Kohonen network, automatic disorders speech recognition,
waveblaster, cwt, continuous wavelet transform,
bark scale, speech prolongations

Ireneusz CODELLO[1], Wiesława KUNISZYK-JÓŹKOWIAK, Elżbieta SMOŁKA, Adam KOBUS

AUTOMATIC PROLONGATION RECOGNITION IN DISORDERED SPEECH USING CWT AND KOHONEN NETWORK

Automatic disorder recognition in speech can be very helpful for the therapist while monitoring therapy progress of the patients with disordered speech. In this article we focus on prolongations. We analyze the signal using Continuous Wavelet Transform with 18 bark scales, we divide the result into vectors (using windowing) and then we pass such vectors into Kohonen network. Quite large search analysis was performed (5 variables were checked) during which, recognition above 90% was achieved. All the analysis was performed and the results were obtained using the authors' program – "WaveBlaster". It is very important that the recognition ratio above 90% was obtained by a fully automatic algorithm (without a teacher) from the continuous speech. The presented problem is part of our research aimed at creating an automatic prolongation recognition system.

1. INTRODUCTION

Speech recognition is a very important branch of informatics nowadays – oral communication with a computer can be helpful in real-time document writing, language translating or simply in using a computer. Therefore the issue has been analyzed for many years by researches, which caused many algorithms to be created such as Fourier transform, Linear Prediction, spectral analysis. Disorder recognition in speech is quite a similar issue – we try to find where speech is not fluent instead of trying to understand the speech, therefore the same algorithms can be used. Automatically generated statistics of disorders can be used as a support for therapists in their attempts at an estimation of the therapy progress.

We have decided to use a relatively new algorithm – Continuous Wavelet Transform (CWT) [1, 3, 11], because by using it we can choose scales (frequencies) which are most suitable for us (Fourier transform and Linear Prediction [7, 9] are not so flexible). We have chosen the bark scales set, which is, besides the Mel scales and the ERB scales, considered as a perceptually based approach [12]. The CWT result is divided into fixed-length windows, each one is converted into a vector. The vectors, using another window are grouped, marked if this group starts with a sound repetition or not and passed onto the Kohonen network which receives 3D data and produces 2D data. On such a modified signal (Kohonen countour) we are searching for the prolongations.

Quite large recognition statistics was created obtaining very high recognition ratios.

Most of the theoretical aspects of this work are exactly the same as in our previous article [5], because in both cases we describe smaller parts of the one, bigger project. Therefore in chapters 2 and 3 we place only brief description of this theory (more details are in our previous article [5]).

[1] Institute of Computer Science, Maria Curie-Skłodowska University, Marii Curie-Skłodowskiej 1, Lublin, Poland.
first author's email: irek.codello@gmail.com.

2. CWT

2.1. MOTHER WAVELET

Mother wavelet is the heart of the Continuous Wavelet Transform:

$$CWT_{a,b} = \sum_t x(t) \cdot \psi_{a,b}(t) \text{, where } \psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right) \quad (1)$$

where $x(t)$ – input signal, $\psi_{a,b}(t)$ – wavelet family, $\psi(t)$ – mother wavelet, a – scale (multiplicity of mother wavelet), b – offset in time. We used Morlet wavelet of the form [7]:

$$\psi(t) = e^{-t^2/2} \cdot \cos(2\pi \cdot 20 \cdot t) \quad (2)$$

which has center frequency F_C=20Hz.

2.2. SCALES

For frequencies of scales we decided to use Hartmut scales [15]:

$$B = \frac{26.81}{1+1960/f} - 0.53 \text{, } f \text{ – freq. in Hz} \quad (3)$$

and the frequency of each wavelet scale a was computed from the equation

$$F_a = F_C F_S / a \text{, } F_S \text{ – sampling frequency} \quad (4)$$

During the research we decided to remove 4 scales as insignificant in the recognition process (marked as crossed), therefore eventually only 18 scales were used.

Table 1. 22 scales a (and scale's shift b) with corresponding frequencies f and bark scales B.
By removing crossed scales we increased recognition ratio.

