A Visual Voice Activity Detection Method with Adaboosting

Qingju Liu, Wenwu Wang, Philip Jackson
Centre for Vision, Speech and Signal Processing
University of Surrey, Guildford, UK
{Q.Liu,W.Wang,P.Jackson}@surrey.ac.uk

***Abstract*—Spontaneous speech in videos capturing the speaker's mouth provides bimodal information. Exploiting the relationship between the audio and visual streams, we propose a new visual voice activity detection (VAD) algorithm, to overcome the vulnerability of conventional audio VAD techniques in the presence of background interference. First, a novel lip extraction algorithm combining rotational templates and prior shape constraints with active contours is introduced. The visual features are then obtained from the extracted lip region. Second, with the audio voice activity vector used in training, adaboosting is applied to the visual features, to generate a strong final voice activity classifier by boosting a set of weak classifiers. We have tested our lip extraction algorithm on the XM2VTS database (with higher resolution) and some video clips from YouTube (with lower resolution). The visual VAD was shown to offer low error rates.**

I. Introduction

Mouth region movements consist of abundant information about human speech, and therefore can be useful for improving the intelligibility of noise embedded speech, since it is hardly affected by acoustic noise. A basic use of video signals is visual voice activity detection (VAD), which classifies whether a short time period, or frame, is silent or not, by exploiting the synchrony of coherent audio and visual cues. Visual VAD has potential applications in noise reduction, speech separation or extraction and speech recognition. Some visual VAD algorithms have been proposed [1], [2], [3]. The method in [1] first projects the mouth region into principal component space, then models silent and non-silent periods with a single Gaussian distribution and a Gaussian mixture distribution respectively for the decision rule. The algorithm in [2] uses a filtered dynamic visual feature calculated from geometric visual features with multi-thresholds for silence detection. The approach in [3] estimates lip motion based on complex discrete wavelet transform, then applies the hidden Markov model for the statistical characterization of the lip motion, which is finally thresholded for the VAD. However, the algorithms above use either only static or only dynamic features, and the features used are fixed for different objects (speakers). Also, any rotation of the head may greatly degrade the performance, which is however inevitable in most recordings.

To overcome those limitations, we propose a new voice activity detection algorithm shown in Fig. 1. In the off-line training process, we track the lip region and extract the visual features, both static and dynamic, that are informative about the voice activity. With the activity vector obtained from the clean soundtrack as training labels, we apply adaboosting [4] to automatically choose and combine the optimum visual features for VAD.

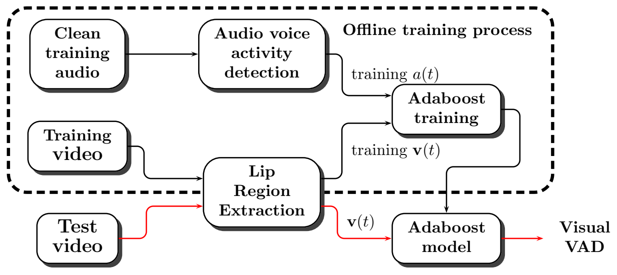

Fig. 1. Flow of our proposed visual voice detection system.

Extracting the lip region accurately and robustly is a prerequisite for the subsequent visual signal processing. However, it is difficult due to the deformable shape, the low lip-non-lip contrast, the rotation and the lighting luminance. Liew et al. [5] proposed a parameterized deformable model to approximate the lip contour, and obtained those parameters by joint probability maximization. However, the lip and non-lip clustering are affected by the luminance and the presence of teeth makes the two-class clustering more difficult. The algorithm in [6] combines greedy active contour models (ACMs) [7], [8] and adaptable template matching to find the expected lip contour of a specific speaker. It is straight forward to implement but it cannot cope with head rotation. Jang's method in [9] also uses the ACM, and adds the shape displacement energy to keep independent snake points from bending abruptly, but the performance suffers from the lip-non-lip contrast, since the method uses the classic image gradient as part of the objective function in the energy minimization process. Ong et al. [10] track lips by selecting linear predictors grouped into a rigid flock, without any prior shape information, and they obtained very good tracking performance but their method requires off-line training. To efficiently implement our visual VAD system, we also propose and present a lip extraction method, robust to the luminance and head rotation. In our proposed lip extraction algorithm, ACMs are used to drive the landmark points towards the lip contours. To cope with head rotation, we use rotational template matching. To avoid points bending abruptly (which often occurs if the resolution is very low), we add shape energy constraints. To prevent the snake points from converging to one point (which happens if the active contours mismatch at some frame), we add an area energy term.

