Ternary Coded Melody as Blind Audio Watermark

Rustam Kh. Latypov, *Senior Member, IEEE* and Evgeni L. Stolov

***Abstract* — In this paper, we developed a new technique for blind embedding of ternary coded watermarks into audio files. Usage of ternary coding increases payload of the method that can be considered as an advantage against binary-coded watermarks. A well-known melody is presented as a sequence of ternary digits (trits) and is used as a watermark. This sequence is embedded into the time domain of a host audio file through amplitude modulation and B-splines. There is a version of that procedure where the clean copy of the container is necessary for extraction watermark [1]. In our approach, we exclude that container and convert the method into a blind one. The strong correlation between neighbor samples in the container is used to this end. A procedure based on neuron net is suggested for enhancement perception of ternary coded music. In this case, we exploit the correlation between samples in the watermark melody. It is supposed that a person checks the mark's existence, and he/she can recognize the melody even after significant distortions. The resistance of the technique to the most successful attacks is investigated. The paper is an extended version of the conference paper [1].**

***Keywords* — audio watermarks, B-spline, blind embedding, neural net, ternary coded melody.**

I. Introduction

DIGITAL technology provides new possibilities for authors of songs and other audio files. The software has been developed that helps inexperienced musicians in the creation of products having popularity. The composers, having no brand, put on the results of their works into open access on the Internet. Anyone can declare another person's work as his/her own. The simplest method for proving the authorship of the product is embedding a watermark into the audio file [2].

The Web version of the composition is thought of as advertising. So a small decrease in the quality of perception of the product is admissible. That is an essential difference between the watermark technique and steganography, where all perceptible for human distortions are undesirable.

Paper received May 07, 2020; revised July 03, 2020; accepted July 04, 2020. Date of publication July 31, 2020. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Irini Reljin.

This paper is revised and expanded version of the paper presented at the 27th Telecommunications Forum TELFOR 2019 [1].

The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University. The Russian Science Foundation grant No19-18- 00202 also supports this work.

Corresponding author: Rustam Kh. Latypov is with the Kazan Federal University, Kazan, Kremliovskaya 18, 420008, Russia, (e-mail: Roustam.Latypov@kpfu.ru).

Evgeni L. Stolov is with the Kazan Federal University, Kazan, Kremliovskaya 18, 420008, (e-mail: ystolov@list.ru).

Many methods for implementation of watermark techniques were suggested during the last 30 years. A review of them can be found in [3]. There are many approaches to the classification of all existing methods. For our purposes, we mention two of them: classification by the area where the watermark is inserted and classification by using or not using the source host file to extract the watermark (blind or non-blind watermarking) [3]. The first implementation of the audio watermark was in the time domain [2]. The method did not require serious resources, so it is used very intensively. However, a drawback of that technique is the low payload that led to the situation that new methods based on various transforms were suggested [4], [5], [6]. Nevertheless, the realization of those methods demands significant calculations.

Nevertheless, in this paper, we leverage the time domain for insertion watermarks. We implement ternary coded watermarks, which made possible to increase the payload of the procedure.

The principal idea of the methods is as follows. The watermark is converted into a sequence of balanced trits belonging to the set $\{-1,0,1\}$. The host file is divided into a series of non-overlapping intervals. Each interval is modified following the current part of the watermark sequence, which is inserted into the host. The algorithm for the transformation of a fragment into a new one is given in [1]. It depends on non-zero trits that must be embedded into the fragment. The transformation is a kind of amplitude modulation that is performed through B-splines [7]. We developed a new version of that algorithm that provides the possibility for extraction of trit from the fragment without knowledge of the clean host. A method for enhancement perception of the ternary version of the audio file is suggested. The ternary sequence is converted into a new sequence consisting of ternary vectors of a prescribed length. A neuron net is trained for transforming each vector into a sample of the original audio file. Usage of the net allows enhancing the sound of the extracted watermark.