***a* [scale]**	***f* [Hz]**	***B* [bark]**		***a* [scale]**	***f* [Hz]**	***B* [bark]**
~~46~~	~~9586~~	~~21,7~~		297	1484	11
57	7736	20,9		347	1270	10
68	6485	20,1		408	1080	9
83	5313	19,1		479	920	8
100	4410	18		572	770	7
119	3705	17		700	630	6
140	3150	16		864	510	5
163	2705	15		1102	400	4
190	2321	14		~~1470~~	~~300~~	~~3~~
220	2004	13		~~2205~~	~~200~~	~~2~~
256	1722	12		~~4410~~	~~100~~	~~0,8~~

After $CWT_{a,b}$ is calculated, we find it more useful to:

- calculate its module |CWTa,b|
- smooth it out (see Figure 1)

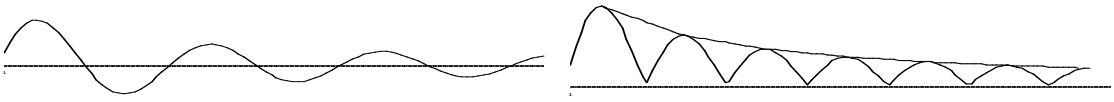

Fig. 1. Left: Cross-section of one *CWTa*,b scale. Right: Cross-section of one |*CWTa,b*| scale and its contour (smoothed version).

- divide it into windows: we cut spectrogram, consisting of 18 smoothed bark scales vectors, into 23.2ms frames (512 samples when FS=22050Hz), with a 100% frame offset. Because every scale has its own offset – one window of fixed width (e.g. 512 samples) will contain different number of amplitudes (CWT similarity coefficients) in each scale (see Figure 2), therefore we take the arithmetic mean of each scale's amplitudes.

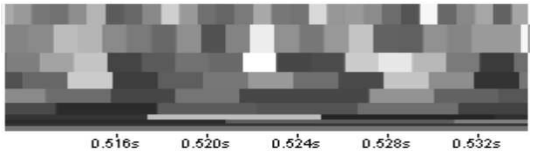

Fig. 2. One CWT window (512 samples when F_S=*22050Hz*).

From one window we obtain the vector *V* of the form presented in eq. 5. Such consecutive vectors are then passed into the Kohonen network.

$$\vec{V} = \{mean(|CWT_{57}|), mean(|CWT_{68}|), ..., mean(|CWT_{864}|), mean(|CWT_{1102}|)\} \quad (5)$$

3. KOHONEN NETWORK

We also use the Kohonen network ([10], [6]) (or "self-organizing map", or SOM, for short) with a standard WTM (winner takes most) learning algorithm and Euclidean metric. As a result of such learning, we can say that, in a Kohonen map, neurons located physically next to each other will correspond to classes of input vectors that are likewise next to each other (Figure 4). Therefore such regions are called maps.

We number the Kohonen neurons by rows from the top to the bottom so that we could present them in 2D form

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14

For every 2D CWT vector (see eq. 5) we obtain one winning neuron. Therefore we use the Kohonen network to convert 3-dimention CWT spectrogram (which consists of 2D CWT vectors laying one next to other) into 2-dimentional winning neuron contour ([13], [14]).

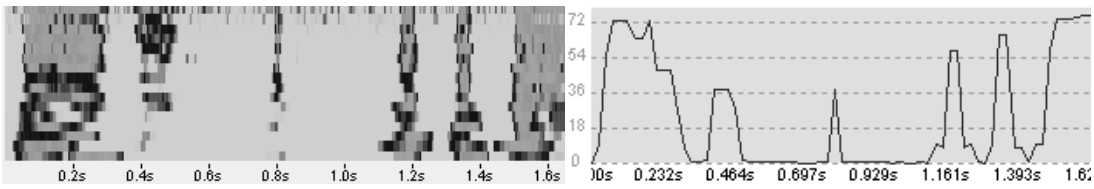

Fig. 3. Converting 3D CWT (Left picture. *Y* axis: the bark scale, *X* axis: the time) into 2D Kohonen winning neuron contour (Right picture. *Y* axis: winning neuron, *X* axis: the time).

