Abstract—Segmenting human faces automatically is very important for face recognition and verification, security system, and computer vision. In this paper, we present an accurate segmentation system for cutting human faces out from video sequences in real-time. First, a learning based face detector is developed to rapidly find human faces. To speed up the detection process, a face rejection cascade is constructed to remove most of negative samples while retaining all the face samples. Then, we develop a coarse-to-fine segmentation approach to extract the faces based on a min-cut optimization. Finally, a new matting algorithm is proposed to estimate the alpha-matte based on an adaptive trimap generation method. Experimental results demonstrate the effectiveness and robustness of our proposed method that can compete with the well-known interactive methods in real-time.

Index Terms—Segmentation, face detection, AdaBoost, graph cut, matting.

EDICS: 4-SEGM Audio/Image/Video Segmentation for Interactive Services

I. INTRODUCTION

OBJECT segmentation plays an important role in digital video processing, pattern recognition, and computer vision. The task of segmenting/tracking a video object emerges in many applications, such as video monitoring and surveillance, video conferencing and videophony, video summarization and indexing, and advanced editing and digital entertainment [1]. For example, in video surveillance, the segmentation result can be used to identify a suspicious subject, and helps to detect his/her action and alert the security center to the possible danger in time.

Unlike the traditional segmentation that aims to extract some uniform and homogeneous regions with respect to texture or color properties, recent segmentation can be defined as a process which typically partitions an image into meaningful objects according to some specified semantics. This approach usually consists of two stages, i.e., desired object detection and object extraction. The first stage is related to the pattern recognition techniques that relies on prior information, whilst the second is concerned with classification technique to separate the closed object boundary.

However, it is usually difficult to locate the desired content based on an unified detection method due to the various object features, such as color, intensity, shape, and contour. In order to achieve better segmentation result, many interactive methods have been proposed [2]– [7], which require the user to define the desired object to be extracted in advance. For example, a system for cutting a moving object out from a video clip and pasting it onto another video or a background image was presented based on 3D graph cut in [6]. To achieve this goal, the user is required to select a few key frames in the video sequence at ten-frame intervals and provide their foreground/background segmentation using the snapping tool [5]. In addition, an interactive system for extracting foreground objects from a video was proposed in [7], which allows users to easily indicate the foreground objects across space and time. Since the complicated step of object detection is avoided at the cost of interactive effort on the part of the user, these methods usually can provide users with much better segmentation performance than automatic ways.

In order to satisfy the future content-based multimedia services, the segmentation of meaningful objects in unsupervised manner is urgently required in the real-world scenes. Fortunately, some specific objects of interest can be detected by designing appropriate detectors based on physical model or training scheme. In this paper, we will concentrate on the problem of segmenting human face automatically, which may be applied to many real scenarios, such as video surveillance, videophone, video-conferencing, and web chatting, etc. An efficient and robust face segmentation algorithm is presented. This method consists of three stages. Firstly, we employ an efficient face detector based on AdaBoost learning algorithm. Then, we perform the face segmentation based on graph cut optimization method. Finally, an effective matting algorithm has been developed, which can be used to refine the face boundary.

This paper is organized as follows. Related work will be reviewed in Section II. The efficient face detection algorithm is introduced in Section III. The proposed segmentation methods based on graph cut and matting will be presented in Section IV and V, respectively. Experimental results are provided in Section VI to support the efficiency of our proposed algorithm. Finally, in Section VII, conclusions are drawn and further research proposed.
II. RELATED WORK

The task of segmenting human faces automatically is generally considered a crucial step for face recognition, video chatting, or human-to-machine communication. For example, face segmentation is one of the most important tasks in model-based video coding for video telephony and video conferencing applications. In addition, the segmented human face can not only be used to the face recognition, but also the digital entertainment, such as face tooning. The obtained face location can also provide important cue for the following human tracking and surveillance.

In the literature, many face segmentation algorithms were presented based on the skin color information [8]-[15]. These methods aim to exploit skin color distribution to locate and extract the face region. For example, a universal skin-color map is introduced and used on the chrominance component to detect pixels with skin-color appearance in [8]. In order to overcome the limitations of color segmentation, five operating stages are employed to refine the output result, such as density and luminance regularization, geometric correction. If the standard deviation of the spatial distribution of the luminance values is below a value of two, then the detected region will be considered as background and eliminated. In addition, a hand and face segmentation method using color and motion cues for the content-based representation of sign language videos was proposed in [14]. This method consists of three stages: skin-color segmentation, change detection, and generation of a face and hand segmentation mask. The skin-color distribution is modeled as a bivariate normal distribution and the image pixels are classified based on their Mahalanobis distance.

A statistical model-based video segmentation for head-and-shoulder type video is addressed in [16]. In that work, the head is modeled with a "blob", which is segmented based on the assumption that a background scene contains no foreground in order to satisfy the creation of a background model. Another head detection based on intensity edges and foreground residue map can be found in [17]. Recently, a face segmentation algorithm is proposed based on our facial saliency map for head-and-shoulder type video application [18] [19]. This method first constructs a saliency map from the input video image based on a facial attention model. Then, a geometric model and an eye-map are employed to localize face region according to the saliency map. In [20], the background cut algorithm was proposed to extract a foreground for a video sequence with a moving foreground object and stationary background. Moreover, many learning based segmentation methods have been proposed in recent years [21]-[23]. For example, an unsupervised method is proposed to segment objects detected in images by integrating keypoint based template object models [21]. The learned object model integrates image gradients, the location and scale of the object, the presence of object parts.