All calculations and graphics are performed employing Python packages [8], [9], [10, [11].

II. Ternary Coded Melody

The method converting the source melody, used as a watermark, into a ternary sequence was presented in [1]. Here we recall the necessary steps of the procedure. Let $W =< s_0, s_1, \dots, s_{M-1} >$ be a fragment of the source file intended for watermarking. Let us create the ternary sequence:

$$ter_n = \begin{cases} 0 & if\ |s_n| < K \\ sign(s_n) & otherwise \end{cases} \qquad (1)$$

The result of the transformation of *W*, according to (1), is the new sequence $Appr =< ter_0, ter_1, \dots, ter_{M-1} >$. In the ideal case, the parameter *K* is chosen in the way of providing an optimal approximation of *W* by *Appr*. The criterion of the optimization on the base of the signal to noise ratio (*SNR*) is

$$SNR = 10 \cdot log_{10}\left(\frac{\sigma^2(norm(W)}{\sigma^2(norm(W)-norm(Appr))}\right) \to max. \quad (2)$$

Here, the normalized version of array $norm(W) = \frac{W}{std(W)}$, and $std(W)$ is the standard deviation of the array. A straightforward algorithm providing a suboptimal value for the *K* in a sense (2) is presented in [1]. It is based on the tabulation of *K* with a prescribed step.

It was mentioned above that any fragment of the host file brings the information about a single trit of the watermark, but the interval cannot be short. It means that the payload of the method is rather small. The watermark melody is written in a high sampling frequency. Usually, it is 44100Hz. On the other hand, a human can recognize the melody even if it is written with a significantly lower sampling frequency. The lower the frequency used by the record of the watermark melody, the more payload of the method. The question arises, how the frequency influences the criterion (2). The results of the experiment with two fragments of a song are presented in Table 1. One can see that the sampling frequency of 5512 Hz provides an acceptable *SNR* value, and humans can recognize the melody very easily. This frequency is used in our following experiments.

III. B-splines and Amplitude Modulation

Let $Fragm =< h_0, h_1, \dots, h_{L-1} >$ be a fragment of the host file. Only a single trit is embedded into *Fragm*. If the current value of the trit is zero, then the fragment stays unchanged; otherwise, the fragment is modulated utilizing a B-spline.

Table 1: Dependence of SNR on Sampling Frequency.

Frequency (Hz)	***SNR(dB)***	***SNR(dB)***
44100	7.51	8.21
22050	7.50	8.2
11025	7.48	8.10
5512	7,45	8.18

Recall the definition of B-spline [7]. We exploit B-spline $B_2(t)$:

$$\mathrm{B_2(t)} = \begin{cases} \frac{t^2}{2}, & 0 \le t < 1, \\ -t^2 + 3t - 1.5, & 1 \le t < 2, \\ \frac{(3-t)^2}{2}, & 2 \le t < 3, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

The graph of the function has the form, as in Fig. 1.

In what follows, we omit the index in the designation of the spline. Let *ter* be a non-zero item of watermark that must be inserted into the fragment *Fragm* of length *L* of the host. We create the sequence $Bs =< b_0, b_1, \dots, b_{L-1} >$ where $b_i = B\left(\frac{i}{L-1}\right)$.

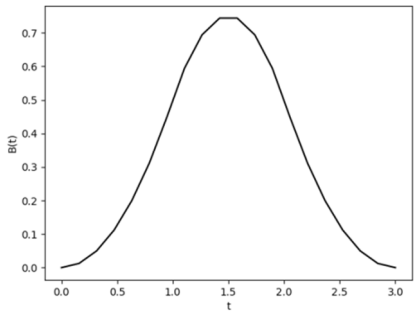


Fig. 1. Graph of $B_2(t)$ spline.

