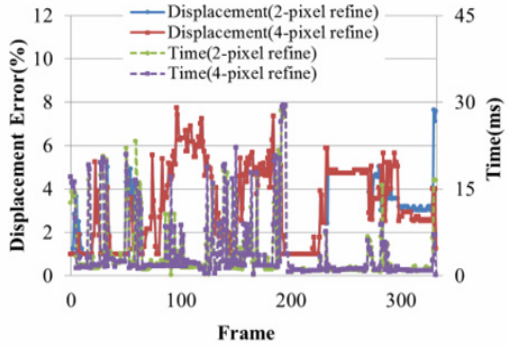
Method	Sequence			Tracking results		Execution time (fps)
	Name	Property	Resolution	Tracking rate (%)	Tracking error	
[27]	Home-made	Home-made	320×240	99.57	-	50
[36]	Home-made	Depth+RGB (Kinect, SR4000)	160×120	-	3-6cm	68
[29]	Home-made	Gray	-	-	RMSE=5.21[pixel]	9
Ours	Home-made (2-pixel refinement)	Depth+RGB(Kinect) Depth+RGB(MPEG)	640×480 1024×768	100 100	Disp. error=3.067% (0.812cm, RMSE=3.25) Disp. error=1.303%(0.234cm, RMSE=1.95)	404 132



(a)

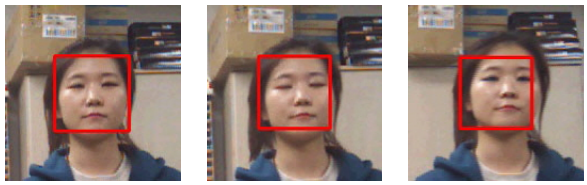


(b)



(c)

Fig. 16. Face tracking experimental results for displacement and execution time for the sequence of: (a) Lovebird 1; (b) WL; (c) S&J.



(a) (b) (c)

Fig. 17. Texture images corresponding to the depth templates, with displacement error ratios of: (a) 0%; (b) 4%; (c) 7.8%

the man. For reference, the distances from the camera to the man in WL, and the woman and the man in S&J are about 120cm, 110cm, and 160cm, respectively.

As in Table 5, the average displacement error was less than 3%, with less than 5ms of the execution time per frame. For reference, Fig. 17 shows three example texture images corresponding to the depth template images, with 0%, 4%, and 7.8% of displacement error, respectively. By considering them, it is obvious that 3% of the average displacement error is quite acceptable. Also, the proposed tracking scheme takes about 5ms in average to track the face per frame, and even for the Lovebird1 sequence, it is less than 8ms, which is more than enough speed for real time tracking.

Table 6 compares our method with some existing methods. Because they do not provide enough information for fair and clear comparison, we have used the data from the papers without modification or recalculation. Instead, we fitted our data to them as close as possible. Most tracking methods used some sequences made by their respective authors as the test sequences, and so did we. Among the three schemes in Table 5, we have chosen the 2-pixel refinement scheme, because both its error rate and execution time are in the middle. In the table, we entered depth+RGB as the property of the sequence of ours, but actually, only depth was used. For tracking rate, ours showed 100%, because ours changes the execution mode to

face detection, when it fails to track the face. It happened only once in the S&J sequence, as explained before. In the tracking error, [29] and [36] provided the displacement amount and root mean square error (RMSE), as the information for how accurate they are, respectively, while we have provided the displacement error rate in the previous table. So, we have changed our data corresponding to them, and showed it in parenthesis. From the data in the table, it is clear that our method outperforms in tracking accuracy, and execution time.

6. Conclusion

In this paper, we have proposed a combination of a face detection scheme and face tracking scheme, to track a human face. The face detection scheme basically uses the Viola & Jones method [2-4], but we have reduced the area to be searched, with skin color and depth information. The proposed tracking scheme basically uses a template matching method, but it only uses depth information. It includes a template resizing scheme, to adapt to the change of the size or depth of the face being tracked. It also incorporates an early termination scheme, by a spiral search for template matching, with a threshold, and a refinement scheme with the neighboring pixels. If it fails to track the face by deciding with a threshold, it automatically returns to the face detection scheme, to find a new face to track.

Experimental results for the face detection scheme showed 97.6% of the true positive detection rate on average, with 0% and 2.1% of false positive rate and false negative rate, respectively. The execution time was about 44[ms] per frame, with 640×480 resolution. From the comparison with the existing methods, it was clear that ours is better, in both detection accuracy and execution time.

The experimental results for the proposed face tracking scheme showed that the displacement error rate is about 2.5%, with almost 100% tracking rate. Also, the tracking time per frame with 640×480 resolution was as low as about 2.5 [ms]. These results much outperform the previous face tracking schemes. Also, we have shown some trade-offs between tracking accuracy and execution time for the size of the search area, PSAD threshold value for early termination, hopping distance, and refinement scheme.

