LIPREADING WITH SPIKING NEURONS ONE PASS LEARNING

R. Séguier, N. Cladel, C. Foucher, D. Mercier

Supélec, Team ETSN
Avenue de la Boulaie, BP28
35511 Cesson Sévigné, France.
{Renaud.Seguier, Nicolas.Cladel, Christophe.Foucher, David.Mercier} @supelec.fr

ABSTRACT

We present in this article a lipreading system implementing spiking neurons (STANs). A new preprocessing is proposed in order to reduce as much as possible the learning phase of the system. This training is done in one pass: the user pronounces once all of the words of the dictionary; the system is then able to recognize these words throughout different sessions during which the position and the chrominance of the images of the speaker's mouth strongly change.

Keywords: HCI, Man Machine Interaction, Lipreading, Visual Speech Recognition, Neural Network, Spiking Neuron

1 INTRODUCTION

Visual speech recognition is a problem on which many teams have been working since a score of years. The potential applications are important: audio-video processing for a robust speech recognition, lip-synchronization in audio-video flows, synthesis of talking heads. Nevertheless, there does not exist yet of really robust and noninvasive system making it possible to analyze the mouth shape in a flexible way. The realization of avatar model of a face for example is still done after a phase of heavy initialization from the user point of view [Leung01].

Our team works on HCI (Human Computer Interaction) and particularly on treatments in which the computer adapts better and better to the person in a natural way. We thus treat the lipreading under the single-speaker perspective: the objective is to produce a robust and powerful recognition system, adapted to a particular speaker and not requiring a very constraining training.

We present in this article a preprocessing sufficiently robust to allow a one pass learning: the speaker reads only once during a first session the whole dictionary to parameterize the recognition system. The robustness of the recognition is tested on different sessions during which the position of the mouth and its chrominance vary strongly.

After a short state of the art (section 2), we will present our lipreading system (section 3) and in particular the suggested preprocessing. The section 4 devoted to the results will end in a discussion on the future improvements.

2 State of the art

2.1 Preprocessing

Two types of preprocessing are distinguished according to whether they are based on low or high level information.

Low level parameters

An extraction of the low level parameters makes it possible to take into account many characteristics such as the appearance of the teeth, the tongue or the particular mouth shapes. Within the same session during which neither the mouth position nor the image chrominance change, these parameters make it possible to carry out a powerful classifier. A PCA (Principal Component Analysis) is sometimes directly applied to the image gray levels [Cueto00]. The projection of the mouth on the first 100 extracted components produces projection coefficients injected into the classifier. [Ség01] applies a VQ (Vector Quantization) to the gray levels in order to extract twenty code vectors which will be used for the identification of the elementary mouth shapes. These extractions based directly on the gray levels are not robust within the test framework carried out on different sessions during which the mouth is slightly shifted or the light comes from a different source.

High level parameters

Within the multi-session framework, it is then necessary to use high level parameters which will be invariants in translation, zoom and change of chrominance. Simple parameters are sometimes evaluated. One computes for example the height and the width of the mouth by carrying out a summation on the gray levels according to the lines and columns of the region surrounding the mouth [Baig99a]. One can also apply a deformable model to the outlines of the mouth [Yuill90], [Tian01]. Then the model parameters are sent to the classifier. Dynamic contours are also used: [Delma99] applies snakes to contours while [Sanch00] models the mouth by a B-Spline. Finally certain teams [Luett97] make mixed one between the low level (signatures in term of gray levels) and high level (dynamic contours) parameters. The disadvantage nevertheless of these high level parameters is that they integrate few details of the mouth (the tongue and the teeth for example are seldom taken into account): all the relevant information which would make it possible to separate particular shapes is not taken into account leading thus to a less powerful classifier.

2.2 Classification

In this state of the art we are interested primarily in the preprocessing, since it is the essential object of our contribution. With regard to the classification systems, two types of tools are mainly exploited: neural network (Multi-Layer Perceptron [Moris01], Time Delay Neural Network [Stief97], Spiking Neurons [Baig99b]) and the Hidden Markov Models ([Dupon00], [Wark00]).