Recall the two methods described in [1] for embedding the trit *ter:*

$$\mathrm{Fragm}' = \mathrm{Fragm} + \mathrm{Q_1} \cdot \mathrm{ter} \cdot \mathrm{Fragm}*\mathrm{Bs}, \quad (4)$$

$$\mathrm{Fragm}'' = \mathrm{Fragm} + \mathrm{Q_2} \cdot \mathrm{ter} \cdot \mathrm{sign(Fragm)}*\mathrm{Bs}. \quad (5)$$

Here $A * B$ denotes the elementwise product of two arrays of the same lengths. The positive coefficients Q_1, Q_2 influence the transparency of the modification and the resistance of the procedure to various attacks. Below, we denote the first method as a multiplicative procedure and the second – as an additive procedure and use MUL and ADD abbreviations.

It is shown in [1] that distortion of the host signals after embedding of non-zero trit in terms of *SNR* is

$$\mathrm{SNR} \approx -20 \cdot \log_{10}(\mathrm{Q_1}) \quad (3)$$

for MUL and

$$\mathrm{SNR} \approx 10 \cdot \log_{10}\left(\frac{60 \cdot \sigma^2(\mathrm{Fragm})}{11 \cdot \mathrm{Q_2^2}}\right) \quad (4)$$

for ADD.

Notice that $Q_1 < 1$ in (6), but Q_2 is chosen so that the *SNR* value is comparable with one defined by (6).

A. Extraction watermark provided access to clean host file

In our approach, if $ter \neq 0$, then the choice of the coefficients Q_1, Q_2 must ensure coincidence of signs of items having equal indices in the original and in modified versions of fragments. This condition imposes restrictions on the upper bounds of the coefficients. If it is fulfilled and *ter* = 1 then

$$\sum |Fragm'[i]| > \sum |Fragm[i]|,$$

and if $ter = -1$ then

$$\sum |Fragm'[i]| < \sum |Fragm[i]|.$$

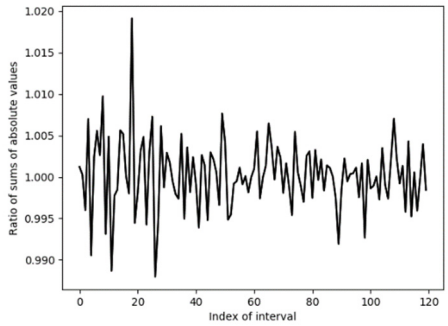
The analogous inequalities hold for $Fragm''$. That leads to a simple method for extraction of ter from the modified fragment: we have to compare the sums of absolute values of items in modified and original fragments. All the problems, which one faces during the realization of that idea are discussed in [1]. The evident drawback of that approach is the necessary utilization of the clean host file. Here we present a trick to convert the previous method into a blind one.

B. Embedding and extraction watermark without access to the clean host file

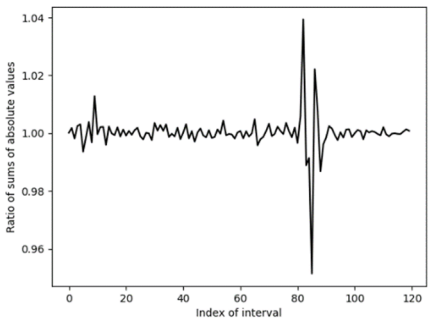
Since the host file is supposed to be an audio file, a strong correlation reveals among neighbor samples. If *Fragm* is intended for embedding a non-zero trit, we change just a half of *Fragm*, whereas another half is unchanged. Unchanged half is used during embedding to approximate the feature of the second half of *Fragm*. We select the sum of absolute values of items in half as a feature. To prove the possibility of the suggested approach for feature extraction, we realized Algorithm 1. An audio file used as a host is leveraged in an experiment. The results of the experiment are shown in Fig. 2.