Therefore, we can conclude that the proposed face detection scheme and face tracking scheme, or their combination, can be used in an application that needs fast and accurate face detection and/or tracking. Because our tracking scheme can also provide higher speed with sacrificing a little tracking accuracy, by increasing the early termination threshold value and hopping distance, or decreasing the search area and taking a simpler refinement scheme, its applicable area can be found much more broadly.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (MEST). (2010-0026245).

References

[1] M.-H. Yang, D. J. Kriegman and N. Ahuja, “Detecting Faces in Images; A Survey,” IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol. 24, No. 1, pp. 34-58, Jan. 2002.

[2] P. Viola and M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features,” Conf. on Computer Vision and Pattern Recognition, pp. 1-9, 2001.

[3] P. Viola and M. Jones, “Robust Real-time Object Detection,” Intl. Workshop on Statistical and computational Theories of Vision-Modeling, Learning, Computing, and Sampling, pp. 1-25, 2001.

[4] P. Viola and M. Jones, “Robust Real-Time Face Detection,” J. of Compter Vision, 57(2), pp. 137-154, 2004.

[5] C. E. Erdem, S. Ulukaya, A. Karaali and A. T. Erdem, “Combining Haar Feature and Skin Color Based Classifiers for Face Detection,” Conf. Acoustics, Speech and Signal Processing, pp. 1497-1500, 2011,

[6] S. A. Inalou and S. Kasaei, “AdaBoost-based Face Detection in Color Images with Low False Alarm,” Conf. Computer Modeling and Simulation, pp. 107-111, 2010.

[7] Y. Tu, F. Yi, G. Chen, S. Jiang and Z. Huang, “Fast Rotation Invariant Face Detection in Color Image Using Multi-Classifier Combination Method,” Conf. EDT, pp. 211-218, 2010.

[8] J. Wu, S. C. Brubaker, M. D. Mullin and J. M. Rehg, “Fast Asymmetric Learning for Cascade Face Detection,” IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 30, No. 3, pp. 369-382, March 2008.

[9] Y.-W. Wu and X.-Y. Ai, “Face detection in color images using AdaBoost algorithm based on skin color information,” Workshop on Knowledge Discovery and Data Mining, pp. 339-342, 2008.

[10] Y. Tie and L. Guan, “Automatic face detection in video sequence using local normalization and optimal adaptive correlation techniques,” J. Pattern Recognition 42, pp. 1859-1868, 2009.

[11] P. Shih and C. Liu, “Face detection using discriminating feature analysis and support vector machine,” J. Pattern Recognition 39, pp. 260-276, 2006.

[12] A. Colombo, C. Cusano and R. Schettini, “3D face detection using curvature analysis”, J. Pattern Recognition 39, pp. 444-455, 2006.