3 LIPREADING SYSTEM

3.1 System

Our global lipreading system is organized according to two modules (Fig. 1). The preprocessing module makes it possible at every moment to identify the elementary shape of mouth by comparing the input images with referenced code vectors. These code vectors are specified by a Vector Quantization (VQ). The classification module consists of a network of STANs (Spatio-Temporal Artificial Neurons): each neuron is associated with a word of the dictionary and must recognize a particular impulses sequence.

3.2 Preprocessing module

We propose in this article a particular preprocessing which jointly uses high and low level parameters. The classifier is both robust during different sessions and powerful within the same session.

Let $\mathcal{E}$ be the image set of the first sequence of the first session. This dataset constitutes our learning base. One applies a VQ (KMeans) to $\mathcal{E}$ in order to extract some code vectors. Each code vector represents an elementary mouth shape (Fig 1). An influence region for each one of these code vectors is evaluated by means of a radius specific to each code vector. This radius r_j is equal to the distance between the code vector and the most distant example which is attached to him. The preprocessing module has as many outputs than there are evaluated code vectors. In exploitation phase, the output $y_j(t)$ of the code vector nearest to the input $X(t)$ is equal to:

$$y(t) = e^{-\frac{D^2(X(t),P_j)}{2r_j^2}}$$

Only the output of the code vector nearest to the input is activated.

All the words of the dictionary are present in the learning base $\mathcal{E}$, we thus generate for each one of these words a characteristic impulses sequence.

The supervised learning during which one specifies the weights of the STANs having to recognize each word is made only once and on the basis of impulses generated by the preprocessing module. Thus, the output of this module must be identical for a same word, under any session.

For example if the output associated with the first code vector characterizes a closed mouth following the VQ carried out during a first session, it

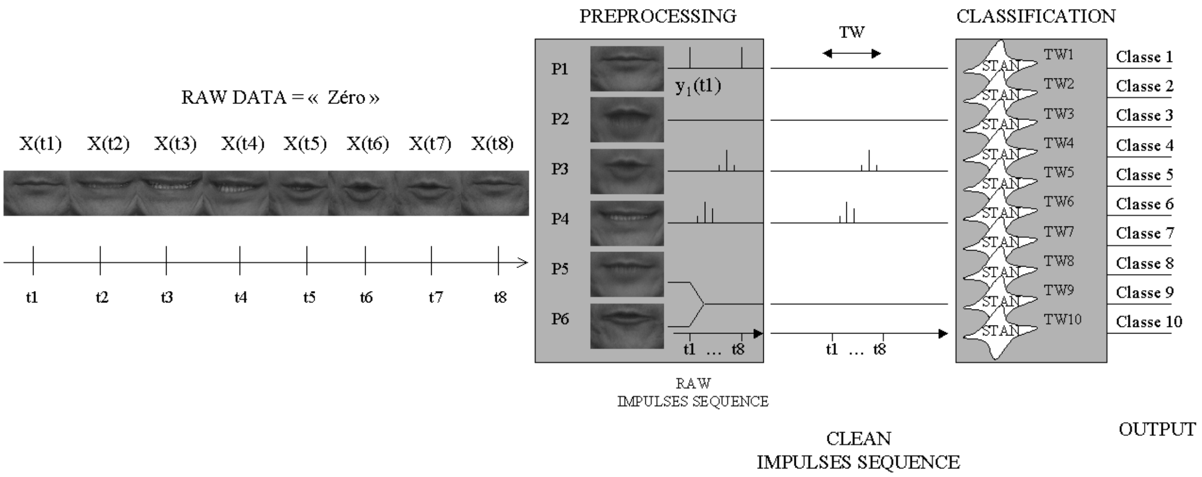

Figure 1: Lipreading System.

is necessary that this same output always characterizes a closed mouth during others sessions. However we do not know a priori which mouth shapes the code vectors characterize because of algorithm used (KMeans has a random initialisation). To overcome this problem, we propose to analyze each code vector generated by the VQ at the beginning of each session, and to recognize the form which it characterizes in order to reorganize the outputs of the preprocessing module and thus to make them coherent under any session.

