Noise Signal Identification by Modified Self-Organizing Maps

Thomas Bryant
Department of Computer Science and Engineering
Oakland University
Rochester, MI
Email: tcbryant {at} oakland.edu

Mohamed Zohdy
Department of Electrical and Computer Engineering
Oakland University
Rochester, MI

Abstract— **This research uses a modified self-organized map to look for similarities and differences between noise signals and provides a context for unknown ones. The program first divides preprocessed data into quadrants in joint time-frequency domain and obtains the features. Features are also extracted in time domain, frequency domain, and joint time-frequency domain from noise files representing different abstract noise colors. These features are able to be input independently and classified accordingly. In addition, several types of distance metrics are tested to find best matching units including the p-norm distance formula and determining which node has the smallest weight. The output is displayed on a two-dimensional map, and the vector is colored according to the noise that has the smallest weight. A context tree allows individual features to be input and classifies the vector to the most similar noise. With the results that were gathered, the clusters have shown that distinct differences between the engine sets contribute to the study of sound discrimination by distinguishing useful sounds that humans have difficulty telling apart.**

Keywords- self-organizing feature map; sounds; unsupervised learning; classification; feature selection; watershed tranformation

I. INTRODUCTION

Unsupervised learning neural networks are characterized as competitive networks that compete on an input vector. Self-organizing maps (SOFM) are an example of unsupervised neural networks. The learning process of the self-organizing maps is competitive, meaning no teacher is needed to define the correct output for a given input [1]. The purpose here is not to find an optimal clustering of the data, but to get good insight into the cluster structure of the data for data mining purposes. Therefore, the clustering method should be fast, robust, and visually efficient. The clustering is carried out using a two-level approach, where the data set is first clustered using the SOFM, and then, the SOFM is clustered [2].

Sounds are multidimensional signals and can have practical applications in device health monitoring, containing complex features such as volume, pitch, frequency, and timbre. Humans can easily discern differences and similarities between sounds, but this paper explores how a computer program would handle this problem. By extracting suitable features from sounds, a self-organizing feature map is able to display (in a visual environment) the discriminate clustering that occurs. Sound can be mathematically represented in a time domain, a frequency domain, and a joint time-frequency domain. Each domain has its own features, but the main focus of this program dealt with the joint time-frequency domain. In the graph, time serves as the x-axis and frequency is the y-axis. However, unlike the frequency and time domains, the joint graph also displays color to represent magnitudes. A red color on the graph equates to a high magnitude, while a blue color is a low magnitude. The color adds another dimension to the data. Key features are extracted from each domain. The key features extracted from the time domain are obtained by using principal component analysis, which is discussed in more detail later. The key features extracted from the frequency domain are resonances, which are obtained by analyzing the peaks and frequencies of the respective graph. Lastly, the features extracted from the joint domain come from the quadtree function and are obtained by continually splitting the matrix into four quadrants until there is only one 2-by-2 matrix left.

In the last step of the feature extraction process, features are entered. Fourteen text boxes manipulate the data. Once the "input" is given, the top node of a context tree changes to the color of the noise it most closely resembles. This comparison is based solely on the information obtained from the input file, and does not affect the SOFM. The sound clips used in this application were relatively short, none of them lasting longer than thirty seconds. Even so, the matrix of the magnitudes from the joint time-frequency diagram was rather large; the biggest matrix was 257 rows by 1922 columns. As a result, the entire matrix could not be used efficiently when comparing against new data. Thus, a new algorithm was developed to select features for the self-organizing feature map (SOFM).

Preprocessed data is divided into quadrants and then used to obtain the salient features. Even though the number of features can change, the optimal amount ranges between 4 and 14 features. Distinct features are obtained through the three domains as discussed above. Once the features are obtained, the program learns the patterns hidden in the data. To classify the sounds, the winning node is calculated using the p-norm distance formula to determine which node is closest. The output is visualized and displayed on a two-dimensional map, and the vector is colored according to the sound that has the closest node. It visualizes sounds as similar colors that act as

agents that compete against each other. The vectors that weigh the least "win" [3].