3.1. LEARNING ALGORITHM MODIFICATION

We changed a little bit the learning algorithm. To make it to give more stable contours (Fig 4.) i.e. every time the same, no matter how the network was initiated, we set 0th neuron weights with zeros and mark them as read-only. They take part in all computations but when it comes to weights changing – we do not allow it. Therefore 0-th neuron always pulls silence (which is always the weakest signal) to the top-left corner, then top-left corner (with neighbors) gathers weak signal, therefore strong signal is naturally placed in bottom-right corner.

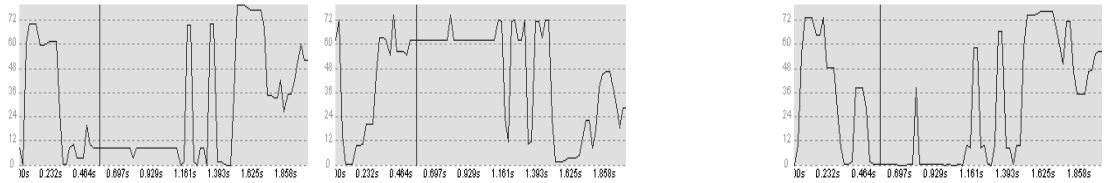

Fig. 4. Different Kohonen winning neuron contours. Right-most result is obtained using modified algorithm therefore silence is always placed in or near 0-th neuron.

We also added additional step into the learning process [4] which was not used in our previous research [5]. This step is applied after the network has been trained using the standard algorithm described previously. The purpose is to reduce each map (which contains similar neurons) to only one neuron within one map.

We do the following:

- Find two closest neurons k_A, k_B (the distance between neurons weights are measured using Euclidean metric)
- If the distance is less than some threshold (algorithm's parameter), fill weights of one of the neurons with zeros. This way input vectors that were assigned to k_B neuron, now will be assigned to k_A
- Repeat steps 1. and 2. until there exists a pair of neurons closer than the threshold.

The result of the reducing procedure is shown in Figures 5 and 6. As we can see, such a result is much clearer and therefore more useful than an unmodified result.

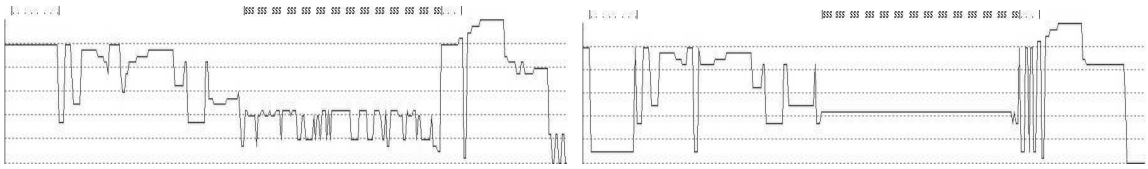

Fig. 5. Winning neuron contour of the 2 seconds long utterance with prolongation "sss" without (on the left) and with (on the right) 'neuron reduction'. Kohonen network size: 5x5. Vertical axis: winning neuron number, horizontal axis: time.
Screenshot from program "WaveBlaster".

The algorithm treats the silence as the prolongation as well. Because we couldn't find clear and easy way to distinguish silence prolongations from utterance prolongation on the winning neuron contour (statistics were showing many algorithm mistakes), we decided to use simple utterance-finding algorithm and search for the prolongations only in utterance fragments.

4. AUTOMATIC DISORDERED SOUND REPETITIONS RECOGNITION

4.1. INPUT DATA

We took Polish disordered speech recordings of 6 persons and Polish fluent speech recordings of 4 persons. In the disordered speech recordings we chose all prolongations with

4-second surroundings and from fluent speech we randomly chose several 4-second long sections. We merged all the pieces together obtaining 18 min 32 s long recording containing 373 prolongations. The statistics are the following:

Table 2. Disordered sound prolongation counts.

a	*e*	*f*	*g*	*h*	*i*	*j*	*m*	*n*	*o*	*s*	*sz*	*ś*	*u*	*w*	*y*	*z*	*ź*	*ż*	***All***
4	10	11	1	8	12	13	17	29	16	65	15	39	6	34	26	46	6	15	373

4.2. ALGORITHM

The procedure of finding prolongations in the file was the following:

- Compute CWT spectrogram for the entire file
- Divide the spectrogram into ‘small’ windows (we used 23.2 ms) with a certain offset (we used 23.2 ms). By using windowing (see section 2 for details) each ‘small’ window is converted into a set of 18 element vectors (each element of a vector corresponds to one bark scale). A vector is marked as silence if all its values are less than -55dB (where 0dB is the maximum value).
- Find words (that is a sequence of non-silence vectors). Only words longer than min ProlongWidth parameter were taken.
- For some tests we use additional parameter wordLength – it means how long a fragment should be. If a word was less than wordLength it was cut with wordLength length anyway and if it was longer – it was divided into windows of wordLength size (with offset equaling 300ms)
- Each word which consists of 18-element vectors was passed into the Kohonen network. After the learning process (with ‘neuron reduction’) we obtained a winning neuron graph (Figure 5)
- If a winning neuron contour (Figure 5) contained a section longer than minProlongWidth parameter, in which only one neuron wins, then this section was considered as prolongation.
- If the section (marked as a prolongation) overlapped some other automatically found prolongation, then the sections were combined creating one long prolongation.
- Finally, we manually compared the pattern with the algorithm output and counted the number of correctly and incorrectly recognized prolongations.
- The recognition ratio was calculated by using the formulas [2]:

$$sensitivity = \frac{P}{A};\ predictability = \frac{P}{P+B} \tag{6}$$

where P is the number of correctly recognized disorders, A is the number of all disorders and B is the number of fluent sections mistakenly recognized as disorders.

5. RESULTS

We wanted to test the following variables:

- minProlongWidth (mpw) – in milliseconds (described in 4.2),
- wordLength (wl) – in milliseconds (described in 4.2), for some tests this parameter was not set, which meant that words had variable length,

- Kohonen's network size (ks) – AxB, where A is the number of rows, B is the number of columns,
- Kohonen's neighbour values (kn) – A_B, where A is a starting neighbor factor, B is an ending neighbor factor. Kohonen network is learning for 100 epochs and the neighboring factor is linearly decreasing (learning factor changed from 0.2 to 0.1 linearly),
- Kohonen's 'reducing neuron' distance (kd) – (described in 3).

The results are presented in the Table 3.

Table 3. Automatic disordered sound prolongation recognition results [in %].
S – sensibility and P – predictability, kd – Kohonen 'reducing neuron' distance, wl – wordLength, ks – Kohonen network size, kn – Kohonen neighbor values, mpw – minProlongWidth.