Let us consider as an example the code vectors generated during three different sessions (Fig. 2)

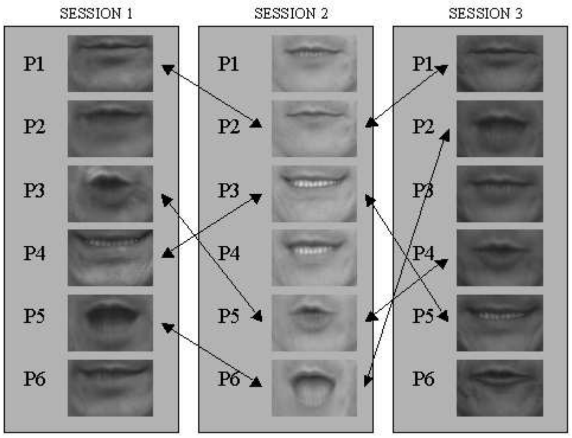

Figure 2: Three sets of code vectors characteristic of three sessions

They are only two high level parameters (the height and the width of the mouth) which will enable us to associate these code vectors two by two between different sessions. The method used to extract these two parameters is presented in appendix.

Implementation

Let us consider three different sessions during which a particular speaker pronounces all the words of the dictionary (digits from 0 to 9) several times running (Fig. 3). In the learning phase (during the first sequence of the first session) the parameters of the preprocessing and classification modules are defined. In the testing phase, certain preprocessing module parameters are specified in an unsupervised way during a so-called calibration phase (during the first sequence of each session), the tests being carried out strictly on all the other sequences of the session.

We will show in the section 4 that six code vectors are necessary. The high level parameters which we use permit to establish the links between four code vectors: a closed mouth (the code vector characterized by the mouth with the smallest height), a wide-open mouth (maximum height), a smile (maximum width) and a mouth forming a little "o" (minimal width). We can thus guarantee that the first four outputs are coherent under any session. Unfortunately these parameters are not sufficient to make it possible to link all the 6 code vectors. For this reason it is necessary to differently configure the output of the code vectors layer in order to allow an inter-session classification. We propose to make the fusion of the fifth and sixth output in order to produce finally a preprocessing module with only five outputs. As we said previously one and only one output emits an impulse, this fusion thus consists in simply generating an impulse on the fifth final output of the module when the code vector five or six emits an impulse (see Fig. 1).

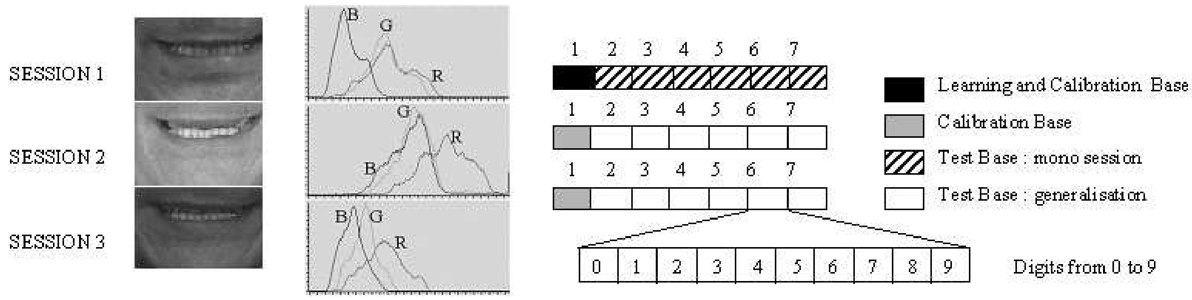

Figure 3: Learning, calibration and testing base.

3.3 Classification module

The STANs are spiking neurons which work in the complex domain. A sequence of impulses is converted into a vector X with complex values in the following way (Fig. 4).