Once the features were obtained, the map was initialized with random weights. Nodes were created to form map anchors with these pseudo-random entities. The algorithm reads the data from the input file and compares it to the random numbers, determining which pseudo-random number it is closest to using the p-norm distance formula. This stage of updating the weights to best adapt the network's behavior is known as the training phase [4]

Before the SOFM algorithm begins to cluster the nodes, it also takes in ten parameters: the neighborhood radius, learning rate, rate decay, neighborhood decay, history limit, and history factor. The default values are best for different types of data, but the user can specify any or all of the values to obtain more effective clustering. The history function in particular is an important contribution. The colors of the previous several iterations are displayed underneath the map. By knowing the previous values, the clustering becomes far more accurate [5].

Lastly, features can also be input by the user specifically. Fourteen text boxes are available for the user to manipulate the data. Once the "input" is given, the top node of a context tree changes to the color of the noise it most closely resembles. This comparison is based solely on the information obtained from the input file, and does not affect the SOFM.

II. Problem Formulation

A. *Feature Extraction Algorithm*

In order to obtain the features, quad division was used to split the data. The densities of each quadrant were calculated using the matrix density equation to determine the most active quadrant:

$$C = \sum_{i=0}^{q_{dim}} (c_i + 1) \quad if\ c_i > avg_{matrix} \qquad (1)$$

where C is equal to the count of the values in the quadrant that are greater than the average of all the values in the matrix. The variable C was then used in the following equation:

$$D = \frac{C}{n * avg_{matrix}} \qquad (2)$$

In essence, C was divided by the total count of all the values in the quadrant, which yielded the density. This process was repeated for each quadrant. The quadrants were then ordered according to highest density. The most active quadrant (the one with the highest density) was then further divided into four sub-quadrants using the same equations shown above, and again was ordered. Using the seven quadrants, the arithmetic mean and the range were calculated:

$$A := \frac{1}{n}\sum_{i=1}^{n} a_i \qquad (3)$$

The value a_i is the value of the magnitude, n is the number of values in the quadrant, and A is the sum of all the values divided by the number of values (in other words, the average). The range is simply equal to the maximum value in the quadrant minus the minimum value in the quadrant. Using the average and range of each quadrant, fourteen features were thus extracted and used as input for the SOFM.

B. *Learning Algorithm*

The map is initialized with random nodes of the weight vector. It takes an input vector and traverses each node in the map using the p-norm distance formula:

$$L_p = \left(\sum_{i=1}^{n} |q_i - p_i|^2\right)^{1/p} \quad p\ -norm\ 1,2 \ldots \infty \qquad (4)$$

This equation finds the similarities between the input vector and node weight vector. The program then tracks the node that produces the smallest distance (this node is called the best matching unit or BMU). The neighborhood function updates the BMU and its neighborhood making them more similar to the inputted vector using the equation below:

$$W_v(t+1) = W_v(t) + \alpha(t) * \beta(t) * [dist(t) - W_v(t)] \qquad (5)$$

Increase t and repeat from 2 while $t < \lambda$:

- *t* denotes current iteration
- λ is the limit on time iteration
- W_v is the current weight vector
- *dist* is the target input
- $\beta(t)$, where β is the estimate of an analysis performed on variables that have been standardized so that they have variances of 1, represents restraint due to distance from BMU, usually called the neighborhood function, and
- $\alpha(t)$, where α is the conventional symbol for angular eccentricity, is learning restraint due to time.

The above parameters determine the efficiency of the SOFM algorithm. The input vectors will have three components, and each will correspond to a color space. The target input will correspond to a color space, somewhere on the RGB map [6]. Each weight vector, including the current one, is of the same dimension as the node's input vector. The current iteration must always be less than the limit on time iteration. The weight vector should be as small as possible. There is also the history limit, which keeps track of the input history, and a history factor to update previous winning nodes has been implemented to decrease the time necessary to complete the training phase.

C. Strengths and Weaknesses of the Self-Organizing Feature Map Algorithm

The main strength, or advantage, of using a SOFM is that the data is easily interpreted and understood. As a result, patterns can be easily obtained. The reduction of dimensionality and grid clustering makes it easy to observe similarities in the data. SOFMs depend on data; in other words, they factor in all the data in the input to generate these clusters and can be altered such that certain pieces of data have more/less of an effect on where an input is placed. Relying on data instead of an exact mathematical formula is essential because no two SOFMs are ever exactly the same.

