ISBN 978-93-84468-20-0
Proceedings of 2015 International Conference on Future Computational Technologies (ICFCT'2015)
Singapore, March 29-30, 2015, pp. 54-60

RTAP: Towards a Real-Time Auditory Periphery Simulation

Ram Kuber Singh

University of Western Sydney

***Abstract*:** *From rocket propulsion to monitoring intricate surgeries to mobile phones, a real-time software application has wide applications for numerous fields as it provides a platform in providing information instantly. It does this by breaking down a continuous stream of information into small manageable packets where they are processed and output from the system. This paper describes the design of a real-time software application of the auditory pathway using multiple threads and multi-core processor to achieve dynamic computational loads. Small to large scale simulations are possible with such a computer model and therefore, it is a useful tool in neuroscience and signal processing as part of expansions of hearing and audio processing researches respectively.*

Keywords: *real-time, auditory, cochlear, simulation*

1. Introduction

A real-time software application (RTSA) relies heavily on optimal access to the features of the central processing unit (CPU). In doing so, the RTSA can be broken down to smaller parts called threads and every thread is processed at fixed short durations. Given that most CPUs are capable of processing vast number of clock cycles, multi-threaded processing in a serialized manner on a powerful CPU provides an outlook of pseudo-concurrent processing. With the advent of multi-core processors, the attribute of concurrent parallel processing can truly be achieved on readily available general purpose operating systems (GPOS) [1].

Real-time auditory periphery (RTAP) is one such RTSA that harnesses multiple-threading attributes of a GPOS such as Microsoft Windows operating system (OS) and the parallel computing prowess of multi-core processors on general purpose computers. It utilizes threads to encapsulate audio data acquisition, algorithm processing and output display as well as data recording to attain real-time behaviour. In addition, RTAP also has dynamic computational loading that is achievable with configurable parameters. These features will be discussed in the following section.

2. Real-time Software Design

RTAP is a graphical user interface (GUI) Windows based application developed in C++ [2]. It utilizes JUCE graphics and audio library and has been optimized to run on Intel processor with the use of Intel Parallel Studio integrated into Microsoft Visual Studio. The sections below describe the major features of the application.

2.1. The Auditory Model

The algorithms of the auditory model are segmented based on the various stages of the auditory pathway as illustrated in figure 1. The input and output (IO) stream for outer and middle ear (OME) modules consists of a single channel as depicted by the thin arrow in figure 1. The OME module is made of a serial cascade of infinite impulse response (IIR) based filters [3].

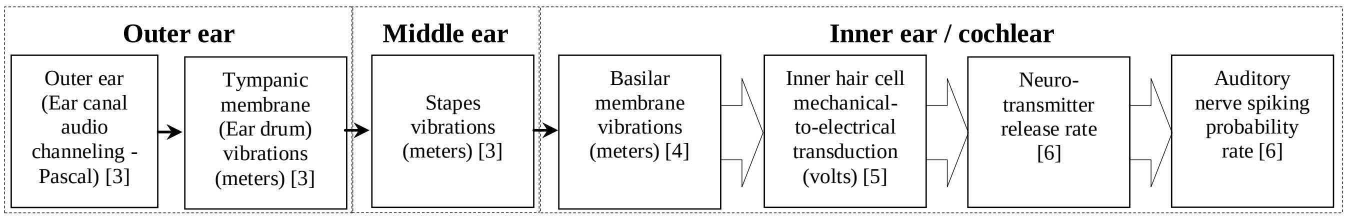
Outer ear
Middle ear
Inner ear / cochlear
Outer ear (Ear canal audio channeling - Pascal) [3]
Tympanic membrane (Ear drum) vibrations (meters) [3]
Stapes vibrations (meters) [3]
Basilar membrane vibrations (meters) [4]
Inner hair cell mechanical-to-electrical transduction (volts) [5]
Neuro-transmitter release rate [6]
Auditory nerve spiking probability rate [6]

Fig. 1: Algorithm modules in RTAP.

From the basilar membrane (BM) onwards, the IO stream consists of multiple parallel channels. This quantity selectivity is set before the simulation is started. Each parallel channel is limited by varying bandwidth with a Gaussian function or bell-shaped frequency response. The peak of each bandwidth called best frequency (BF) corresponds to the maximum response each discrete site of a BM is capable of generating. This feature of the BM is implemented using a nonlinear gammatone filterbank, which comprises of a number of bandpass filters with varying bandwidth and an exponential function to represent its inherent log-scaled nonlinearity [4].