The impulse of amplitude η_1 emitted at time t_1 is coded at current time t by the complex number:

$$x_j(t) = \eta_1 e^{-\mu_S \tau_1} e^{-i \arctan \mu_T \tau_1}$$

with

$$i = \sqrt{-1},\ \tau_1 = t - t_1 \text{ et } \mu_S = \mu_T = 1/TW$$

TW depends on the application and represents the size of the temporal window inside which one wishes to identify impulses sequences. When a new impulse η_2 is emitted at time t_2 on the same component, it is accumulated in the component j of the vector X:

$$\begin{aligned} x_j(t_2) &= \eta_1 e^{-\mu_S (t_2 - t_1)} e^{-i \arctan \mu_T (t_2 - t_1)} + \eta_2 \\ &= \rho e^{i\phi} \end{aligned}$$

and later:

$$x_j(t) = \rho e^{-\mu_S (t - t_2)} e^{-i \arctan tan\phi + \mu_T (t - t_2)}$$

Each component of the vector X is thus reactualized as soon as an impulse is presented to the input. The comparison between X and the weight W of the neuron itself characterized by a complex vector can be done by the means of an Hermitien distance D:

$$D(X, W) = \sqrt{\sum_{j=1}^{N} (x_j - w_j)\overline{(x_j - w_j)}}$$

knowing that $\overline{x}$ is the complex conjugate of x.

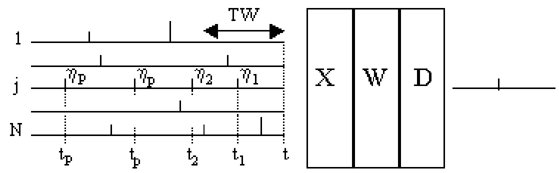

Figure 4: STAN : Spatio-Temporal Artificial Neuron.

The temporal duration TW of the sequence which is taken into account by the neuron is a parameter which remains to be defined by the user. We propose here to regulate TW in an automatic way to adapt to the duration of each word which will have to be recognized. Before and after each pronounced word, the person who expresses herself has a closed mouth. We detect the beginning and the end of each word by comparing each image with the code vector which corresponds to the closed mouth. It is thus possible during the learning phase to evaluate the duration of each word which must be recognized.

There are as many output STANs than of words to be recognized. The characteristic duration (TW) of each neuron is thus evaluated in an automatic way when the word having to be recognized is presented. The weight of the neuron is then the simple conversion of the impulses sequence into a complex vector.

In exploitation phase, we act in the same way in the sense that we evaluate for each sequence the beginning and the end of the word pronounced by comparing each image with the code vector of closed mouth found during calibration. Thus the impulses sequence which we send indeed in the layer of STANs is cleaned as indicated by the Figure 1. Each output neuron then compares its weight vector with the transposition into complex vector of the impulse sequence by measuring a distance D_k, k going from 0 to 9. The class will be identified by the neuron presenting the minimal distance D.

4 RESULTS AND CONCLUSION

4.1 Protocole

The images were acquired at a frequency of 50 Hz (after de-interlacing) in RGB and with a resolution of 288 X 192 pixels. Let us note that for the computation of the code vectors by Kmeans, these images converted into 256 gray levels were reduced by a factor 4 (72 X 48) in order to accelerate the processing.

Our objective is to recognize the ten digits (from zero to nine) in French expressed by only one speaker. We carried out three sessions during which the speaker pronounces all the ten digits eight times running. A sequence in a session corresponds to the pronunciation of the ten digits one after the others, the mouth being closed one half-second at least between each digit. There are thus 3 sessions, each one containing 8 sequences.

As one can see it on the Figure 3 the images of the three sessions are different in brightness, chrominance and position of the mouth. On the opposite during a session, the mouth evolves neither in position nor in chrominance.

The first sequence of the first session is used for the training, the first sequences of the two other sessions are used for calibration (evaluation of the code vectors), the seven remaining sequences in each one of these two sessions are used for the test.