Algorithm 1 Ratio of sums of two parts of fragment

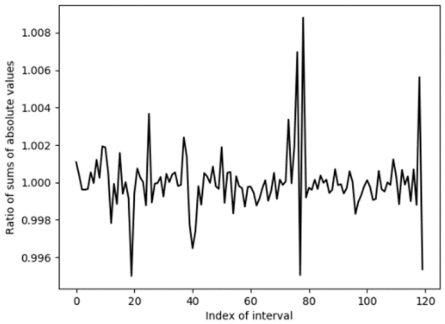
Input: $ListOfFragm$ {List of fragments}
Output: Graph of ratios
1: $AllRatios$ {Create empty list}
2: **for** $Fragm$ in $ListOfFragm$ **do**
3: $Fragm_0 \leftarrow Fragm$ {Part of $Fragm$ with even indices}
4: $Fragm_1 \leftarrow Fragm$ {Part of $Fragm$ with odd indices}
5: $S_0 \leftarrow Fragm_0$ {Sum of absolute values of items}
6: $S_1 \leftarrow Fragm_1$ {Sum of absolute values of items}
7: $AllRatios.append(S_1/S_0)$
8: **end for**
9: **return** $AllRatios$



(a)



(b)



(c)

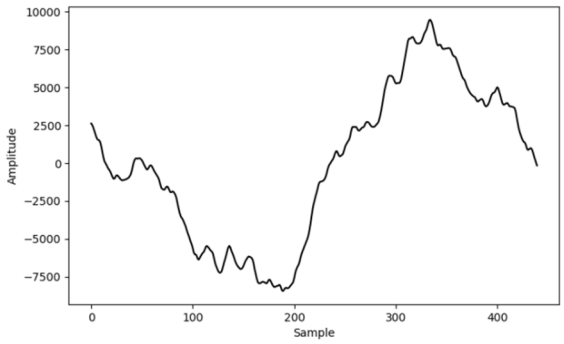
Fig. 2. The ratio of sums of absolute values of items. Length of fragments equals: (a) 420, (b) 840, (c) 1780.

One can see that the length of the fragments does not significantly influence the distribution of ratios. Algorithm 2 describes the embedding procedure.

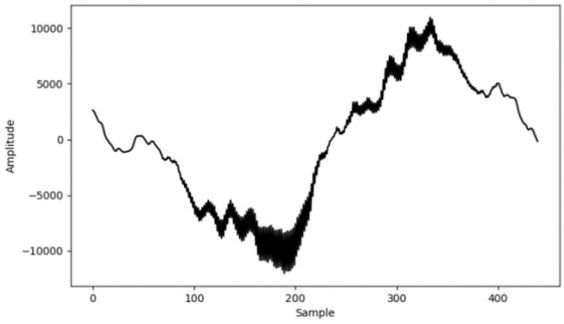
Algorithm 2 Modification fragment by non-zero trit

Input: $Fragm, ter$ {Non-zero trit ter}
Output: Modified fragment
1: $Fragm_1 \leftarrow Fragm$ {Part of $Fragm$ with odd indices}
2: $MFragm_1 \leftarrow Fragm_1, ter$ {Modification by MuL or Add method}
3: $MFragm \leftarrow Fragm, MFragm_1$ {Replace items with odd indices in $Fragm$}
4: **return** $MFragm$

The first question that must be answered is - how the embedding procedure influences the perception of the modified host file. Formally, all distortions can be described in terms of SNR (see (6) and (7)). Graphs of a fragment before and after embedding are presented in Fig. 3.



(a)



(b)

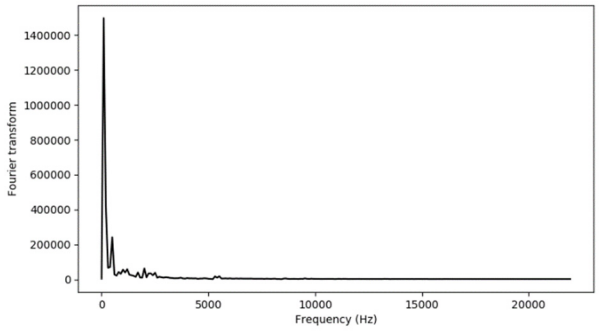
Fig. 3. Original fragment (a) and its modified version (b).