Inner hair cells (IHC) are stationed along the length of the BM. The BM vibrations mechanically influence the IHC corresponding to discrete sites of unique frequencies along the BM to vibrate as well. These nonlinear motions cause the inflow and outflow of charged potassium ions to the IHC resulting in variations of voltages in the IHC [5]. The voltage, in turn influences the rate of neurotransmitter release (NRR) from the base of the IHC, which finally results in electrical spikes initiated from the auditory nerve (AN) [6]. This activity of the AN is defined quantitatively as the AN spiking probability (ANSP) rate. Therefore, the mechanical travelling wave of the BM corresponding to the specific BF and bandwidth is able to activate the firing of the associated AN spikes based on the spectral contents of the streamed input audio.

2.2. Software Attributes

The software application is required to run in real-time on Windows which is a general purpose operating system (GPOS). This can be achieved by altering the process priority of RTAP to run at either ‘Real-time’ or ‘High’ setting, thereby ensuring that the application receives as much central processing unit (CPU) time as possible.

The audio is sampled at 22.05 KHz and is streamed continuously as a block of 1,280 samples with each sample being 32-bit floating point (FP). The streamed audio is available at a deterministic time interval of 58ms via JUCE audio library which provides an abstract of microphone audio streams available through the Windows operating system (OS). A software sine wave generator is integrated into the audio data stream acquisition function to utilize its regular 58ms timing intervals to provide a continuous sine tone based on the selectivity feature.

Parallel processing is achieved using multiple threads coupled with features of Intel Parallel Studio. Each of the three tasks namely algorithm, display and data recording are grouped into three separate threads that are able to be invoked from the application. The algorithm thread encompasses the algorithm modules from figure 1. The display thread projects output signals on to the computer screen. The record thread, which is an optional feature triggered based on a separate setting, writes output signals on to a file. Figure 2 illustrates the RT characteristics implemented using POSIX threads.

To ensure as many BF channels are included in the real-time (RT) processing, several schemes are utilized for algorithm optimization and processing time reduction. One such scheme is the initialization phase before its runtime. During this phase, memory resources are allocated, variables initialised and constants computed based on the user defined parameter settings. Display and record threads are started in this phase and all its resources are allocated before they are temporarily suspended in a wait mode. These two threads are put in a ready-to-run state after the algorithm thread is processed. Hence, this initialization stage accommodates for the notorious latency timings inherent in memory management as well as start-up of threads and removes computation redundancy in variables that remain fixed during runtime.

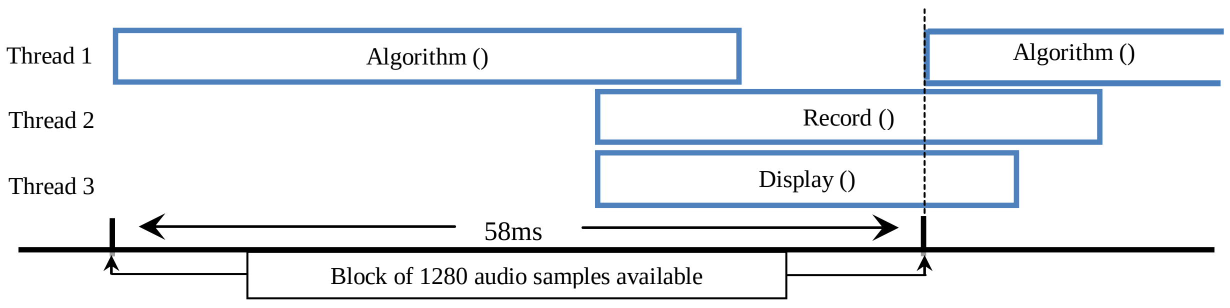

Fig. 2: Multiple threads implementation of RTAP.

2.3. Output Response Display

RTAP is capable of displaying parallel output signals processed from any one of the inner ear algorithm modules described in section 2.1. As every BF is logarithmically spaced from one another, the output signals are also spaced in a nonlinear manner on the y-axis. Displaying all output signals in constant intervals is made possible with a Greenwood function [7]. A modified version of the Greenwood equation is implemented in RTAP and is defined as follows:

$$y_{\log}(t) = Y_{start} - 21.4\log_{10}(0.00437f + 1) + \frac{y_{BF}(t) - y_{\max}(t)}{y_{\max}(t) - y_{\min}(t)} 2Y_{spacing} \tag{1}$$

$y_{log}(t)$ is the y-coordinate representation of the response signal to be displayed on log scale. Y_{start} is the offset from the point of origin of RTAP display window. y_{BF} is the raw output of an algorithm module. y_{max} and y_{min} are the maxima and minima for output signals in a single frame. $Y_{spacing}$ is the vertical distance between two adjacent BF processed signals defined by equation 2 as follows:

$$Y_{spacing} = \frac{Y_{end} - Y_{start}}{n_{BF} + 1} \tag{2}$$

Y_{end} and Y_{start} are the final and beginning vertical points on RTAP display window where the pixels are plotted respectively. n_{BF} represents the number of BF channels.