4.2 Parameter setting of the system

It is necessary to fix a priori in the algorithm of KMeans the number of code vectors to be computed. In theory an optimal number of code vectors exists: not enough code vectors does not allow a good discrimination between rather close words, on the contrary too many code vectors prejudice generalization performance of the system. To evaluate this optimal number in an automatic way, we vary the number of code vectors during the learning phase and test the system generalization performance on the same session (sequences 2 to 8, session 1). The curve (Fig. 5) characterizing the percentage of good classification according to the code vector number thus makes it possible to fix at 6 the optimal code vector number in the preprocessing module. The results are expressed on average on ten tests for each different number of code vectors, the best and worse results are related to the graph. Figure 6 allow us to jauge standard deviation of the results during the increase in the number of code vectors. The more code vectors we have, the more the mouth shapes which characterising these code vector look like each other, and the more the impulses sequences at output of the preprocessing module will be different for the same word. Let us note that for this evaluation, all the preprocessing module outputs were preserved since it is about a classification in the same session.

The optimal number of code vectors being evaluated, it remains to calibrate the system for each session, i.e. to define the code vectors for each session. We use for that only the first sequence of each session. As we said previously KMeans is carried out, then a height and width evaluation of the mouths represented by the code vectors is made, which enables us to reorganize in an automatic way 4 first code vectors (see Fig. 2).

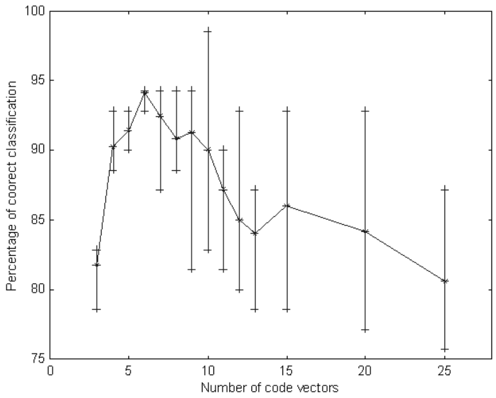

Figure 5: Percentage of correct classification versus number of code vectors.

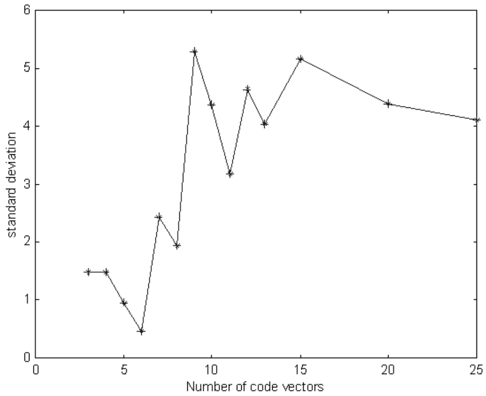

Figure 6: standard deviation of the results versus number of code vectors.

The weights of the STANs in the classification module are specified simply on the basis of the first sequence of the first session as we had proposed in section 3.3.

4.3 Tests

A learning was carried out on the first sequence of the first session, and the tests on the seven last sequences of the sessions two and three (see Fig. 3). The system is then able to recognize in an effective way the words which were pronounced in 81% of the cases on average on ten tests (standard deviation of the results: 4%). Let us note the stability of the results independently of the session number tested: 80% of good classification in the session two and 82% in the third session.

4.4 Discussion

Within a monosession framework, the algorithm is more powerful (87%) than within a multisessions framework (81%) when the fusion of outputs 5 and 6 of the preprocessing module is realized. The problem can come from the fusion of the two last outputs of the preprocessing module. Indeed the observation of the curves which enabled us to evaluate the optimal number of code vectors makes it possible for us to say that a monosession classification is more powerful with six distinct code vectors (94%) than with five (91%) as Figure 5 shows it (no fusion of the outputs were achieved). The fact of carrying out the fusion of the last two code vectors thus makes fall the results. It would be necessary to be able to link the code vectors five and six of all the sessions but with high level parameters a little finer: the deformable models or the snakes could be useful. Nevertheless, it appears difficult if not impossible to connect the code vectors $P2$ and $P6$ of session 1 with the code vectors $P3$ and $P6$ of the third session (see Fig. 2). The mouths shapes are different: KMeans thus did not produce the same types of code vectors, certainly because the mouths shapes variability in the calibrations bases were not identical.