The remainder of the paper is organised as follows. Section 2 introduces the detailed lip extraction algorithm. The adaboosting algorithm for visual VAD is presented in Section 3. Experimental results are demonstrated in Section 4, followed by the conclusions.

II. Lip extraction

Snake algorithms, or active contour models (ACMs) [7], [8] are energy minimization processes that pull the snake control points towards object contours or other salient image features such as color boundaries, depending on what kind of energy is defined. In our lip extraction process, a greedy snake algorithm [8] is exploited, as described below.

A. Snake initialization and initial templates

The initialization for our snake algorithm is similar to [6], with both manually selecting four corner points as in Fig. 2(a). After that, spline interpolations are applied to initialize the other eight control points on the lower and upper lips respectively, as in Fig. 2(b).

Then an initial rotational square template $\mathcal{T}_{init}(n)$ is assigned to each snake point $(x(n), y(n))$, with a rotation angle $\theta(n), n = 1, ..., 12$. Each square template is centered on one control point, rotated and scaled relative to the contour. For the convenience of presentation, we use the coordinate vector $\mathbf{s}(n) = [x(n); y(n)]$ to represent the n-th snake point. The rotation angle $\theta(n) \in [-\pi, \pi]$ is calculated as

$$\theta(n) = \begin{cases} c \arccos \frac{y(1)-y(\frac{N}{2}+1)}{\|\mathbf{s}(1)-\mathbf{s}(\frac{N}{2}+1)\|}, & n=1 \\ \theta(1) \pm \pi, & n = \frac{N}{2}+1 \\ c \arccos \frac{x(n+1)-x(n-1)}{\|\mathbf{s}(n+1)-\mathbf{s}(n-1)\|}, & \text{otherwise} \end{cases},$$

where $N = 12$ is the number of snake points, $\|\cdot\|$ is the Euclidean norm and

$$c = \begin{cases} 1, & \text{if } \begin{cases} x(1)>x(\frac{N}{2}+1), & n=1 \\ y(n+1)<y(n-1), & n\neq 1 \end{cases} \\ -1, & \text{otherwise} \end{cases}.$$

We use the rotational template to adapt to head motion, which makes our method more reliable and less likely to fail under variations in head rotation. Also, the side length $L(n)$ of each square template varies depending on its location but remains constantative. Fig. 3 shows the first, the fourth and the eighth initialized templates.

B. Snake energy minimization

As in [7], [8], the overall contour energy is computed as:

$$E = \sum_{n=1}^{N} (E_{int}(n) + E_{ext}(n)), \tag{1}$$

where $E_{int}(n)$ and $E_{ext}(n)$ are respectively the internal and the external energy for the n-th point.

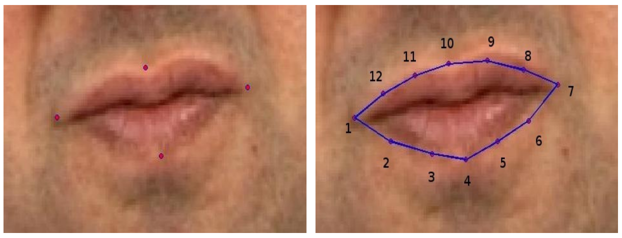

Fig. 2. Snake initialization process. (a) Manually selected four points. (b) Initial contour after spline interpolation.