The modified Greenwood plot display is ideal for small number of BF channels. It becomes difficult to comprehend contents of such a plot for large number of BF channels. A spectrogram becomes advantageous in such a situation as it uses colours to differentiate intensity in the output signals as an added dimension and provides a linear method of display by removing empty spaces in the y-axis. Therefore, every row in a spectrogram represents a BF channel. The pixel colours are defined in the range of maxima and minima of signals within all the BF channels in a single frame of 58ms. Equation 3 defines the colour index acquisition format.

$$m = \frac{y_{BF}(t) - y_{\min}(t)}{y_{\max}(t) - y_{\min}(t)} N \tag{3}$$

m is the color index for a pixel in the spectrogram defined by output signal, $y_{BF}(t)$. N is the total number of discrete colors set at a constant of 30.

To accommodate the display of output signals in real-time, image scroll is utilized where the output plot is shifted on the computer screen from right to left. This is achieved by off-screen rendering of the image buffer which ensures that a plot is drawn in its entirety before its display. Display data subsampling is also implemented where every frame of display data are condensed before display. This is to ensure that as many continuous frames of data are displayed on screen before scrolling off. This is achieved by retaining every one-hundredth data sample within every signal of a BF channel. Figure 3 displays output signals from RTAP using the modified Greenwood function and a spectrogram respectively with a 500Hz pure sine tone input.

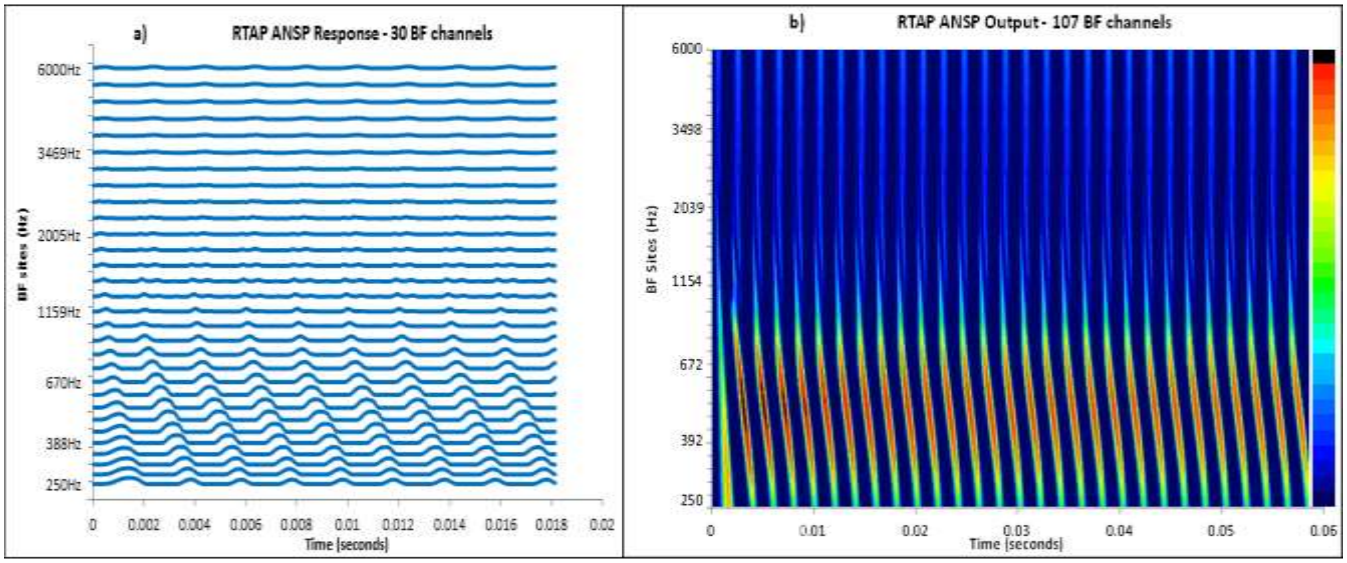

Fig. 3: RTAP output (a) Modified Greenwood plot; (b) Spectrogram plot.

2.4. Software Optimization

Software optimization is achieved through utilization of fast mathematical operations. Basic mathematical operators such as addition, subtraction, multiplication and division use either one or several CPU clock cycles [8]. These operators are not considered for optimization. Only complex mathematical operators such as exponential, natural log and sine and cosine functions are considered for optimization.

Sine and cosine functions are used only during the initialization stage for pre-setting variables based on selectivity of parameters and thus are not time critical. Out of the remaining operators, exponential and natural log functions, the former is utilized five times more per BF channel processing during RT runtime. Therefore, the exponential function was given credence over natural log in terms of optimization. Table I summarises the use of exponential and natural log functions during RT runtime.