	kd=0.30		kd=0.35		kd=0.40		kd=0.45		kd=0.50		kd=0.55	
	S	P	S	P	S	P	S	P	S	P	S	P
series 1												
wl=not_set ks=**3x3** kn=2.5_**0.5** mpw=250	63	94	68	94	71	95	74	93	78	90	81	87
wl=not_set ks=**3x3** kn=2.5_**1.0** mpw=250	74	80	76	76	80	76	82	73	85	69	86	67
wl=not_set ks=**4x4** kn=2.5_**0.5** mpw=250	49	94	52	94	60	94	63	94	68	93	75	92
wl=not_set ks=**4x4** kn=2.5_**1.0** mpw=250	62	92	65	89	71	88	76	84	78	81	84	81
series 2												
wl=not_set ks=**3x3** kn=2.5_**0.5** mpw=200	75	76	81	77	82	77	84	76	87	74	90	70
wl=not_set ks=**3x3** kn=2.5_**1.0** mpw=200	77	80	78	76	81	72	84	70	87	67	90	51
wl=not_set ks=**4x4** kn=2.5_**0.5** mpw=200	61	87	66	87	72	86	76	86	81	82	84	80
wl=not_set ks=**4x4** kn=2.5_**1.0** mpw=200	76	78	80	76	83	73	84	70	86	67	89	63
series 3												
wl=not_set ks=**5x5** kn=2.5_**0.5** mpw=250	37	95	43	95	52	98	57	96	62	96	68	95
series 4												
wl=1500 ks=**3x3** kn=2.5_**0.5** mpw=250	78	83	81	84	82	83	85	82	86	80	87	78
wl=1500 ks=**3x3** kn=2.5_**1.0** mpw=250	84	72	80	68	81	64	90	64	91	61	93	59
wl=1500 ks=**4x4** kn=2.5_**0.5** mpw=250	62	89	63	90	72	90	80	89	83	88	84	86
wl=1500 ks=**4x4** kn=2.5_**1.0** mpw=250	77	86	80	84	85	82	86	78	90	77	91	74
series 5												
wl=1500 ks=**3x3** kn=2.5_**0.5** mpw=200	88	66	91	68	92	66	92	64	93	62	94	60
wl=1500 ks=**3x3** kn=2.5_**1.0** mpw=200	90	55	92	52	93	50	94	47	96	45	96	43
wl=1500 ks=**4x4** kn=2.5_**0.5** mpw=200	77	81	81	80	87	80	89	77	92	73	94	73
wl=1500 ks=**4x4** kn=2.5_**1.0** mpw=200	89	72	90	66	93	65	95	63	96	59	97	56
series 6												
wl=1500 ks=**5x5** kn=2.5_**0.5** mpw=250	49	92	63	93	65	93	66	91	74	91	77	88
series 7												
wl=**1000** ks=4x4 kn=2.5_0.5 mpw=250	51	93	59	92	62	92	69	91	73	90	80	88
wl=**1500** ks=4x4 kn=2.5_0.5 mpw=250	62	89	63	90	72	90	80	89	83	88	84	86
wl=**2000** ks=4x4 kn=2.5_0.5 mpw=250	67	91	71	91	75	89	78	87	80	86	85	85
wl=**2500** ks=4x4 kn=2.5_0.5 mpw=250	73	91	77	88	78	88	81	87	82	85	**92**	**82**
wl=**3000** ks=4x4 kn=2.5_0.5 mpw=250	76	89	78	87	81	88	83	86	84	84	87	84
wl=**3500** ks=4x4 kn=2.5_0.5 mpw=250	73	88	77	88	77	88	82	86	82	82	87	79
series 8												
wl=**2500** ks=5x5 kn=2.5_0.5 mpw=250	60	92	65	91	68	90	75	89	79	88	82	86
wl=**3000** ks=5x5 kn=2.5_0.5 mpw=250	65	93	70	92	74	91	75	89	79	89	83	86
wl=**3500** ks=5x5 kn=2.5_0.5 mpw=250	69	93	72	91	72	90	77	89	82	87	82	83

6. CONCLUSIONS

In first two series of tests we wanted to check the impact of:

- minimal prolongation width (mpw) parameter and Kohonen network size. Two values were checked 200ms and 250ms because most of the prolongations are longer than 250ms, but there are some shorter ones (the shortest has 226ms length),
- Kohonen network size (ks) – two values were checked 3x3 (9 neurons) and 4x4 (16 neurons). Larger or smaller nets were ignored because neurons count is not proportional to a number of phonemes in a word.
- Kohonen neighbor factor (kn) – two values were checked 2.5_1.0 and 2.5_0.5. First set does not narrow its neighboring into a single neuron so its learning is more general, while the second set has more sharpening ending value.

All tests were performed for Kohonen reduction (kd) changing from 0.30 to 0.55.

In most cases mpw=250ms gave better results – it gave a little worse sensibility (so it found less prolongations - which was obvious as the length condition is more demanding), but it gave much better predictability (the algorithm made less mistakes). In all cases kn=2.5_0.5 gave better results then the same configuration but with kn=2.5_1.0. Network size (ks=3x3/4x4) did not give significant differences.

Just in case we did the 3rd series of tests for ks=5x5, but it gave worse results.

Series 4,5 and 6 corresponded to series 1, 2 and 3, but with different word cutting – wl parameter was set to 1500ms. All results were equal or better then the same set of parameters but with wl not set. We can see here also that ks=4x4 gave better results than ks=3x3.

So as a conclusion from six series of test we can see that the best results are for ks=4x4, kn=2.5_0.5 and mpw=250 ms.

As the last parameter we checked the wl – which we numbered as series 7. Because for longer words number of phonems increases, just in case, we checked the bigger Kononen net too (series 8) to have number of neurons corresponding to number of phonems in the word but like in series 3 and 6, net size 5x5 gave worse ratios.