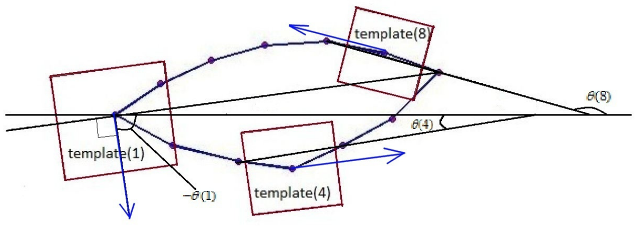

Fig. 3. Initialized templates $\mathcal{T}_{init}(n)$ for each snake point.

$E_{int}(n)$ is determined only by the position of the snake points, and contains the continuity energy $E_{cont}(n)$ and the curvature energy $E_{curv}(n)$:

$$E_{int}(n) = \alpha(n) E_{cont}(n) + \beta(n) E_{curv}(n), \tag{2}$$

and

$$E_{cont}(n) = \left| \|\mathbf{s}(n) - \mathbf{s}(n-1)\| - \bar{d}(n) \right| / \bar{d}(n),$$
$$E_{curv}(n) = \left\| \frac{\mathbf{s}(n) - \mathbf{s}(n-1)}{\|\mathbf{s}(n) - \mathbf{s}(n-1)\|} - \frac{\mathbf{s}(n+1) - \mathbf{s}(n)}{\|\mathbf{s}(n+1) - \mathbf{s}(n)\|} \right\|^2, \tag{3}$$

where $|\cdot|$ is the modulus, $\bar{d}(n)$ is the average distance between neighboring pixels ($\bar{d}(n)$ has two values, depending on whether it is an upper lip point or a lower lip point), $\alpha(n)$ and $\beta(n)$ are weights.

$E_{ext}(n)$ is, however, much more complex and depends on the image itself. In our proposed algorithm, we define three energy components:

$$E_{ext}(n) = \gamma_1(n) E_{temp}(n) + \gamma_2(n) E_{disp}(n) + \gamma_3(n) E_{area}(n), \tag{4}$$

where $\gamma_1(n), \gamma_2(n)$ and $\gamma_3(n)$ are weighting parameters and

$$\begin{aligned} E_{temp}(n) &= -\text{Corr}(\mathcal{T}_{init}(n), \mathcal{T}_{temp}(n)), & \text{(a)} \\ E_{disp}(n) &= \|\overline{\mathcal{C}_{init}} - \overline{\mathcal{C}_{temp}}\|, & \text{(b)} \\ E_{area}(n) &= \exp \left| \frac{\mathcal{A}_{temp}}{\mathcal{A}_{init}} + \frac{\mathcal{A}_{init}}{\mathcal{A}_{temp}} - 2 \right|. & \text{(c)} \end{aligned} \tag{5}$$

In equation (5), Corr(·) calculates the normalized correlation coefficient between its arguments, $\overline{\mathcal{C}_{init}}$ is the normalized initial contour while $\overline{\mathcal{C}_{temp}}$ is the normalized current (temporary) contour. The term 'normalized contour' means a contour centered at origin and with unit inside area. $\mathcal{A}_{init}$ is the initial area and $\mathcal{A}_{temp}$ is the current (temporary) contour area. The above components are analytically chosen as the external energy:

(a) E_{temp} is the template energy, which is the negative correlation between the initial template and the temporary template associated with each searching location. It drives one snake point to a region with similar neighboring pixel distribution as the original associated lip template. We use the rotational template matching, therefore it is robust to head rotation.

(b) E_{disp} is the displacement energy, which drives the snake contours to be similar to the original lip shape, and prevents independent snake points from converging separately to irregular contours.

(c) E_{area} is the area energy, which increases exponentially when the area inside the contour is either too big or too small as compared to the initial area.