In this paper, we propose an automatic face segmentation method, which can extract face regions effectively and accurately from input images/video sequences. An efficient face detector is first employed to detect human faces in the input image. A training method based on AdaBoost learning algorithm is used to build the face classifier by using non-normalized Haar transform (NHT) coefficients. Then, a coarse-to-fine segmentation method is developed based on graph cut and matting approaches.

Our approach has the following advantages. First, an efficient face detector based on AdaBoost algorithm is developed, which can achieve more accurate detection rate than the skin-color based methods. Then, we developed an effective coarse-to-fine segmentation approach using min-cut optimization. Finally, a new approach for alpha-matte computation is proposed based on energy minimization and an adaptive trimap generation method.

III. FACE DETECTION

As a highly non-rigid object, human face holds a high degree of variability in size, shape, color, and texture [24]. The boundary between the face and non-face patterns is also highly non-linear because of variations in facial appearance, lighting, head pose, and expression [27]. In this section, we aim to identify and locate frontal-view human faces from the input image regardless of their positions, scales, and illumination.

A. The Proposed Face Detection Method

The proposed face detector scans the image at many scales, looking for face locations within the scaled windows. It consists of three parts, i.e., skin color filtering, rejector cascade, and cascades of boosted face classifier. The filter is used to clean up the non-skin regions in the color image during face detection. The rejector is designed to remove most of the non-face candidates while allowing 100% accuracy for face detection. The promising face-like locations will be examined in the final boosted face classifier.

B. Skin Color Filtering

It is known that there are many color space models relevant to different applications, such as RGB for display purpose, HSV for computer graphics and image analysis, and YCbCr. Since the YCbCr color space is usually employed for the video storage and coding, and can provide effective analysis for human skin color [8] [18], we will use this color space for the input video images.

To investigate the skin-color distribution, we manually segmented a large number of training images into face patches. Different background, lighting conditions, face types can be found for these image sources. It is shown that skin-color can be detected by the presence of a certain range of chrominance values with narrow and consistent distribution in the YCbCr color space. The empirical range for the chrominance values employed are typically $C_r_{skin} = [133, 173]$ and $C_b_{skin} = [77, 127]$ [8]. Since face region usually exhibits the similar skin-color feature regardless of different skin types, we can quickly eliminate or skip most of non-skin color regions to save considerable computation time.

Assume $S(m,n)$ represents the filtered binary result at position $(m,n)$ in the current image, i.e., 1 for the skin-color chrominance values or 0 otherwise. We then compute the integral image of the binary map. Based on the method in [25],
we can compute the sum of mask values within any rectangle. If no or less skin-color pixels appear in this rectangle, we can declare that no human face is found in the current window and can be skipped. Here, the threshold is set to a small value in order to detect almost all human faces while rejecting many of the non-skin color windows.

C. Rejector Cascade

In this section, a rejection cascade is designed to reject a large number of non-face samples while detecting almost 100% of the faces. The cascade can significantly reduce the computation time before more complex face classifiers are called upon to achieve low false positive rates. In [33], the relationships between the mean and variance of image segments are used to form a cascade of rejectors. The image is first segmented into several segments of temporally correlated pixels. Every segment is then approximated with a small number of representative pixels and used to perform the training process.

In our work, we demonstrate that even simple features can be used for constructing an efficient rejector cascade. Since these features are also used for the following boosting face classifier, we can see that no additional computation is needed for feature generation in some sense.

Algorithm 1 Training for Rejector Using Variance features

1. Input training examples \((x_1, y_1), \ldots, (x_n, y_n)\) where \(y_i = 0, 1\) for non-face and face examples respectively.
2. Initialize rejection label \(l_i = 0\), for \(y_i = 0\).
3. For \(t = 1, \ldots, T\):
   4. Find the minimal and maximal values of \(\sigma_k\) for each region \(k\) from the face training examples, which is denoted by \(\sigma_k^{-1}\) and \(\sigma_k^1\), respectively.
   5. Compute the rejection number \(r_k\) for non-face training set with a parity \(p\) adjusting the inequality direction, i.e.,
      \[
      r_k^p = \sum_{y_i = 0, l_i = 0} \text{sign}\left[p \sigma_{i,k} - p \sigma_k^p\right].
      \]
   6. Choose the region with the highest rejection number.
   7. Set label \(l_i = 1\) for all rejected samples \(\{i\}\).
   8. Repeat from step 3.

The first feature set is region variance, which can be obtained from two integral images, i.e., integral image and integral image of the image squared. It is known that these integral images are used to perform lighting correction during image scanning process. For \(24 \times 24\) pixel images, there are 76176 variance features to be considered. Assuming \(\sigma_i\) denotes the variance of the \(i\)th region, our training process can be described in Algorithm (1).