One major disadvantage of a SOFM is that it requires sufficient data in order to develop meaningful clusters. The weight vectors must be based on data that can successfully group and distinguish inputs. Lack of data or extraneous data in the weight vectors will add randomness to the groupings. Finding the correct data involves determining which factors are relevant; the ability to determine a good data set is a deciding factor in determining whether to use a SOFM or not.

Another problem with SOFMs is that it is often difficult to obtain a perfect mapping where groupings are unique within the map. Instead, anomalies in the map often generate where two similar groupings appear in different areas on the same map. Clusters will often get divided into smaller clusters, creating several areas of similar neurons [8]. This can be prevented by initializing the map, but this is not an option if the state of the final map is not obvious.

III. PROBLEM SOLUTION

A. An Application to Sound

This program can accept any type of sound file, so long as it is in the text format described above. As a result, several types of different sounds were analyzed, but the focus of this project went into color noises, such as white noise, pink noise, etc. These types of noises were chosen due to their many different uses. For example, white noise negates other sound, while pink noise is often used as a reference signal. Red noise, violet noise, blue noise, and gray noise were also used. Black noise, which is total silence, was not used, since humans cannot perceive it and can only be heard in a vacuum [9].

B. Simulation Results and Discussion

First, different color noise sounds were gathered. These types of sounds were used because, to the human ear, they sound very similar, but not quite the same. When displayed in a joint time-frequency diagram, however, the differences are quite clear. This makes them excellent candidates to use in the SOFM, since the overall objective is to distinguish between sounds that humans have difficulty doing.

The sounds were first preprocessed in MATLAB by plotting the sound data into a joint time-frequency domain [10]. The magnitude of the matrix was calculated using the abs() function, which created a matrix. This matrix could be converted into a text file, which was then be read by the Java application. The extracted matrix is analyzed by the feature extraction algorithm. The output is a feature vector whose content can range from 4 to 14 features, which is the optimal range.

Time and Frequency Domain:

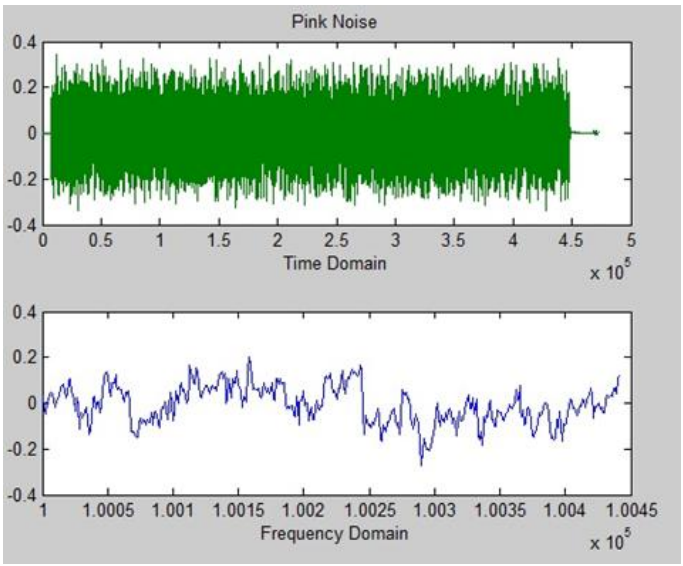

Figure 1: Pink noise in the time and frequency domain

Joint Domain:

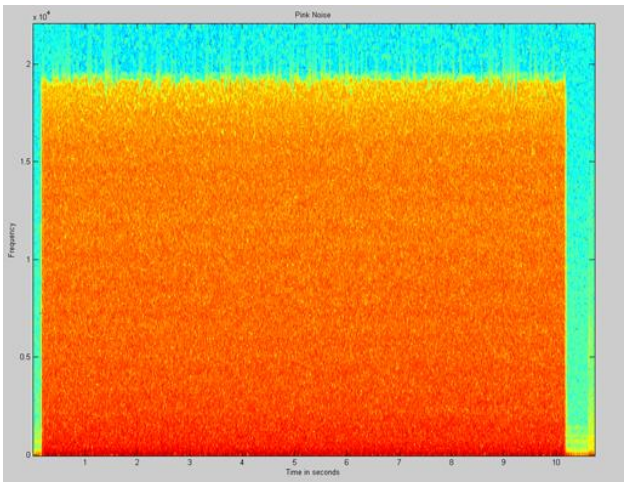

Figure 2: Pink noise in the joint time-frequency domain

The nodes in the SOFM are initialized with small random weights. This is an important step, since the initial weight determines the learning speed of the map. The larger the learning rate, the more iterations it will take. However, if the numbers are too small, then the overall quality of the map will be affected. The user will be able to see on the SOFM that the noises did not cluster very well, since the colors will not neatly group together on the visual display. The next step is to cluster and classify the training sounds using the learning algorithm. In this step, the clustering of the sound could be affected by the input sequence, since this SOFM uses a history function. Using a random input sequence makes it harder to track the pattern, in

comparison to a sequential input sequence. After training, different random noises were used for testing the map. In order to obtain similar but slightly different sound samples, the color noises were altered using Adobe Sound Booth.

In addition, the differences between random input and sequential input were analyzed. Random input means that the files were selected randomly to be put into the SOFM. The sequential input used the data in the files in the order that they were written. The results are shown below.