The greedy snake algorithm [8] is implemented by updating each snake point $\mathbf{s}(n)$ to one of its neighboring pixels in a search area that minimizes $E_{int}(n) + E_{ext}(n)$ iteratively. The neighborhood, i.e., the search area, is defined by an ellipse area centered at the current snake point $\mathbf{s}(n)$ and aligned with its template:

$$[\tilde{\mathbf{s}}(n) - \mathbf{s}(n)]^T \mathbf{V}\mathbf{D}\mathbf{V}^T [\tilde{\mathbf{s}}(n) - \mathbf{s}(n)] < 1, \tag{6}$$

and

$$\mathbf{V} = \begin{pmatrix} \cos(\tilde{\theta}(n)) & \sin(\tilde{\theta}(n)) \\ -\sin(\tilde{\theta}(n)) & \cos(\tilde{\theta}(n)) \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{pmatrix},$$

where $\tilde{\theta}(n) = \pi/2 + \theta(n)$, and a and b are the major and minor axes of the scanning ellipse. We choose ellipses perpendicular to the lip contour as search areas, since lip muscle movements in mouth opening and closing coincide with these directions. Every time, we scan one neighboring pixel for the possible update of a snake point, we assign a temporary template $\mathcal{T}_{temp}(n)$ to it, centered at the scanning pixel with the rotation angle $\theta(n)$. In each iteration, all snake points are updated once. We found that satisfactory results can be obtained after two iterations. Further iterations yield better results, but involve a higher computational load.

C. Static lip features

After the lip extraction, we extended three static visual features associated with each video frame:

- Width $W = \|\mathbf{s}(1) - \mathbf{s}(7)\|$.
- Height $H = \|\mathbf{s}(4) - \mathbf{s}(10)\|$.
- Mean intensity A of the rotational rectangle in the mouth center with a rotation angle of $\pi/2 + \theta(1)$ and size of $W_0/2 \times H_0/2$.

The first and second are geometric features, which can be used to model the lip shape movement. The third is an appearance-based feature, which captures key inner lip cues, such as movement of the teeth and tongue.

III. Voice activity detection with Adaboosting

The voice activity detector is a classifier that detects whether the speaker is silent or not in each frame. To obtain a classifier with a high accuracy, i.e. a so-called *strong* classifier with a low error rate, we can apply the adaboosting technique [4], which combines or boosts a set of *weak* classifiers (with high error rates, but overall better than random guessing).

A. Visual features

To detect the voice activity from the video, we mainly focused on the movement of lips over a short time period and the current lip shape. Hence, we combined both static and dynamic features. Denote $d_W(\tau) = W(t) - W(t-\tau)$ where t is the discrete frame number and τ is the frame offset. In the same way we get difference features $d_H(\tau)$ and $d_A(\tau)$. Then we define a new visual feature vector as

$$\begin{aligned}\mathbf{v}(t) = [&W(t), H(t), A(t), d_W(T), d_H(T), d_A(T),\\ &d_W(T-1), ..., d_W(-T), d_H(-T), d_A(-T)]^T,\end{aligned} \tag{7}$$

and $2T + 1$ frames are used to obtain the above visual feature. The first three elements of $\mathbf{v}(t)$ correspond to the static geometric and appearance features, and the other elements are dynamic features. However, it is less clear which elements in the above feature vector are most informative about the voice activity. Information redundancy could also be a problem, for example, $d_W(T)$ and $d_W(T-1)$ are highly coherent. Yet, as discussed in the next section, adaboosting can be employed as a feature selection process, since in each iteration it selects one feature that best classifies the outputs. This adaboosting selection process has been successfully used in the Viola-Jones object detection algorithm [11], for example.

B. Adaboost training

We apply a simple threshold to the energy of the clean speech signal associated with the training video to get the audio voice activity vector as training labels $a(t)$. Then adaboost training is applied to the extracted visual features. The weak classier $h_i(\mathbf{v}(t), m, p, \phi)$ used in the i-th iteration contains the selected m-th element of $\mathbf{v}(t)$, a threshold ϕ and a polarity p:

$$h_i(\mathbf{v}(t), m, p, \phi) = \begin{cases} 1, & \text{if } pv_m(t) > p\phi \\ 0, & \text{otherwise} \end{cases}. \tag{8}$$

Suppose ϵ_i is the error rate in the i-th iteration, which is the sum of the updated weights associated with the samples misclassified in the i-th iteration. Then, in the $(i+1)$-th iteration, the weights of misclassified samples are amplified by a factor of $\frac{1-\epsilon_i}{\epsilon_i}$ before the normalization of all sample weights.