The second feature set is the difference between two low- (LL) non-normalized Haar transform (NHT) coefficients. Since these coefficients will be used to compute the high frequency NHT coefficients as the input features for the Adaboost training stage, we can see that there is also no additional computation load on this feature set generation. Based on the rescaling advantage of the Haar features, we only use the LL coefficients with the block size of \(4 \times 4\) as the training set, which includes 97020 features for a \(24 \times 24\) pixel image. The training method is similar to the first one with small modifications from step 4 to 6:

Algorithm 2 Training for Rejector Using Block Differences

4. Find the minimal and maximal values of \(D_{(k,j)} = LL_h - LL_j\) for two arbitrary coefficients from the face training examples, which is denoted by \(D_k^{-1}\) and \(D_k^1\), respectively.
5. Compute the rejection number \(r_k\) for non-face training set, i.e.,
   \[
   r_k^p = \sum_{y_i = 0, l_i = 0} \text{sign}\left[p D_{i,(k,j)} - p D_k^p\right],
   \]
6. Choose the difference feature with the highest rejection number.

We obtained 40 and 50 rejectors for the region variance set and LL NHT coefficients set, respectively, which can yield the rejection rate of 98.269% and detection rate of 100% on a testing dataset. Fig. 1 shows the rejection rate of the cascade on a training set of 500000 non-face images and 12536 face images. The upper curve denotes the combined result of the two feature sets. Compared with the LL NHT coefficients, region variance exhibits relatively higher rejection rate. It is observed that the first variance feature rejector can reject about 47.6% of the non-face images while yielding 100% detection rate for training face set.

D. Learning Face Detection

For those windows that contain many skin-color pixels and are accepted by the rejector cascade, they should be further evaluated by learning based face detector. In this section, we present our algorithm to construct a strong classifier using AdaBoost algorithm.

1) Features: The simple Haar-like features have been successfully applied to face detection by Viola and Jones based on the proposed fast calculation method [25]. In order to improve performance, more rotated Haar-like features and scalar Haar features were extended in [26] and [27] for dealing with in-plane rotations and multiview face detection, respectively. Recently, even simpler features, i.e., the relationship between two pixels’ intensities, were used to perform the sex identification [32].

Since the Haar-like features can be computed very rapidly using the integral image, most of these methods construct
the weak classifier by selecting one feature from the given feature set. In order to improve the performance of the weak classifier, joint Haar-like feature was introduced based on co-occurrence of these features in [28]. Four kinds of features shown in Fig. 2 are constructed based on different NHT coefficients. In some sense, these coefficients are more like the 'toy bricks', which can be built according to a certain style composition. Each feature consists of one or more 'bricks', which are combined by means of addition, subtraction, and absolute value operations. Examples can be found in the right-bottom column of Fig. 2. The first two features are obtained by computing the differences between two LH and HL coefficients, respectively. The sixth center-surround feature can be obtained by the addition and substraction of four HH bricks.

For a window size of $24 \times 24$, the number of overcomplete features is quite large, such as 2646 for feature A, 69768 for feature B, 830574 for feature C, and 1045386 for feature D.

2) Learning Face Classifiers: There are many boosting approaches [30] to machine learning for face classifiers, such as AdaBoost [25], [26], [31], FloatBoost [27], Kullback-Leibler Boosting [29]. In our work, we use AdaBoost algorithm [30] to combine the weak learners into a strong classifier, which was used in [25].

It is known that AdaBoost approach can be interpreted as a greedy feature selection process, which selects a small set of classifiers with lowest errors and also their associated weights. The detailed step of this algorithm is described in Algorithm (3). The final classifier is considered strong because it is a weighted combination of many weak classifiers. Although each weak classifier cannot provide good classification for the training samples, the appropriate combination of its weight with others can improve the performance of the final classification significantly.

3) Cascade of Classifiers: A cascade of classifiers is employed to reject as many non-face samples as possible at the earliest stage, which can reduce the detection time efficiently. Fig. 3 shows the constructed detection cascade. The first part is the skin-color filtering for color input images and skipped for gray images. The proposed rejection cascade is used as the second part, which rejects almost 98.269% of the non-face samples while retaining the detection rate of 100%. The third part is constructed by using the AdaBoosting face classifiers. Each classifier is adjusted to have a very high detection rate (e.g. 99.9%), but a moderate false positive rate (50%) after the AdaBoost learning. If 25 of the above classifiers are bounded together, the false alarm rate and detection rate would be $2.98 \times 10^{-8}$ and 0.9753, respectively.

It is noticed that with the decrease of the false positive rate, more weak classifiers would be required in a stage especially for those features with less combinations. The main reason is that the negative samples that are accepted after several stages will have stronger 'resistance' to the used features than the rejected samples. In order to distinguish them from the positive samples, more constraints (i.e., more weak classifiers) should be added to the strong classifier. Fortunately, in our work, various features can be constructed for the learning stage rather than a small set of features in [25], [26]. As mentioned early, the basic NHT coefficients is like the "toy bricks" that can be used to construct a large number of feature modes in terms of the combinations given in Fig. 2. Since the number of the features is very large, the training time will be unacceptable if all features are included. In our work, we employ different set of features with small number of overlaps that can obtain the appropriate running time and good classification results.