At the present time, our system is unable to recognize words absent from the training base. Moreover it is adapted to only one speaker. If another speaker presents himself with a strongly different mouth, the system will fail. It would be thus interesting to work firstly with a multi-speaker large vocabulary training base. That would make it possible to initialize our system of recognition before the speaker adaptation phase presented in this article. Thus the system would be likely to recognize words not used by the speaker during this learning phase even if this recognition might be less powerful than for the words belonging to the training base. One could even imagine a natural training of the system during the exploitation of the recognition tool: the phase of specialization to a particular speaker on a large vocabulary (200 000 words for example) would be continuous and of this fact much less heavy.

Nevertheless the visual speech recognition on a large vocabulary is a difficult problem and we do not think to be able to reach the same recognition rate than on a restricted vocabulary. In the futur, we will take into account the sound to constitute a really robust multimodal system of speech recognition.

REFERENCES

[Baig99a] A. R. Baig, R. Séguier, and G. Vaucher. Image sequence analysis using a spatio-temporal coding for automatic lipreading. In *International Conference on Image Analysis and Processing*, 1999.

[Baig99b] A. R. Baig, R. Séguier, and G. Vaucher. A spatio-temporal neural network applied to visual speech recognition. In *ICANN*, 1999.

[Cueto00] P. de Cuetos, N. Chalapathy, and W. Andrew. Audio-visual intent-to-speak detection for human-computer interaction. In *ICASSP*, 2000.

[Delma99] P. Delmas, P.Y. Coulon, and V. Fristot. Automatic snakes for robust lip boudaries extraction. In *ICASSP*, 1999.

[Delma00] P. Delmas. *Extraction des contours de lèvres d'un visage parlant par contours actifs.* PhD thesis, Université Grenoble, 2000.

[Dupon00] S. Dupont and J. Luettin. Audio-visual speech modeling for continuous speech recognition. *IEEE Transactions on multimedia*, 2000.

[Férau01] Raphaël Féraud, Olivier J. Bernier, Jean-Emmanuel Viallet, and Michel Collobert. A fast and accurate face detector based on neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[Leung01] W. H. Leung and T. Chen. Creating a multiuser 3-d virtual environment. *IEEE Signal Processing Magazine, Immersive interactive technologie*, 2001.

[Luett97] J. Luettin. *Visual Speech and Speaker Recognition.* PhD thesis, Univesity of Sheffield, 1997.

[Moris01] S. Morishima. Face analysis and synthesis. *IEEE Signal Processing Magazine, Immersive interactive technologie*, 2001.

[Sanch00] M.U.R. Sanchez. *Aspects of facial biometrics for verification of personal identity.* PhD thesis, University of Surrey, 2000.

[Ség01] R. Séguier and David Mercier. A generic pretreatment for spiking-neuron. application on lipreading with stann (spatio-temporal artificial neural networks). In *International Conference on Artificial Neural Networks and Genetic Algorithms*, 2001.

[Stief97] R. Stiefelhagen, U. Meier, and J. Yang. Real-time lip-tracking for lipreading. In *Proc. of Eurospeech*, 1997.

[Tian01] Y. Tian, T. Kanade, and J. F. Cohn. Recognizing action units for facial expression analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2001.

[Wark00] T. Wark, S. Sridharan, and V. Chandran. The use of temporal speech and lip information for multi-modal speaker identification via multi-stream hmm's. In *ICASSP*, 2000.

[Yuill90] A.L. Yuille, D.S. Cohen, and P.W. Hallinan. Feature extraction from faces using deformable templates. In *Proc. of the IEEE*, 1990.