We abbreviate the i-th classifier as $h_i(\mathbf{v}(t))$, so the final combined strong classifier $\mathcal{C}(\mathbf{v}(t))$ is a weighted voting process:

$$\mathcal{C}(\mathbf{v}(t)) = \begin{cases} 1, & \text{if } \sum_{i=1}^{I} w_i h_i(\mathbf{v}(t)) > \frac{1}{2}\sum_{i=1}^{I} w_i \\ 0, & \text{otherwise} \end{cases}, \tag{9}$$

where $\frac{1}{2}\sum_{i=1}^{I} w_i$ is the voting threshold and $w_i = \log \frac{1-\epsilon_i}{\epsilon_i}$.

To be more cautious about rejecting silent frames, i.e., if we want fewer active frames to be misclassified as silent frames while tolerating the situation that more silent frames are misclassified as active frames, the voting threshold $\frac{1}{2}\sum_{i=1}^{I} w_i$ can be slightly increased.

IV. Experimental Result

A. Lip extraction

1) Parameter setup: Suppose in the first frame, the initial lip width is W_0 and the height is H_0 after manual selection of four points (measured by the pixel number). The parameter setup is given in Table I. Note the values of W_0 and H_0 will change depending on the video resolution. As an example, W_0 and H_0 in Fig. 4 are 88 and 34 respectively.

TABLE I
Parameter setup for the greedy snake algorithm

Snake Points		n=1,7	**otherwise**
Energy Parameters	$\alpha(n)$	0.4	4
	$\beta(n)$	0.1	1
	$\gamma_1(n)$	4	4
	$\gamma_2(n)$	5	5
	$\gamma_3(n)$	1	1
Search Area	a	M	$M = \frac{W_0}{4}(1 + exp(-\frac{3W_0}{H_0}))$
	b	$0.5M$	$0.5M$
Square Template length $L(n)$		$1.5M$	M

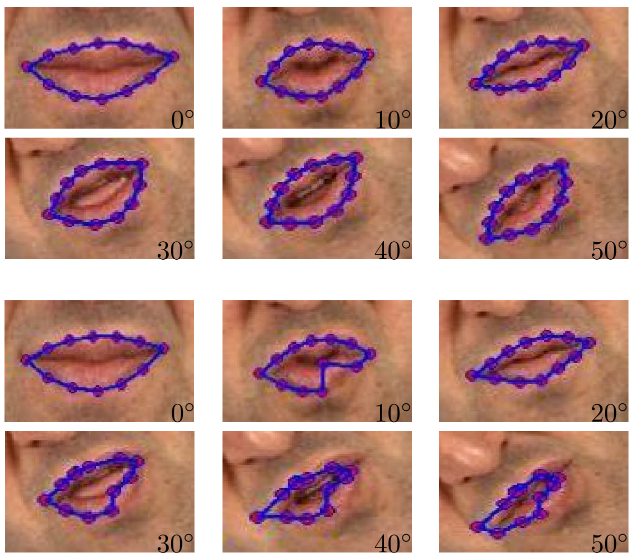

Fig. 4. Example of lip extraction results with the proposed method applied to XM2VTS. We rotated the above frame images by six angles: 0°, 10°, 20°, 30°, 40°, 50°. Row 1 and Row 2 are with rotational template matching while Row 3 and Row 4 are with upright template matching.

2) Lip extraction results: We first applied the proposed lip extraction algorithm on the XM2VTS database [12], which was recorded with high resolution, and subjects seldom moved or rotated their heads. To test our algorithm, we rotated each frame by $0.2°$ cumulatively in an anti-clockwise direction. Fig. 4 shows the lip contour with (the first and the second rows) and without rotational template matching with different rotation angles. With rotational templates, our method is more robust to the head motion and accurately tracks the lip contour.

However, most recordings in real life do not have the high resolution around the mouth (e.g. the lip region shown in Fig. 4 occupies about 88×34 pixels). To test the robustness of our algorithm, we used several videos downloaded from YouTube with relatively low resolutions. In these videos, the speakers' heads rotated, and the lighting conditions varied. Results may be found on-line via the following link: http://www.youtube.com/watch?v=j7jr7sh5fmQ (the lip region occupied about 45×17 pixels).