[13] C. A. Waring and X. Liu, “Face Detection Using

Spectral Histograms and SVMs," IEEE trans. Syst., Man, And Cybernetics, Vol. 35, No. 3, pp. 467-476, Jun. 2005.
[14] W.-K. Tsao, A. J. T. Lee, Y.-H. Liu, T.-W. Chang and H.-H. Lin, "A Data mining Approach to Face Detection," J. Pattern Recognition 43, pp. 1039-1049, 2010.
[15] A. Sagheer and S. Aly, "An Effective Face Detection Algorithm based on Skin Color Information," Conf. Signal Image Technology and Internet Based Systems, pp. 90-96, 2012.
[16] H. Fan, D. Zhou, R. Nie and D. Zhao, "Target Face Detection using Pulse Coupled Neural Network and Skin Color Model," Conf. Computer Science & Service System, pp. 2185-2188, 2012.
[17] S. Kherchaoui and A. Houacine, "Face Detection Based on a Model of the Skin Color with Constraints and Template Matching," Conf. Machine and Web Intelligence, pp. 469-472, 2010.
[18] H.-Y. Chen, C.-L Huang and C.-M. Fu, "Hybrid-boost Learning for Multi-pose Face Detection and Facial Expression Recognition," J. Pattern Recognition 41, pp. 1173-1185, 2008.
[19] C. Huang, H. Ai, Y. Li and S. Lao, "High-Performance Rotation Invariant Multiview Face Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 29, No. 4, pp. 671-686, April 2007.
[20] T. Maurer and C. Malsburg, "Tracking and Learning Graphs and Pose on Image Sequences of Faces," International Conf. Automatic Face and Gesture Recognition, pp. 176-181, 1996.
[21] S. McKenna, S. Gong, R. Wurtz, J. Tanner and D. Banin, "Tracking Facial Feature Points with Gabor Wavelets and Shape Models," International Conf. on Audio and Video-based Biometric Person Authentication, pp. 35-42, 1997.
[22] Q. Wang, W. Zhang, X. Tang, and H.-Y. Shum. "Real-time Bayesian 3-D Pose Tracking," IEEE Trans. Circuits Syst. Video Techn., Vol. 16, No. 12, pp. 1533-1541, Dec. 2006.
[23] W. Zhang, Q. Wang, and X. Tang. "Real Time Feature Based 3-D Deformable Face Tracking," ECCV, pp. 720-732, 2008.
[24] T. Cootes and G. Edwards, "Active Appearance Models," IEEE Trans. Pattern Anal. Mach. Intell., Vol. 23, No. 6, pp. 681-685, June 2001.
[25] T. Cootes, G. Wheeler, K. Walker and C. Taylor, "View-based Active Appearance Models," Image Vision Comput. Vol. 20, No. 9, pp. 657-664, 2002.
[26] J. -W. Sung and D. Kim, "Large Motion Object Tracking using Active Contour Combined Active Appearance Model," International Conference on Computer Vision Systems, p. 31, 2006.
[27] M. Zhou, L. Liang, J. Sun and Y. Wang, "AAM based Face Tracking with Temporal Matching and Face Segmentation," IEEE Conf. on Computer Vision and Pattern Recognition, pp. 701-708, 2010.
[28] P. Wang and Q. Ji, "Robust Face Tracking via Collaboration of Generic and Specific Models," IEEE Trans. Image Processing, Vol. 17, No. 7, pp. 1189-1199, July 2008.
[29] Y. Lui, J. Beveridge and L. Whitley, "Adaptive Appearance Model and Condensation Algorithm for Robust Face Tracking," IEEE Tran. Syst., Man, and Cybernetics, Vol. 40, No. 3, pp. 437-448, May 2010.
[30] Y. Raja, S. McKenna, and S. Gong, "Colour Model Selection and Adaptation in Dynamic Scenes," European Conference on Computer Vision, pp. 460-474, 1998.
[31] G. Jang and I. Kweon, "Robust Real-time Face Tracking using Adaptive Color Model," International Symposium on Mechatronics and Intelligent Mechanical System for 21 Century, 2000.
[32] H. Stern and B. Efros, "Adaptive Color Space Switching for Tracking under Varying Illumination," Image Vision Computation, Vol. 23, No. 3, pp. 353-364, 2005.
[33] H. Lee and D. Kim, "Robust Face Tracking by Integration of Two Separate Trackers: Skin Color and Facial Shape," J. pattern recognition, Vol. 40, pp. 3225-3235, March 2007.
[34] P. Vadakkepat, P. Lim, L. Silva, L. Jing and L. Ling, "Multimodal Approach to Human-Face Detection and Tracking" IEEE Trans. Industrial Electronics, Vol. 55, No. 3, pp. 1385-1393, March 2008.
[35] R. Qian, M. Sezan, K. Matthews, "A Robust Real-time Face Tracking Algorithm," International Conference on Image Processing, pp. 131-135, 1998.
[36] X. Suau, J. Ruiz-Hidalgo and J. Casas, "Real-Time Head and Hand Tracking Based on 2.5D Data", IEEE Trans. Multimedia, Vol. 14, No. 3, pp. 575-585, June 2012.
[37] D. Chai, et al., "Locating Facial Region of a Head-and -Shoulders Color Image," Int'l Conf. Automatic Face and Gesture Recognition, pp.124-129, April 1998.
[38] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 3rd Ed., Pearson Ed. Inc., Upper Saddle River, NJ, 2008.

Dong-Wook Kim He has received his M.S degree in 1985 from Dept. of Electronic Engineering of Hangyang University in Seoul, Korea and his Ph.D degree in 1991 from Dept. of Electrical Engineering of Georgia institute of Technology in GA, U.S.A. He is a Professor of Dept. of Electronic Materials Engineering at Kwnagwoon University in Seoul, Korea.

Woo-Youl Kim He has received his B.S degree in 2012 from Dept. of Information & Communication Engineering of Anyang University in Anyang, Korea. He is currently pursuing M.S degree in Dept. of Electronic Materials Engineering of Kwangwoon University in Seoul, Korea. His research interests include digital image processing, digital holography, image compression.

Jisang Yoo He received the B.S. and M.S. degrees from Seoul national university, Seoul, Korea in 1985 and 1987, all in electronics engineering, and Ph.D. degree from Purdue University, West Lafayette, IN, USA, in electrical engineering in 1993, respectively. From September 1993 to august 1994, he worked as a senior research engineer in industrial electronics R&D center at Hyundai Electronics Industries Co., Ltd, Inchon, Korea, in the area of image compression and HDTV. He is currently a professor with the department of electronics engineering, Kwangwoon University, Seoul, Korea. His research interests are in signal and image processing, nonlinear digital filtering, and computer vision. He is now leading 3DTV broadcast trial project in Korea.

Young-Ho Seo He has received his M.S and Ph.D degree in 2000 and 2004 from Dept. of Electronic Materials Engineering of Kwangwoon University in Seoul, Korea. He was a researcher at Korea Electrotechnology Research Institute (KERI) in 2003 to 2004. He was a research professor of Dept. of Electronic and Information Engineering at Yuhan College in Buchon, Korea. He was an assistant professor of Dept. of Information and Communication Engineering at Hansung University in Seoul, Korea. He is now an associated professor of College of Liberal Arts at Kwangwoon University in Seoul, Korea and a director of research institute in TYJ Inc. His research interests include realistic media, digital holography, SoC design and contents security.