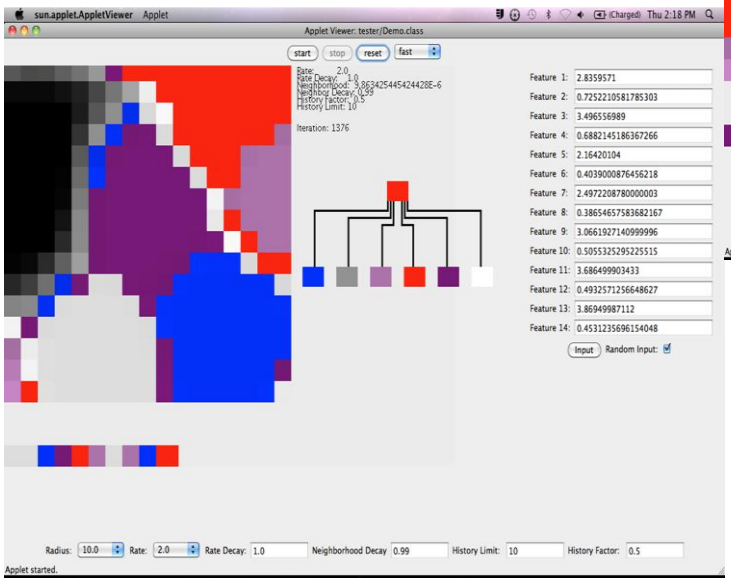

Figure 3: Visualization of random input

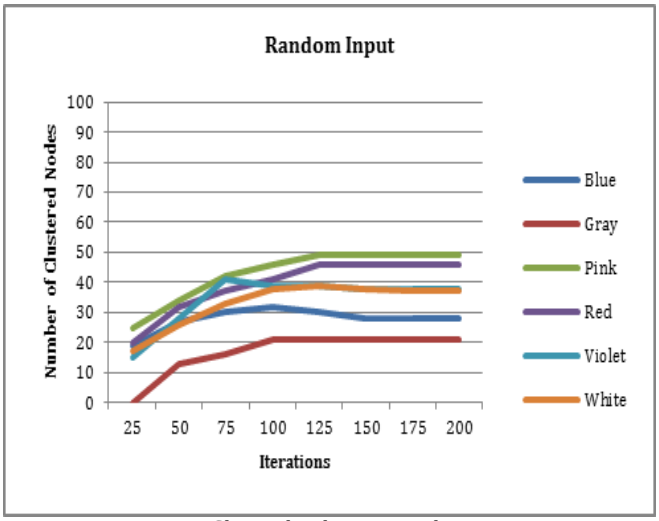

Figure 4: Clustered nodes using random input

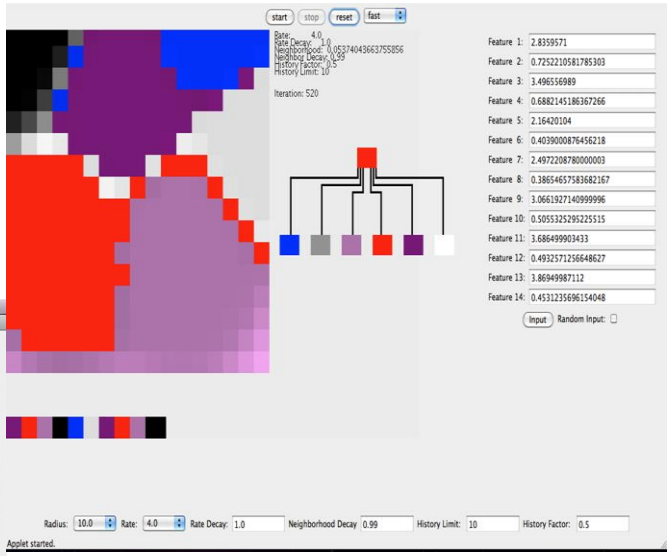

Figure 5: Visualization of sequential input

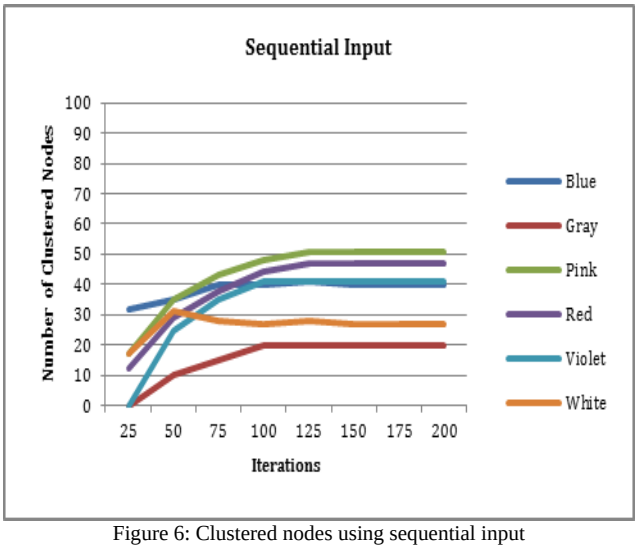

Figure 6: Clustered nodes using sequential input

The graphs above represent a sample clustering of two hundred iterations. The y-axis displays the number of nodes that were clustered of each color. The x-axis shows how many nodes were clustered at each iteration. As seen in Fig. 3 and Fig. 4, the sequential input achieves a stable level much faster than the random input. This is because it is much easier to predict patterns when the input order is known, which Fig. 5 and Fig. 6 make clear.

To further investigate this theory, another sample clustering was done. The range of the small random number initialization was reduced from 0 to 1 to 0 to .5. This test is to evaluate if smaller numbers affect the clustering speed. In essence, the results showed that smaller numbers would cluster faster than larger random numbers.