APPENDIX A: Mouth parameters extraction

We present in this appendix the details of the algorithm allowing us to evaluate in a robust way the height and the width of the mouth characterized by a code vector.

Each code vector is the average of the examples in the calibration base which are associated to him. For this reason the image which represented by the code vector is fuzzy. Consequently it is difficult to evaluate on the code vector itself the height and the width of the mouth. We thus evaluate these parameters on the images set associated with the code vector and we make a simple average of these parameters to characterize the height and the width of the mouth contained in the code vector.

The images are coded in RGB and transformed in Y (brightness), Rn and Gn (knowing that $Rn = \frac{R}{R+G+B}$ and $Gn = \frac{G}{R+G+B}$). For each one among it we will detect the vertical coordinate of the mouth, then the corners and finally the position of the upper and lower lips (see Fig. 7).

Mouth horizontal coordinate detection

On the image coded in Y, we evaluate the number of the line which crosses the center of the mouth in accordance with the algorithm developed in [Baig99a]. An accumulation of the pixels values is made along the lines. The signal which results from it is a function of the line number. Taking into account the fact that the mouth interior, even when it is closed, is darker than the lips or the skin, the minimal value of this signal indicates the number of the line which crosses the center of the mouth (l_{mouth}, see Fig. 7).

Mouth width detection

On the basis of the idea suggested in [Delma00], we define a region of interest of the image width and 10 pixels high centered on l_{mouth} previously detected. In each column C_j of this window we evaluate the pixel of minimal value. We propose here to evaluate a threshold (equal to the average of the minima on all the columns) to which is compared each minimum. We preserve then only the minima which have a lower value than this threshold. If a local minimum after this thresholding is isolated (the minima of the connected columns were higher than the threshold), it is removed. It remains a certain number of minima: the one which is on the left column of the window gives us the position of the mouth left corner, the one which is on the right column, the mouth right corner.

Mouth height detection

The evaluation of the mouth height cannot be done on the gray level images because the teeth and the tongue have sometimes a higher brightness value than the lips. We thus propose to filter the images in order to emphasize the lips. Inspired from [Férau01] we produce a color filter adapted to the lips colour which allows us to binarize the image coded in Rn, Gn. The filter coefficients are based on the evaluation of the lips colour in a window of 20 rows and ten columns. This window is extracted in an automatic way on the first image of the sequence; it is centered on l_{mouth} and on the central column calculated starting from the two corners of the mouth. Knowing that the mouth is systematically closed at the beginning of each sequence, we thus postulate that this window is centered on closed lips. The upper and lower lips are characterized now by very weak indices of brightness (Fig. 7).

To detect the height of the mouth, we apply the same type of algorithm as previously when it was necessary to detect the mouth center, but on a window centered on l_{mouth} and whose left and right edges are defined by the left and right corners of the mouth previously detected. An accumulation of the pixels values of this window is carried out along the lines. The resulting signal is thus a function of the line number. One compares the signal values to a threshold equal to half of the maximum value of the signal. That leads to the binary signal shown in the figure 7. It is then simple to locate the line numbers corresponding to the lower and higher edge of the mouth.

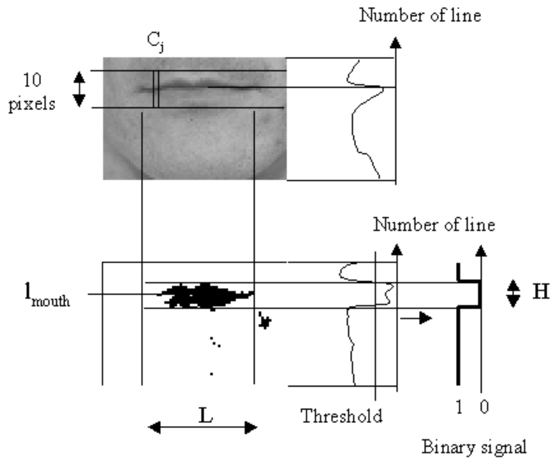

Figure 7: Extraction of the height (H) and the width (L) of the mouth.