E. Training

1) Test Data: The test data consists of 12536 hand-labeled frontal faces, which are collected from many face databases such as Cortadas, CVL FaceDB, BioID, ORL, IIS, PICS, JAFFE, IMM, and the Web. Our face samples cover varying lighting, different quality, age, gender, facial expressions (open/closed eyes, smiling/non-smiling) and facial details (glasses/no-glasses). All face data are cropped and scaled to
24 × 24 pixels. We collected 9000 non-face images including the natural images and texture images, which are cropped to form a total of 944,338,068 non-face training images of size 24 × 24 pixels. Some face and non-face examples are shown in Fig. 4.

2) Results: We use 12 classifiers in the cascade structure. The negative samples of each layer are obtained from the false outputs of the last classifiers. Each layer achieves the detection rate of 99.8% with about the false rate of 30–50%. The final cascade classifier has about 99.10% detection rate and 66 false samples in the non-face training data.

As shown in Fig. 3, we use three parts to detect the faces in the real-time video. The detection window scales the video image with the zooming factor of 1.5- and 2- pixel steps. The detector runs at 13ms per frame of QVGA size (320 x 240 pixels) on a CPU of 1.85GHz. Some results for a real-time video are shown in Fig. 5.

IV. GRAPH CUT BASED FACE SEGMENTATION

After the face detection, we can access the face locations in the input image rapidly. In this section, we use the min-cut optimization method [2] to perform our face segmentation algorithm in the obtained face regions.

Given the subsets of marked object and background pixels, Boykov and Jolly [2] used graph cut to find the globally optimal segmentation based on minimum cut algorithm, which also acts as the foundation work in [4], [5], and [7]. This algorithm introduces a soft constraint incorporating both the boundary and region information for hard segmentation. Rother et al. [4] extended this work to deal with the interactive segmentation by using iterative estimation and incomplete labelling.

Generally, when the foreground or background is clearly defined by the user inputs, such as [4], [5], and [7], the segmentation process can be carried out based on graph cut optimization. Unfortunately, in the unsupervised manner, we only have the coarse face locations that are obtained from the boosting face detector. The corresponding sub-window may cover the complete face contour or only some parts of face regions. It means that the pixel outside the window may belong to the background, while the pixel inside the window is likely to be part of the face. We cannot determine with confidence which pixel should be marked as the background or foreground. In this section, we will concentrate on how to segment the face region under the over-incomplete labelling condition.

A graph \( G = (V, E) \) is defined by a set of nodes \( V \) (e.g., pixels or regions) and a set of directed edges \( E \) that connect these nodes [2]. There are also two special nodes, which are called the source terminal \( s \) and the sink terminal \( t \). In this graph, all the nodes are connected by a certain edge with different weights, i.e., the bidirectional \( n \)-links between two neighboring nodes and the \( t \)-links between the nodes and the terminals. A cut \( C \) aims to separate the nodes into two subsets, which can be labeled either as source terminal or sink terminal. The min-cut is to search a cut that has the globally minimal cost (i.e., the sum of the weights of the edges) [34].

Min-cut optimization aims to assign each node in a graph \( G \) to be either source (foreground \( F \)) or sink terminal (background \( B \)). In fact, it is very important to assign appropriate edge weights to their links for many graph cut based applications in computer vision. If each pixel denotes a node, the segmentation of an image \( Z = \{z_i\} \) can be expressed by solving an energy function based on two cost function, i.e., data cost \( E_1 \) for assigning each node \( i \) to the foreground or Background label, and the smoothness cost \( E_2 \) for measuring the similarity between two nodes:

\[
E(Z) = \sum_{i \in V} E_1(z_i) + \lambda_1 \sum_{(i,j) \in E} E_2(z_i, z_j),
\]

Data cost \( E_1 \): This term is used to set the penalties for assigning each pixel to foreground or background. Generally, in the interactive method, two terminals \( F \) and \( B \) are defined by the user input. Thus, some hard constraints can be imposed to guarantee the consistent label with the user’s paints by the min-cut. It means that infinite cost might be employed when the assigned label violates the user’s paint strokes. However, there are no predefined terminals in automatic manner.

In our work, Gaussian mixture models (GMM) are used to model the color distributions of the foreground or background based on the initial face locations. For a given pixel \( i \) with the color \( z_i \) and label \( \alpha(i) \), its distances to the foreground \( F \) and \( B \) are defined as follows:

\[
E_1(\alpha(i) \in F) = \frac{\log(d^F_i)}{\log(d^F_i) + \log(d^B_i)}
\]

\[
E_1(\alpha(i) \in B) = \frac{\log(d^B_i)}{\log(d^F_i) + \log(d^B_i)}
\]

with

\[
d^F_i = \sum_{k=1}^{K} w^F_k \left( \frac{1}{1 + 2 \det \Sigma_k} \right)^{\frac{k}{2}} \exp \left( -\frac{1}{2} (z_i - \mu^F_k)^T \Sigma_k^{-1} (z_i - \mu^F_k) \right) \tag{4}
\]

\[
d^B_i = \sum_{k=1}^{K} w^B_k \left( \frac{1}{1 + 2 \det \Sigma_k} \right)^{\frac{k}{2}} \exp \left( -\frac{1}{2} (z_i - \mu^B_k)^T \Sigma_k^{-1} (z_i - \mu^B_k) \right), \tag{5}
\]
where $w_k$ denotes the weights corresponding to the percentage of the spatial samples for the $k$th component of the GMM, and $\mu_k$ and $\Sigma_k$ represent the mean color and the covariance matrices, respectively. $K$ denotes the number of the components of the GMM in the foreground and the background. An example of the weights computation is illustrated in Fig. 6(a), where the yellow and blue colors denote the foreground and background, respectively. The spatial samples for the pixel (labeled as gray color) can be calculated according to the number of clustered pixels within the red circle region.