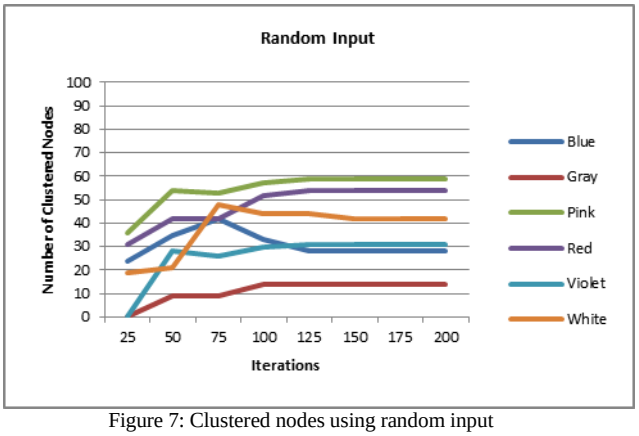

Figure 7: Clustered nodes using random input

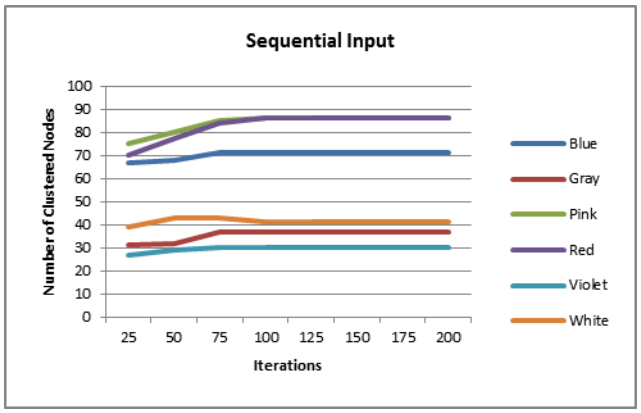

Figure 8: Clustered nodes using sequential input

Even by significantly changing the factors, the sequential input clustered far better than the random input by achieving a stable level in fewer iterations. The graph with the random input did not level out until 150 iterations, while the sequential achieved the same result between 75 and 100 iterations. Thus, the sequential input proved to be far more successful in the SOFM clustering algorithm.

After testing the program multiple times, it was found that the best result was achieved using a sequential input rather than a random input sequence, as shown by the pictures. The order in which the data is presented to the program clearly makes a difference. By using the sequential method, a faster learning and a more precise clustering was obtained. Additionally, in the testing phase of the program, we got a proximal 73.3% of positive hit of 30 unknown sounds. However, the random input is not a truly random selection; it is based on the time executed, as well as other several different factors. While this proved sufficient for this project, a more accurate representation of random input might be obtained by other methods. Moreover, the previous version of this project determined the importance of the history to increase accuracy. There is also the history limit, which keeps track of the input history, and a history factor to update previous winning nodes has been implemented to decrease the time necessary to complete the training phase. The range (range=max-min) and arithmetic average of each quadrant and sub-quadrant proved to be satisfactory features for the SOFM.

C. Watershed transform analysis in joint (time-frequency) domain

There is a way of making the SOFM images clearer. Watershed segmentation is often used to separate touching objects in an image. The watershed transform is often used by finding "catchment basins" and "watershed ridge lines" in an image by treating it as a surface with high light pixels and low dark pixels.

Segmentation using the watershed transform works better if one can identify, or "mark," foreground objects and background locations. Marker-controlled watershed segmentation follows this basic procedure:

1. Compute a segmentation function, which is an image whose dark regions are the objects you are trying to segment.
2. Compute foreground markers. These are connected blobs of pixels within each of the objects.
3. Compute background markers. These are pixels that are not part of any object.
4. Modify the segmentation function so that it only has minima at the foreground and background marker locations.
5. Compute the watershed transform of the modified segmentation function.

Fig. 9 shows the image of the self-organizing map in gradient magnitude, but it is not clear enough without further preprocessing. This visualization illustrates how the locations of the foreground and background markers affect the result. In around two locations, partially occluded darker objects were merged with their brighter neighbor objects because the occluded objects did not have foreground markers.
A visualization technique that is shown in Fig. 10 is to superimpose the foreground markers, background markers, and segmented object boundaries on the original image. Another useful visualization technique is to display the label matrix as a color image. This is superimposed by transparency, as displayed in Fig. 11.