Series 7 gave us the best result S=92%, P=82% which we find a very good ratio. Predictability could be higher (algorithm could make less mistakes) but we need to remember that this is recognition in the continuous speech, therefore number of fluent words is disproportionately higher than disordered fragments). The similar recognition ratio (91%) was achieved by our research group using FFT and fuzzy logic [16] but the research was performed on the manually cut fragments. Our test was done on the continuous speech which we find to be more difficult.

For every series, all kd values were checked. As we can see, increasing this parameter causes increasing of sensibility but decreasing of predictability. Higher values were not checked because, in most cases, the decrying of predictability was equal or higher than increasing of sensibility.

All the results are leading us to the final conclusion, that wl=2500 ks=4x4 kn=2.5_0.5 mpw=250 kd=0.55 is the best configuration from the searched space of parameters.

BIBLIOGRAPHY

[1] AKANSU A.N, HADDAD R.A., Multiresolution signal decomposition, Academic Press, 2001.

[2] BARRO S., Marin R., Fuzzy Logic in Medicine, Physica-Verlag Heidenberg, New York, 2002.

[3] CODELLO I., KUNISZYK-JÓŹKOWIAK W., Wavelet analysis of speech signal, Annales UMCS Informatica, 2007, AI 6, pp. 103-115.

[4] CODELLO I., KUNISZYK-JÓŹKOWIAK W., KOBUS A., Kohonen network application in speech analysis algorithm, Annales UMCS Informatica, 2010, (Accepted paper).

[5] CODELLO I., KUNISZYK-JÓŹKOWIAK W, SMOŁKA E., KOBUS A., Disordered sound repetition recognition in continuous speech using CWT and Kohonen network, Journal Of Medical Informatics & Technologies, 2011, Vol. 17, pp. 123-130.

[6] GARFIELD, S., ELSHAW M., WERMTER S., Self-orgazizing networks for classification learning from normal and aphasic speech, In The 23rd Conference of the Cognitive Science Society, Edinburgh, Scotland, 2001.

[7] GOLD, B., MORGAN, N., Speech and audio signal processing, JOHN WILEY & SONS Inc, 2000.

[8] GOUPILLAUD P., GROSSMANN A., MORLET J., Cycle-octave and related transforms in seismic signal analysis", Geoexploration, 1984-1985, Vol. 23, pp. 85-102.

[9] HUANG, X., ACERO, A., Spoken Language Processing: A Guide to Theory, Algorithm and System Development, Prentice-Hall Inc., 2001.

[10] KOHONEN, T., Self-Organizing Maps, 34:p.2173-2179, 2001.

[11] NAYAK J., BHAT P.S., ACHARYA R., AITHAL U.V., Classification and analysis of speech abnormalities, Elsevier SAS, 2005, Vol. 26, No. 5-6, pp. 319-327.

[12] SMITH J., ABEL J., Bark and ERB Bilinear Transforms, IEEE Transactions on Speech and Audio Processing, November, 1999.

[13] SZCZUROWSKA I., KUNISZYK-JÓŹKOWIAK W., SMOŁKA E., The application of Kohonen and Multilayer Perceptron network in the speech nonfluency analysis, Archives of Acoustics. 2006, Vol. 31 (4 (Supplement)): pp. 205-210.

[14] SZCZUROWSKA, I, KUNISZYK-JÓŹKOWIAK W., E. SMOŁKA, Application of Artificial Neural Networks In Speech Nonfluency Recognition, Polish Jurnal of Environmental Studies, 2007, Vol. 16, No. 4A, pp. 335-338.

[15] TRAUNMÜLLER H., Analytical expressions for the tonotopic sensory scale, J. Acoust. Soc. Am., 1990, Vol. 88, pp. 97-100.

[16] SUSZYŃSKI W., KUNISZYK-JÓŹKOWIAK W., SMOŁKA E., DZIEŃKOWSKI M., Prolongation detection with application of fuzzy logic, Annales Informatica Universitatis Mariae Curie-Skłodowska, 2003, pp. 133-140.