The most distorted part of the fragment is the middle area: here, alternate the original and modified values of the host fragment. One can predict that this phenomenon must reveal in the upper part of the spectra of the fragment. Fig. 4 supports that hypothesis.

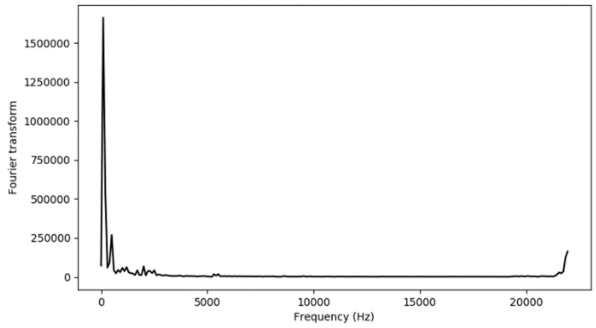
Now, consider the extraction procedure. The extraction idea is based on calculating the ratio of the sums of absolute values of items in two halves of the fragment. One can see in Figs. 2, that ratio deviates if an even fragment was not modified. It is impossible to set a bound for a ratio for distinguishing zero and non-zero trits. Algorithm 3 shows the solution to the problem we suggest. Here, we use the designation of the functions from [9].

Algorithm 3 Extraction of watermark. Multiplicative case

Input: $ListOfFragm$ {List of fragments}
Output: Watermark as a list of trits
1: $Ratios$ {Create empty list}
2: **for** $Fragm$ in $ListOfFragm$ **do**
3: $Fragm_0 \leftarrow Fragm$ {Part of $Fragm$ with even indices}
4: $Fragm_1 \leftarrow Fragm$ {Part of $Fragm$ with odd indices}
5: $S_0 \leftarrow Fragm_0$ {Sum of absolute values of items}
6: $S_1 \leftarrow Fragm_1$ {Sum of absolute values of items}
7: $Ratios.append(S_1/S_0)$
8: **end for**
9: $Centr \leftarrow kmeans(Ratios, 3)$ {Create centroids of 3 clusters}
10: $Watermark \leftarrow vq(Ratios, sort(Centr)$ {Distribute ratios among clusters}
11: **return** $Watermark$



(a)



(b)

Fig. 4. Spectra of the fragments: (a) original fragment, (b) modified fragment.

We calculate all ratios and implement the standard k-means procedure (function *kmeans*), which distributes all ratios among the three clusters. This function returns three centroids of the clusters, but the order of the centroids is random. To refer each cluster to the corresponding trits values, we implement *sort* procedure to the centroids and the function *vq* from the package that returns the numbers of the clusters according to the increasing order of the centroids.

If we leverage the additive version of the embedding, then the algorithm for extraction looks slightly different. It follows from (5) that

$$\sum|Fragm''[i] - Fragm[i]| = Q_2 \cdot ter \sum Bs\,[i],$$

thus this difference is independent of $Fragm$. The algorithm for extraction of watermark differs from Algorithm 3 just in a single line 7; instead of ratio Sm_0 to Sm_1, we have to leverage subtraction of those values.

IV. Experiment

We exploit k-means procedure for or extraction of the watermark, so we cannot be sure that all trits are recognized correctly. As usual, we use a ternary error ratio (TER) for description of the quality of recognition. That is the ratio of the number of incorrectly restored trits to the total number of trits in the watermark. This measure is useful if there is no change of positions of samples in test and output signals. If an attack is undergoing then, as a rule, such replacement presents. In this case, we leverage SNR in the frequency domain to compare two signals (8):

$$SNR_f = 10 \cdot \log_{10}\left(\frac{\sigma^2(Fnorm(W)}{\sigma^2(Fnorm(W) - Fnorm(Appr))}\right) \quad (8)$$

instead of (2). In our tests, various audio files are used as containers. All the below presented data relate to a big song recording file (44100 Hz, 16 bit, 4 min 40 sec duration) different fragments that are leveraged as containers.