B. Adaboost Training

1) Parameter Setup: To provide data of sufficient duration for adaboost training, we used another long video clip from YouTube (available from http://www.youtube.com/watch?v=zXBpW8GCDtY). The first 120 seconds were used for adaboost training, and the following 30 seconds for testing.

In the training process, the labels (the voice activity for each frame) were obtained by thresholding the short-time energy of the clean audio signal. We set $T = 10$, therefore, the neighboring frames of 800 ms (frame step is 40 ms) could influence the current time frame. In adaboosting, 100 iterations (weak classifiers) were applied ($I = 100$).

2) Visual VAD: After the training process, we obtained an adaboost model, then we applied this model to the 30-second test video via majority voting. Our algorithm accurately detected most of the active frames. Outliers, i.e., a silence frame with neighboring frames being active, were set as active in the post-processing. We implemented the method in [1] for comparison. The extracted lip region (the so-called 'ROI', i.e. region of interest) was first mapped into a 6-dimensional space via principal component analysis, then the first order difference was used as the visual feature. The voice class was modeled with a 5-kernel Gaussian mixture model (GMM), and instead of a single Gaussian distribution, the non-voice class was approximated with a 3-kernel GMM. Fig. 5 shows the comparison of our proposed method against the references from the clean audio and the algorithm in [1].

We used the error rate as a criterion to evaluate the performance:

$$\epsilon = \frac{1}{Q}\sum_{t=1}^{Q}(\mathcal{C}(\mathbf{v}(t)) \neq a(t)), \tag{10}$$

where Q is the number of frames and $a(t)$ is the reference VAD label. In the same way, we defined the positive error rate ϵ_p and the negative error rate ϵ_n when we only consider the frames for which $a(t)$ is positive or negative. With 120 seconds for training and $I = 100$, we got $\epsilon_p = 0.11$, $\epsilon_n = 0.21$ and $\epsilon = 0.14$. As a contrast, using method in [1], we got $\epsilon_p = 0.26$, $\epsilon_n = 0.40$ and $\epsilon = 0.30$.

To investigate how the number of iterations and the length of the training data influence the performance, we set the iteration number from 1 to 100, and the error rate tended to decrease

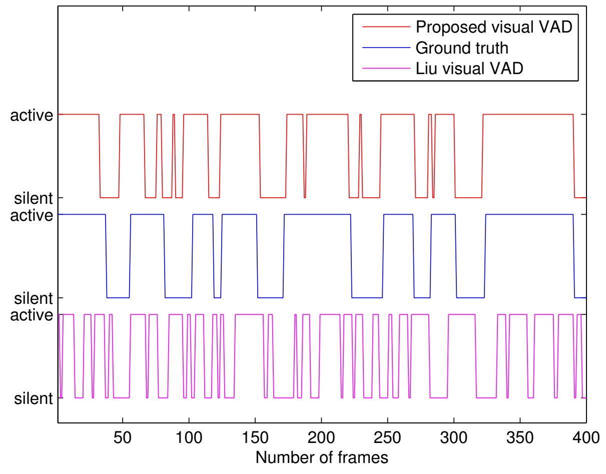

Fig. 5. Visual VAD on the test data against the ground truth and the method in [1]. The beginning 400 frames are shown. The adaboost model was trained on 120 s training data with 100 iterations and 40 ms frame step.

both in the training process and testing process. With post-processing to eliminate the outliers, we gained another 5% improvement. Fig. 6 presents the performance of our algorithm with respect to the number of iterations. We then increased the length of the training data from 10 seconds to 120 seconds, and the performance is shown in Fig. 7.