**Smoothness cost $E_2$:** This term is used to set a penalty for a discontinuity between two nodes (e.g., two pixels). It becomes larger when the smaller change is found between pixels $i$ and $j$, which means smaller probability of an edge appearing between the adjacent pixels. The general format is defined in terms of the local intensity gradient. Similar to the existing work [2], [4]–[6], we also adopt this exponential function based on the gradient but without label constraints, i.e.,

$$E_2(z_i, z_j) = \exp \left(-\beta \|z_i - z_j\|^2\right)$$  \hspace{1cm} (6)

where $\beta$ is a robust parameter that weighs the color contrast [6].

**Coarse-to-fine segmentation** For the previous work in [4], [5], [6], and [7], the color distributions can be obtained accurately by the user’s paints. For example, to specify an object in [5], a user should mark a few lines on the image to indicate they are foreground or background. However, in our work, we must build the color distributions from the incompletely marked face regions. The proposed method works at two levels, i.e., coarse and fine scales.

The initial segmentation is performed at the coarse level. As shown in Fig. 6, four regions, i.e., (A+B+C), (A+D+F), (C+E+H), and (F+G+H), are selected for estimating the color distributions of the background. We take their means and variances as the initial clusters for the background. The data term is estimated based on the spatial samples within a defined window of $100 \times 100$ pixels that is centered with the current node. The min-cut is also used for the final optimization.

**V. MATTING BASED ON ENERGY MINIMIZATION**

Matting is an important operation to separate a foreground element of arbitrary shape from an image by estimating the opacity of the foreground element at every pixel [35]. Since the matting equation shown in (7) has too many unknowns, matting is inherently under-constrained. Many algorithms were proposed to solve the inverse problem [3], [4], [35]–[39]. In this section, we present a new matting approach that can be used to refine the face boundaries obtained from Section IV.

$$I = \alpha F + (1-\alpha)B.$$  \hspace{1cm} (7)

From (7), we can see that when matte $\alpha$ changes from 1 (or 0) to 0 (or 1), there will be a gradual color transition from foreground (or background) to background (or foreground). Because $\alpha$ value is restricted to the range of [0, 1], the original data $I$ in the matting regions must be a certain value between the foreground and the background. A typical matting example is illustrated in Fig. 7(a), where there are gradually changed progress between the F and B regions. In addition, there are still many cases where the distinct boundary can be observed between the background and foreground, which means there is no blending effect occurring between the two regions. When the foreground and the background color distributions are well estimated in the unknown region shown
in Fig. 7(b), the corresponding α values (1 or 0) can be solved according to the similar method in the first case. However, this method fails in some cases, where one of distributions of two regions cannot be estimated from the neighboring regions. For example, as shown in Fig. 7(c), there are no mixed regions between the foreground and the background although a transition region can be observed in the foreground area. If we take the regions between the two dashed lines as the unknown region to perform the α computation, the optimized matte will smooth the foreground region in the unknown area which results in the distinct artifacts. Of course, the ill-condition may be avoided by using the manually defined trimap, i.e., three regions of ΩF ("definitely foreground region (α = 1)"), ΩB ("definitely background region (α = 0)"), and ΩU ("unknown region (α ∈ [0, 1])"), but it is difficult to deal with in unsupervised manner. Unfortunately, the last case is usually observed for the face boundary.

In order to avoid the unmanageable case and reduce the possible estimation errors, we propose an adaptive method to construct the trimap. Generally, the standard approach of generating a trimap automatically can be achieved by using the morphological erosion and dilation to yield the unknown regions based on the initial object boundary [4], [37]–[38], [40]. A consistent structure size is employed, such as 5 pixels in [38], [40], and 6 pixels in [4]. In our work, the trimap size depends on the blur degree of the centered pixel with respect to its neighborhood. It means that larger size will be required if this region is well mixed between the foreground and the background, otherwise a small unknown region will be considered. Since our matting begins with a closed face contour, we perform the trimap calculation on each pixel on the face contour. We have

\[ r_p = r_{max} \cdot \exp(-k \cdot \| I_p - (g \ast I)_p \|) \]  

where \( g \) is a Gaussian function with standard deviation \( \sigma = 5 \), which is used to estimate the smooth degree by means of the convolution operation. \( r_{max} \) denotes the maximal radius that is set to 5 in our work.

A new trimap is generated based on the erosion and dilation operations of the initial face boundary and the corresponding size \( r \) of the unknown region. We erode and dilate less than \( r_p \) pixels for the foreground and the background on the either side of the initial face boundary pixel \( p \). This means that the corresponding structuring element employs the circle structure element with the computed radius \( r \). An example is illustrated in Fig. 8, where the second image is the segmentation result of the original image (a). Fig. 8(c) is the trimap based on the consistent square structuring element whose width is 4 pixels, where white, gray, and black regions correspond to the face, unknown, and background regions, respectively. The obtained trimap by our proposed adaptive approach is shown in Fig. 8(d). It can be found that different trimap sizes are obtained according to the smooth feature of the initial face boundary.