A. Ideal conditions

Tables 2 and 3 show dependence on the quality of watermark recognition depending upon the parameters of embedding in ideal noiseless conditions. Here SNR denotes distortion of the container after embedding. One can see that the multiplicative version of embedding has an advantage against the additive version and provides acceptable results.

B. Resistance to attacks

Now, we present some results related to the resistance to attacks. In this case, we face a problem concerning the measure of the quality of the extracted sound. The TER used above does not fit this situation since trit displacement is possible as a result of the attack. Formally, the PEAQ technique is intended for this goal [13]. However, the mentioned procedure supposes that the sound under test is written with a sampling frequency of 48000 Hz. In our case, the sampling frequency of *W* is just 5512 Hz, and we are compelled to utilize SNR in the frequency domain, which compares the extracted signal with original, for evaluation of the quality. The result of the compression attack is presented in Table 4. The container in wav format with inserted watermark was converted into mp3 format with various bitrates. Then, that file was converted into wav format again for extraction of watermark. We use [14] to this end. The evaluations of the quality of extraction by both the methods of insertion of watermark are presented in Table 4. Although a human can recognize the known melody, the quality is awful. The additive version has a small advantage in this case.

TABLE 2: Quality of Extraction of the Watermark. Multiplicative Case.

Q_1	SNR_f (dB)	TER (%)
0.8	14.7	0.14
0.6	17.3	0.18
0.4	20.9	0.27
0.2	27.0	0.7
0.15	29.5	1.1

TABLE 3: QUALITY OF EXTRACTION OF THE WATERMARK. ADDITIVE CASE.

Q_2	SNR_f (dB)	TER (%)
1000	23.5	0.78
800	25.4	0.76
600	27.9	0.79
400	31.4	1.29
200	37.4	5.08

TABLE 4: RESISTANCE TO COMPRESSION ATTACK

Bitrate (kps)	MUL SNR_f(dB)	ADD SNR_f(dB)
705	1.4	3.1
600	1.4	3.1
400	1.3	3.1
200	1.2	3.0

In Table 5, resistance to filtering attacks is shown. The container was filtered by symmetric FIR filters created by the function *firwin* from [9]. The filtered container was processed by the programs for extracting watermark. Both methods produce files that can be perceived by a human, and the melody is recognized without problems.

TABLE 5: RESISTANCE TO FILTERING ATTACK.

Length of FIR filter	MUL SNR_f(dB)	ADD SNR_f(dB)
3	28.9	19.2
5	27.5	18.9
7	27.4	18.3
9	25.8	17.2
17	21.1	13.9

V. NEURAL NET AND ENHANCEMENT OF SOUND

Beforehand, we compared the original ternary signal with the ternary signal obtained while watermarking extraction. In our scheme, a human is a final detector that has to recognize the melody. That is the reason for the enhancement of the ternary coded melody for perception by a human being. While developing the procedure for blind extraction watermark, we used correlation among neighbor samples in the host. Here we show the way for partially solving the enhancement problem using correlation of neighbor samples in source watermark file. Let

$$\text{Appr} =< \text{ter}_0, \text{ter}_1, \dots, \text{ter}_\text{P} >$$

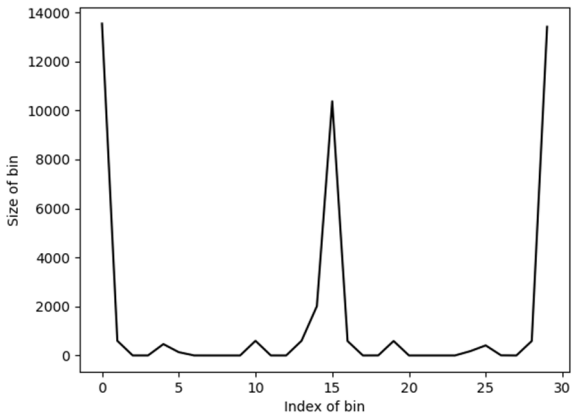
be a part of the ternary representation of the watermark *W*. Let us chose a window of length Ln that is moving (sliding) along *Appr.* There are a total of $P - Ln + 1$ possible positions of the window. Let