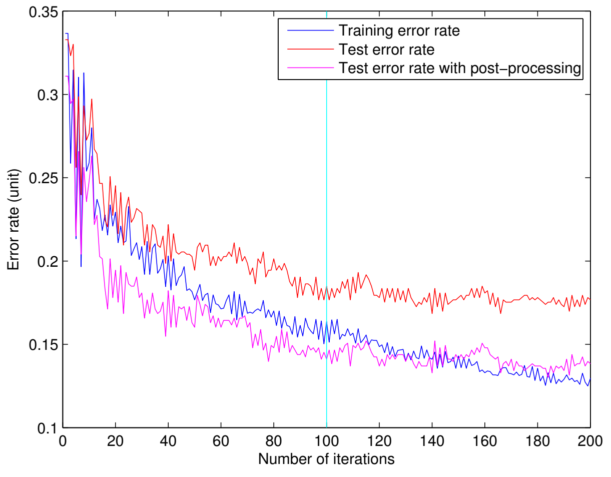

Fig. 6. Performance with respect to the iteration number.

V. Conclusions

A lip extraction method with rotational template matching and model constraints was implemented using a greedy snake algorithm. We used both static and dynamic features extracted from the lip region to form a classifier for visual voice activity detection, which was obtained by adaboost training. We tested our lip extraction on both XM2VTS database and some low-resolution videos from YouTube (the mouth region occupied about 45×17 pixels). The visual VAD algorithm applied to the low-resolution video demonstrates its good performance, offering low error rates.

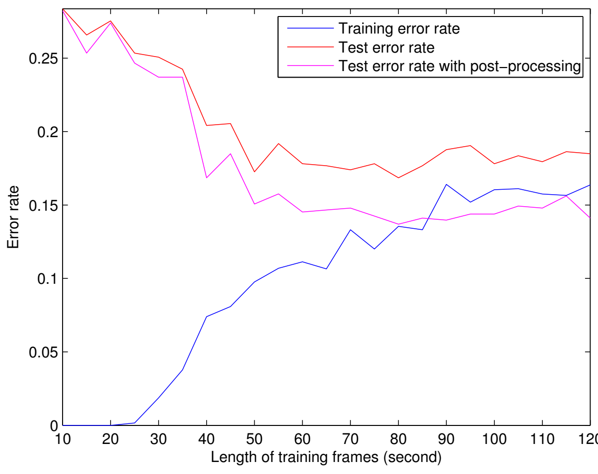

Fig. 7. Performance with respect to the length of training data.

Acknowledgment

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) (Grant number EP/H012842/1) and the MOD University Defence Research Centre on Signal Processing (UDRC).

References

[1] P. Liu and Z. Wang, "Voice activity detection using visual information," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, May 2004.

[2] D. Sodoyer, B. Rivet, L. Girin, C. Savariaux, J.-L. Schwartz, and C. Jutten, "A study of lip movements during spontaneous dialog and its application to voice activity detection," *The Journal of the Acoustical Society of America*, vol. 125, no. 2, pp. 1184–1196, Feb. 2009.

[3] A. Aubrey, Y. Hicks, and J. Chambers, "Visual voice activity detection with optical flow," *Image Processing, IET*, vol. 4, no. 6, pp. 463–472, Dec. 2010.

[4] Y. Freund and R. E. Schapire, "A short introduction to boosting," *Japonese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, 1999.

[5] A. Liew, S. Leung, and W. Lau, "Lip contour extraction using a deformable model," in *International Conference on Image Processing (ICIP)*, vol. 2, Sept. 2000, pp. 255–258.

[6] M. Barnard, E. J. Holden, and R. Owens, "Lip tracking using pattern matching snakes," in *the 5th Asian Conference on Computer Visio (ACCV)*, Jan. 2002.

[7] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," vol. 1, no. 4, pp. 321–331, 1988.

[8] D. J. Williams and M. Shah, "A fast algorithm for active contours and curvature estimation," *CVGIP: Image Understanding*, vol. 55, no. 1, pp. 14–26, 1992.

[9] K. S. Jang, "Lip contour extraction based on active shape model and snakes," in *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 7, Oct. 2007, pp. 148–153.

[10] E.-J. Ong and R. Bowden, "Robust lip-tracking using rigid flocks of selected linear predictors," in *the 8th IEEE Conference on Automatic Face and Gesture Recognition*, 2008.

[11] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2001, pp. 511–518.

[12] XM2VTS, Website, http://www.ee.surrey.ac.uk/CVSSP/xm2vtsdb/.