In order to estimate the \( \alpha \) in the unknown region \( \Omega_U \), we define the following energy function:

\[ E = E_1 + \lambda E_2 \]  

where \( E_1 \) is the error energy, measuring the degree of data estimation, and \( E_2 \) is the smoothing energy, denoting the changes between adjacent \( \alpha \) values.

The energy \( E_1 \) is defined as:

\[ E_1 = \sum_{(m,n)\in \Omega_U} \left[ I_{m,n} - \alpha(m,n)\tilde{F}_{m,n} - (1 - \alpha(m,n))\tilde{B}_{m,n} \right]^2 \]  

Here, \( L \) is a normalized factor for the weights \( w_{j,k} \) similar to those in [35], which are defined as \( \alpha^2d \) and \( (1 - \alpha)^2d \) for the foreground and the background, respectively. \( d \) denotes the distance with respect to the pixel \( (j,k) \). \( \delta_{m,n}^F \) and \( \delta_{m,n}^B \) denote the regions centered on the pixel \((m,n)\). In our work, a circular region with a radius of 11 is employed to perform the estimation.

The smoothing term \( E_2 \) is defined as:

\[ E_2 = \sum_{(m,n)\in \Omega_U} \min \left\{ (f_k \ast \alpha)^2_{m,n}, k = 0, 1, \ldots \right\} \]  

where \( f_k \) denotes a filter that corresponds to the direction \( k \). It measures the change of adjacent \( \alpha \) in a certain orientation, which encourages \( \alpha \) to change smoothly. In our work, we perform the filtering process based on four 4-tap filters with respect to the 0, \( \pi/4 \), \( \pi/2 \), \( 3\pi/4 \) directions.

To minimize the energy function \( E \), we use the gradient descent optimization method. The gradient of energy \( E \) can be written as

\[ \nabla E = 2 \sum_{(m,n)\in \Omega_U} \left[ I_{m,n} - \alpha(m,n)\tilde{F}_{m,n} - (1 - \alpha(m,n))\tilde{B}_{m,n} \right] \cdot \left[ \tilde{B}_{m,n} - \tilde{F}_{m,n} \right] + 2 \sum_{(m,n)\in \Omega_U} (f_{m,n} \ast (f_{m,n} \ast \alpha))_{m,n} \]  

At each iteration, we update the \( \alpha \) value as

\[ \alpha_{n+1}(m,n) = \alpha_n(m,n) - \tau \nabla E_{m,n} \]  

where \( \tau \) is the stepsize, which is used to minimize the energy \( E \) along the direction and set to 1.5.

The energy minimization problem is an iterative optimization process, which consists of three steps. The first step is the
initialization of $F$ and $B$ in unknown region $\Omega_U$ according to (11) and (12). The second step is the update of $\alpha$ value based on the gradient descent optimization. The initial $\alpha_0(x, y)$ for $(x, y) \in \Omega_U$ will be set to 0.5. Then (15) is used to perform the update of the matte value. The final step is $(F, B)$ refinement. If the condition $\alpha(x, y) > 0.99$ for $(x, y) \in \Omega_U$ is satisfied, we will update the face with this matte value set to 1. For the case of $\alpha(x, y) < 0.01$ for $(x, y) \in \Omega_U$, the pixels will be classified to the background. Otherwise, in the case of $0.01 < \alpha(x, y) < 0.99$, the value of pixel $(x, y)$ will be treated as the mixed result by the foreground $\Omega_f$ and background $\Omega_b$.

VI. TRACKING BASED FACE SEGMENTATION

A. State Prediction

The face segmentation in the successive frames is achieved by the tracking technique. The first step to track the face in the current frame $n$ is the prediction of the state space in the previous frame $(n - 1)$. Here the state space refers to the position of the face region.

A second-order autoregression model is used to describe the state change.

$$s_n = As_{n-1} + Bs_{n-2} + CN(0, \Sigma)$$

(16)

where state $s$ denotes the position of the face mask. A, B, C, and $\Sigma$ are the model parameters, which can be learned from the previous tracks. When the object’s movement is smooth, AR model can generally provide better prediction result. However, human object usually changes the direction with sudden movement, which leads to the bad prediction for the following segmentation. In order to avoid this worse case, we always check the error between the prediction and the original positions, and select the best one. In addition, a small window centered at the prediction is used to find the best matched position, which is set to $5 \times 5$ pixels. The sum of absolute difference (SAD) is used as the similarity measure between two regions.

B. Segmentation Using Updated Mask

After the state prediction, we obtain the coarse mask in the current frame $n$. In order to reduce the computation time, we adopt two modes to perform face segmentation. The first is based on a trimap, which consists of three regions, i.e., foreground, background, and unknown regions.

As shown in Fig. 9, $O_{n-1}$ denotes the candidate face region in the $(n-1)$th frame. The projected area $O_n$ is obtained by prediction in the $n$th frame. We perform the erosion and dilation morphological operations on the projected region $O_n$. The obtained regions are denoted by $O_n^f$ and $O_n^b$, respectively. The structuring element used is a square structuring element whose width is 10 pixels. Using (2), (3), and (6), we can calculate the data and smooth costs for the nodes in the unknown region. The minimum cut is then employed to assign the label to each node.