$$Vec =< ter_i, ter_{i+1}, \dots, ter_{i+Ln-1} >$$

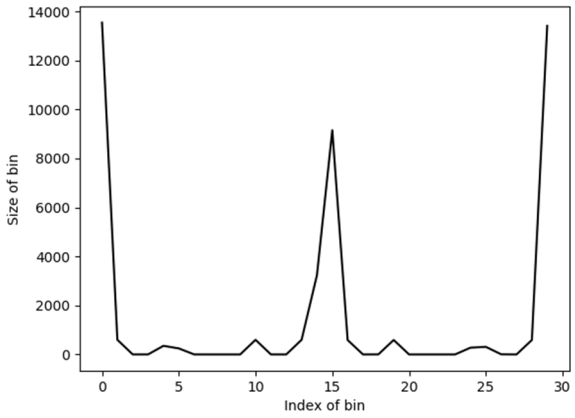
be the items inside of the current position of the window. We correspond to *Vec* an integer according to the rule

$$Int_{Vec} = \sum_{u=0}^{Ln-1} ter_{i+u} \cdot 3^u$$

Since correlation presences among neighbor samples in the audio file, the integer Int_{Vec} cannot be situated randomly. The histogram of those values corresponding to an interval of the ternary sequence is shown in Fig. 5.



(a)



(b)

Fig. 5. Histogram of Int_{Vec} related to a song. Length of window: (a) Ln = 7, (b) Ln = 11.

There are three peaks referred to vectors with constant components: $< -1, -1, \dots, -1 >$, $< 0,0, \dots, 0 >$, and $< 1,1, \dots, 1 >$. The experiments show that the structure of the histogram related to the audio file is independent of Ln in some interval. The length of that interval depends on sampling frequency and begins from $Ln = 7$. One can see that the histograms built for $Ln = 7$ and $Ln = 11$ cannot be distinguished visually. For comparison, the histogram related to white noise is depicted in Fig. 6. Here are a few bins of the same size distributed among all 30 bins exploited in the histogram. It means that the nature of an audio file is mirrored in related histograms. Due to this circumstance, we can state the following. We can rely on the sample restoring ability, corresponding to the vector middle position in the source file.

We use a neural net for performing the needed calculation. The net is realized by means of the package *TensorFlow-Keras* [11], [12]. The net has the following parameters:

- Number of layers =3.
- The first layer has 128 neurons, the input shape *Ln* and the activation function *tanh.*
- The second layer has 128 neurons and the activation function *tanh.*

- The output layer has one neuron, the activation function *tanh,* and the loss function *mse.*

The training procedure includes the steps:

- The source watermark melody W converted into the ternary sequence *Appr*.
- The watermark W transformed into the normalized array W' with values in the interval [-1,1].
- Interval of odd length Ln slides along *Appr*, each time shifting by one position. Each slot is considered as a vector *Vec* that is applied to the net inputs. The output of the net is the signal of W' corresponding to the center of *Vec.*

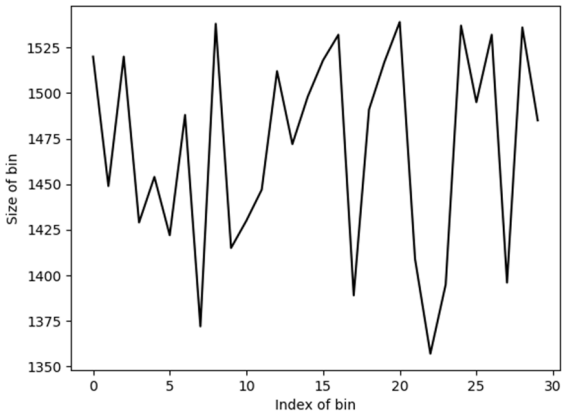


Fig. 6. Histogram of Int_{Vec} related to white noise. Length of window Ln = 11.