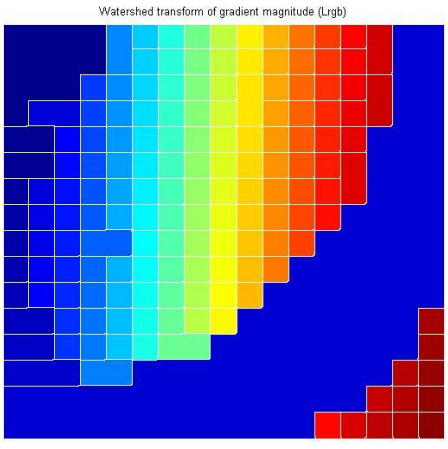

Figure 9: Watershed transform of gradient magnitude

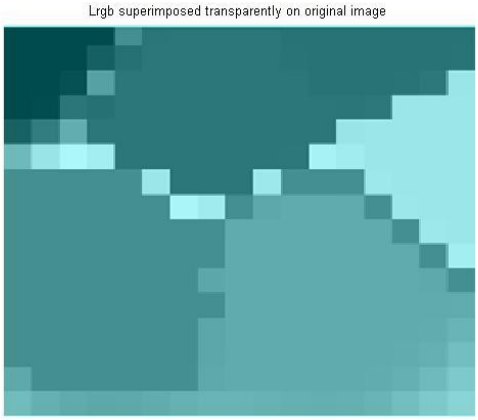

Figure 10: Watershed transform superimposed transparently on original image

Figure 11: Watershed transform superimposed transparently on original image

Pre processing methods are used to reduce the noise of image and adjust the image intensity [11]. This is not altogether different from the process of extracting features from sounds. As can be seen, the watershed transformations help to further subdivide between clusters and make each one clearer. If a cluster were to overlap, image segmentation helps to tell them apart. The watershed transform also assists the smaller clusters to stand out more than they normally would.

IV. CONCLUSION

Overall, the results turned out to be successful. The program was able to accept the text files of the magnitudes from the joint time-frequency matrices, divide them into quadrants, and obtain 14 features to put into one feature vector. The SOFM (self-organizing feature map) classified random inputs according to the features and accurately clustered the data. Moreover, it provided a context for input features with a tree diagram and displayed the color that most resembled the sound. After rigorous testing, the SOFM performed best with nonrandom input, a radius of 10.0, a learning rate of 4.0, a rate decay of 1.0, a neighborhood decay of 0.99, a history limit of 10.0, and a history factor of 0.5. This was determined by how well the colors clustered together and how many randomly colored squares were left on the SOFM. Of course, future work can always be done. The project can be expanded to include several different noise samples, rather than just the color noises. Moreover, even more features could be added for different results.

ACKNOWLEDGMENTS

First and foremost, the authors would like to thank Oakland University, for the research opportunity. Also, thank you Matthew Bradley, Kay Jantharasorn, and Keith Jones, who created this program that the future contributors. Additional thanks to Kate LaBelle and Kemuel Cruz for their further work on the program. This research work was originally conducted at Oakland University in the UnCoRe program - REU funded by the NSF under grant number 1062960, and then independently at Oakland.

REFERENCES

[1] Phillip D. Waserman, “Advanced Methods in Neural Computing”, International Thompson Publishing, UK’ 93.

[2] Vesanto, J., and E. Alhoniemi, “Clustering of the Self-Organizing Map.” IEEE Transactions on Neural Networks, vol. 11, No. 3, May 2000. Neural Networks Res. Centre, Helsinki Univ. of Technol., Espoo, Finland.

[3] Kohonen, Teuvo. Self-Organizing Maps. Berlin: Springer, 2001.

[4] Nsour, Ahmad R.. and Mohamed A. Zohdy, “Self organized learning applied to global positioning system (GPS) data”, Proceedings of the 6th WSEAS International Conference on Signal, Speech and Image Processing. September 22-24, 2006. pp. 203-208, Lisbon, Portugal.

[5] Bradley, Matthew, Kay Jantharasorn, Keith Jones, and Dr. M. Zohdy, "Self Organized Neural Networks Applied to Animal Communications", REU Report, unpublished.

[6] Abdel-Aty-Zohdy, H. S., and M. A. Zohdy. “Self-Organizing Feature Maps.” Wiley Encyclopedia for Electrical Engineering. Dec 27, 1997. pp. 767-772.

[7] *Audio Processing*. Web. <http://www.aquaphoenix.com/lecture/matlab10/page4.html>.

[8] Pang, Kevin, “Self-organizing Maps”, unpublished.

[9] "The Science of Noise." Web. <http://www.youtube.com/watch?v=FKg1dm5tF4>.

[10] Aguado, Alberto S., and Mark S. Nixon, "Feature Extraction and Image Processing", Linacre House, Jordan Hill, Oxford OX2 8DP, UK. 2008.

[11] Bala, Anju, An Improved Watershed Image Segmentation Technique using MATLAB, International Journal of Scientific & Engineering Research Volume 3, Issue 6, June-2012.