The first mode runs very fast due to the small set of nodes, while it also can achieve similar segmentation result as the fine segmentation when human face has no unexpected change. However, in order to avoid the possible accumulated errors caused by the noise effect in the first mode, we also need the second mode, i.e., fine segmentation, to correct the mistakes. As shown in Fig. 9, the pixels in the face and unknown regions are considered as the nodes in the constructed graph. The minimum cut is then employed to assign the label to each node.

VII. EXPERIMENTAL RESULTS

In this section, we evaluate our proposed face segmentation method with some color images and video sequences.

A. Robustness to the Face Detection Window

First, we investigate the impact of the face detection window on the segmentation performance. It is noted that the detection window with a certain size may not cover the actual face region accurately. Thus it is necessary to analyze the effects of the possible positions obtained by the face detection algorithm on the segmentation results. Fig. 10 shows examples which consists of different manually labeled windows, where the first window (Fig. 10(a)) fits well for the boy face compared to others. The second (Fig. 10(b)) is on the left of the face, while the third (Fig. 10(c)) has the smaller size. The final window (see Fig. 10(d)) contains all the boy’s face and many background regions. The segmentation results are given in the second row, which shows that our approach is robust to the size or shift of the detection window with respect to actual face position. We can see that the boy’s face can also be extracted successfully even when the detection window cannot cover the face well.
B. Evaluation Using Real-time Videos

We next validate our proposed segmentation method on the real-time videos, which were captured by a Philips web camera. The frame rate is set to 30 fps with the frame size of QVGA (320 \times 240).

1) Face segmentation under different poses and clutter background: We evaluated the proposed system in two different cases. Firstly, we tested face segmentation for different poses, size and movement. Some representative frames are shown in the first and third rows of Fig. 11. We alternately changed the pose to evaluate the system’s robustness. The local movements of human face are tested under the stationary background. The face region with different lighting can be observed by adjusting the view of the face to the light source. Their corresponding segmentation results are given in the second and the fourth rows, respectively. From the segmented results, it is clear that the human faces are segmented accurately. In addition, we also evaluate the segmentation in the case of occlusion, where some parts of face region was lost due to the human face moving out of the camera view. As shown in the fourth row, the lost face region is recovered accurately after it moves back into view again.

Secondly, we evaluate the proposed algorithm with clutter background. Some representative frames are shown in the third and fifth rows of Fig. 11. We can see that two persons first appear in the background with arbitrary movements. Then the third person joins in increase the complexity of the background. The segmentation results given in the fourth and last rows indicate the good performance for our method.

2) Face segmentation under abnormal lighting: We then evaluate the proposed system under abnormal lighting conditions. Some representative frames are shown in the first and third rows of Fig. 12. In this experiment, we first turn off some of the ceiling lights in order to generate the abnormal lighting conditions. The original images are shown in the first six frames in Fig. 12. Then the lights are turned on normal conditions as illustrated in the next four frames of Fig. 12. The corresponding segmented results are contained in the second and the last rows of Fig. 12, which show the robustness of our system to the abnormal lighting.

3) Face segmentation in dynamic background: In order to further evaluate the effectiveness of the presented system under complex situations, we next select the outdoor scenes as the dynamic backgrounds. The first experiment with the representative frames are given in Fig. 13. The background includes the moving cars and the pedestrian that appeared randomly. Apart from the above movements, we also perform camera panning during the face segmentation to test the system’s robustness, which can be observed from the first and third rows in Fig. 13. The final results are given in the second and the last rows, respectively. We can see that the good performance can be achieved even in a dynamic background.

The second experiment is performed with the camera moving around the object. Some representative frames are shown in the first and third rows of Fig. 14. The moving backgrounds with complex scenes are observed along with the camera movement. Since the camera is held by hand, some other camera motions, such as tilt and pan, are also generated which increase the complexity of the background. From the segmentation results in the second and the last rows of Fig. 14, we can see that satisfactory results can be achieved by our method.

C. Comparison with Other Methods

In order to perform the comparison with Graph Cut [2] and GrabCut [4] methods, we downloaded a free version of the software “Microsoft Expression Graphic Designer” from the website as recommended by the authors, because GrabCut will be a part of Microsoft Expression Graphic Designer. In addition, the LazySnap++, which is an version of Lazy
Snapping tool used for the video object cut and paste in [6], is also considered in the comparison. It is known that the interactive efforts by the user are required for these methods. For Graph Cut approach [2], a user marks a few lines on the image by dragging the mouse cursor to specify an object or background. To make it easier for the user to define the foreground, GrabCut allows modest interaction by dragging a rectangle around the desired object [4]. Unlike the previous snapping version, a user only needs to mark a few strokes on the object for the LazySnap++ tool.