The results of the experiment with Ln = 11 are presented in Table 6. The values shown are obtained by averaging the results with six containers. Experiments with other containers and watermark bring identical values, which differ from the presented no more than to 10%. The SNR_f evaluates the quality of approximation of the source file by extracted watermark and by an enhanced version of the extracted watermark. Here, I denotes MUL, II – MUL enhanced, III – ADD, IV – ADD enhanced procedures.

TABLE 6: COMPARE EXTRACTED AND ENHANCED WATERMARKS.

Attack	**I.** SNR_f **(dB)**	**II.** SNR_f **(dB)**	**III.** SNR_f **(dB)**	**IV.** SNR_f **(dB)**
Compr, 600 kps	1.5	2.0	2.8	3.7
Compr, 400 kps	1.5	1.9	2.8	3.7
Compr, 200 kps	1.4	1.8	2.8	3.6
Filt, Len. 3	9.6	10.3	9.3	10.0
Filt, Len. 9	9.6	11.5	9.2	10.9
Filt, Len. 17	9.4	11.3	8.7	10.3

REFERENCES

[1] K. S. Absalyamova, R. Kh. Latypov, E. L. Stolov, "Ternary Code of Melody and Reliable Audio Watermarking," *in Proc. 27th Telecommunications Forum (TELFOR*), Belgrade, 2019.

[2] D. Kirovski and H. Malvar, "Robust spread-spectrum audio watermarking " , *in Proc Acoustics, Speech, and Signal Proc. (ICASSP'01)* , vol. 3, pp. 1345-1348, Feb. 2001.

[3] G. Hua, J. Huang, Y. Q. Shi, J. Goh, and V. L. Thing, "Twenty years of digital audio watermarking—a comprehensive review," *Signal Processing,* vol. 128, pp. 222-242, Nov. 2016.

[4] H. Sakai and M. Iwaki, "Audio Watermarking Method Based on Phase-shifting Having Robustness Against Band-Pass Filtering Attacks*", in Proc .7-th Global Conf. on Consumer Electronics (GCCE 2018)*, pp.343-346, 2018.

[5] A. Kaur and M. K. Dutta, "A Blind Watermarking Algorithm for Audio Signals in Multi-Resolution and Singular Value Decomposition*", in Proc Int. Conf . Computational Intelligence and Communication Technology, (CICT 2018),* pp. 1-5, 2018.

[6] Hwai-Hsu Hu and Tung-Tsun Lee, "High-Performance Self-Synchronous Blind Audio Watermarking in a Unified FFT Framework" , *IEEE Access*, vol. 7, pp. 19063-19076, Feb. 2019.

[7] A. Unser, "Splines. A perfect fit for signal and image processing", *IEEE Signal Proc. Magazine* , vol. 16, no. 6, pp.22-38, Nov. 1999.

[8] NumPy package manual. Available: https://numpy.org/

[9] SciPy package manual. Available: https:scipy.org

[10] Matplotlib package manual. Available: https://matplotlib.org

[11] TensorFlow package manual. Available: //www.tensorflow.org

[12] F. Chollet, *Deep Learning with Python.* Manning Publications, 2017.

[13] P. Kabal, *An Examination and Interpretation of ITU-RBS.1387: Perceptual Evaluation of Audio Quality.* Available: http://www-mmsp.ece.mcgill.ca/ Documents/Reports/2002/KabalR2002v2.pdf

[14] PyDub package manual. Available: http://pydub.com