For fair comparison, further user editing as in [4] and [5] will not be used because the desired result can be obtained by additional user interactions which give an unfair advantage. Here, we only compare the segmentation results using GrabCut algorithm after applying the first user-defined region. Fig. 15 shows the comparison results for some of the frames of the video sequence given in Fig. 11. The original frames are illustrated in the first row, while the third, fourth, and fifth rows display the results of GrabCut, GraphCut, and LazySnap++, respectively. We can see some false alarms can be found for the existing interactive methods. Further edits are needed to fix the missed or false segmentation. Our results are shown in the final row, which shows good performance for extracting human faces. The similar results can also be observed in Fig. 16, where some face images with different backgrounds and lighting conditions in Caltech 101 [41] are tested.

In order to perform objective comparison, we first manually segmented the reference maps (or ground truths) for the test images, which are shown in the second rows in Fig. 15 and Fig. 16. The spatial distortion [40] is used as the objective criterion, which is defined as the ratio of the difference between the segmented and reference binary masks (i.e., the binary "XOR" operation) to the reference binary mask. As shown in Table I, it is evident that our algorithm has lower distortion measures than those from other methods and these measures are well matched with subjective evaluation.

From the above comparisons, we can see that our proposed automatic face segmentation method can compete with the well-known interactive methods with desired results.

D. Computational Complexity Analysis

We next evaluate the computational complexity of the proposed real-time face segmentation method. In our method, we use three parts to speed up the face detection in the real-time video. Since a lot of non-face regions are rejected after
the skin-color filtering and rejector cascade, the computation load for the final boosting stage will be reduced significantly. The detector runs at 13 ms per frame of QVGA size (320 x 240 pixels) on a CPU of 1.85GHz. The coarse-to-fine segmentation will be first performed when a new face is detected. Then the tracking based segmentation is employed in the successive frames. A lot of computational time can be saved due to the state prediction and fast segmentation. Table II shows the execution time consumed for the segmentation stage on the same PC. When the face moves out of the camera view, the face detection algorithm will be activated to find the new human face.

E. Extensions

The proposed real-time segmentation system for human face can be easily extended to other applications. For example, if the coarse segmentation is performed on the appropriately defined body region, we can extend this work to solve the more challenging "head-shoulder segmentation" problem. Because of the limitation of this paper, more details on this application can be found on the web site [42].

VIII. CONCLUSION

In this paper, we proposed an accurate unsupervised human face segmentation method, which extracts faces from real-time videos. We have demonstrated that this method can provide effective and robust segmentation for frontal view faces, which can achieve similar or superior results compared to those of the well-known interactive techniques.

In our system, a boosting face detector is first developed, which can rapidly find faces with high detection rates. Secondly, we developed a face rejection cascade to speed up the detection process by rejecting most of the negative samples while retaining all face samples. Thirdly, a coarse-to-fine segmentation approach is proposed based on the min-cut optimization. Finally, we presented a new matting algorithm to estimate the alpha-matte based on an adaptive trimap generation method.

REFERENCES


TABLE II

| Face region size (pixels) | 1089 | 2028 | 2687 | 3971 | 6086 | 7666 | 10682 | 15961 | 21380 |
| Time (msec.)             | 10.94 | 14.70 | 18.49 | 24.21 | 27.56 | 41.80 | 44.41 | 58.66 | 61.58 |


Hongliang Li (M’06) received his Ph.D. degree in Electronics and Information Engineering from Xi’an Jiaotong University, China, in 2005. From 2005 to 2006, he joined the visual signal processing and communication laboratory (VSPC) of the Chinese University of Hong Kong (CUHK) as a Research Associate. From 2006 to 2008, he was a Postdoctoral Fellow at the same laboratory in CUHK. He is currently a Professor in the School of Electronic Engineering, University of Electronic Science and Technology of China. His research interests include image segmentation, object detection and tracking, image and video coding, and multimedia communication system.

King Ngai Ngan (M’79-SM’91-F’00) received the Ph.D. degree in Electrical Engineering from the Loughborough University in U.K. He is currently a chair professor at the Department of Electronic Engineering, Chinese University of Hong Kong, and was previously a full professor at the Nanyang Technological University, Singapore, and the University of Western Australia, Australia. Professor Ngan is an associate editor of the Journal on Visual Communications and Image Representation, U.S.A., as well as an area editor of EURASIP Journal of Signal Processing: Image Communication, and served as an associate editor of IEEE Transactions on Circuits and Systems for Video Technology and Journal of Applied Signal Processing. He chaired a number of prestigious international conferences on video signal processing and communications and served on the advisory and technical committees of numerous professional organizations. He is a general co-chair of the IEEE International Conference on Image Processing (ICIP) to be held in Hong Kong in 2010. He has published extensively including 3 authored books, 5 edited volumes and over 200 refereed technical papers in the areas of image/video coding and communications.

Professor Ngan is a Fellow of IEEE (U.S.A.), IET (U.K.), and IEAust (Australia), and an IEEE Distinguished Lecturer in 2006-2007.

Qiang Liu received his B.E. degree from Anhui Institute of Architecture and Industry, Hefei, china in 2004 and M.S. degree from Huaqiao University, Quanzhou, Fujian in 2006, respectively. From Jan. 2007 to July 2008, he was a Research Assistant in the visual signal processing and communication laboratory of the Chinese University of Hong Kong. He is currently pursuing his Ph.D. degree at the Department of Electronic Engineering, the Chinese University of Hong Kong, Hong Kong. His research interests are face detection and tracking, object segmentation, object-based coding.