An optimised exponential function is achieved using Schraudolph formula [9]. It is based on an 8-byte floating point (FP) format divided into 2 integer halves. The upper half containing the sign, exponent and mantissa bits are manipulated while the lower half is ignored. A bias of 1023 is added to the integer and shifted to the left by 20-bits giving the result 2^y. The result is divided by ln (2) and then read back as FP format to attain e^y. This is defined by equation 4 as follows:

$$i := ax + (b - c) \tag{4}$$

i represents the integer-based manipulation of a FP number. *x* is the input number to be exponentiated. *a* is the 20-bit left shifted scalar defined by $2^{20}/\ln(2)$. *b* is the 20-bit left-shifted bias 1023×2^{20}. *c* is a fine-tuning parameter.

TABLE I: Survey of exponential and natural log functions used in RTAP

Algorithm Modules	Number of invocations per BF channel		Maximum number of BF channels used in simulation*	Total number of invocations in simulation*	
	exp()	log()		exp()	log()
BM	1	1	178	178	178
BM-to-IHC	3	1	125	375	125
BM-to-NRR	5	1	102	510	102
BM-to-ANSP	5	1	89	445	89

*Computer used is specified in figure 5.

3. Results

Figure 4 displays the runtime attributes of the threads utilized in RTAP under debug mode for two frames of audio input and a loading of 88 BF channels recorded by Intel thread checker as part of Intel Parallel and Microsoft Visual Studios. The computer used is described in figure 5. The algorithm thread represented by ‘Algothread’ begins first followed by data record thread and finally pixel draw thread. After initializations, all three threads are placed in wait mode where parameters are initialized and memory resources are managed.

The algorithm thread runs at 0.25s and 0.38s respectively after acquiring audio input. Upon its conclusion, signals record and draw pixels threads are signalled to run and at approximately 0.35s and 0.47s, the two threads are executed in parallel. The time duration of the algorithm thread exceeds 58ms because RTAP was compiled in debug mode for acquiring runtime feedback data. However, the purpose of this test was to ensure threads operational integrity is maintained during runtime and that they matched closely to its design.

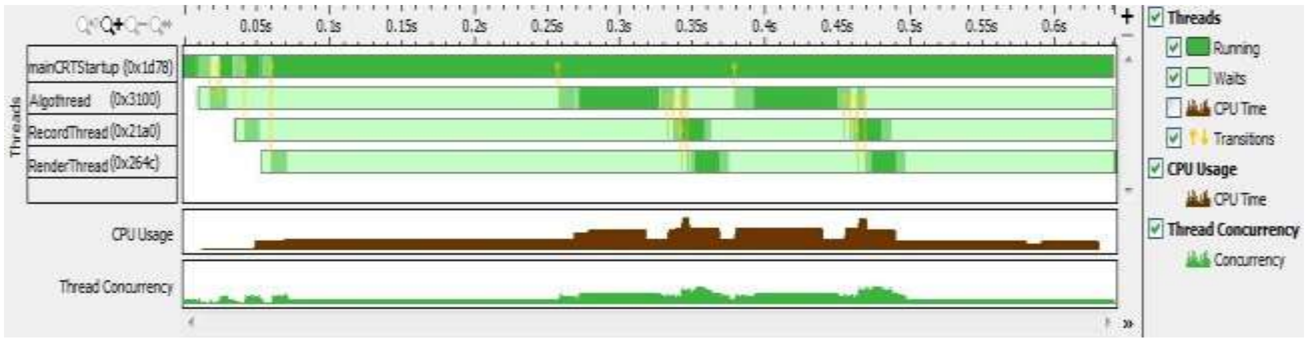

Figure 4: RTAP thread utilization with 88 BF channels.

Figure 5 displays RTAP runtime load information based on its runtime priority on Windows OS, selected algorithm and the number of parallel BF channels the software is capable of processing within a time cap of 58ms. The audio used is a 500Hz sine tone. The same figure also specifies the computer type used for attaining the results.

Figures 5a and 5b indicate that running the BM algorithm on RTAP alone is able to produce optimum BF channel loading under real-time priority. On the contrary, running BM-to-ANSP algorithm under normal priority generates the least loading. This is because real-time priority allows RTAP to maximize the use of CPU resources whereas normal priority allows RTAP to either share the CPU with other software application running under the same priority or yield the CPU more often to higher priority applications. Furthermore, running ANSP module requires the accumulative processing of BM, IHC and NRR algorithm modules. Therefore increasing algorithm modules reduces BF channel loading.

Figure 5b illustrates higher BF channel loading across all algorithms and priority levels as a result of the utilization of Schraudolph exponential function as opposed to conventional math C++ library usage as demonstrated in figure 5a. Approximately 5% increase is observed for BM algorithm as opposed to IHC, NRR and ANSP algorithm processing, which attained an increase loading of approximately 21%, 28% and 25% respectively. As the BM algorithm does not encompass as many exponential functions as the other algorithms, the improvement of increased BF channel loading is not as significant.

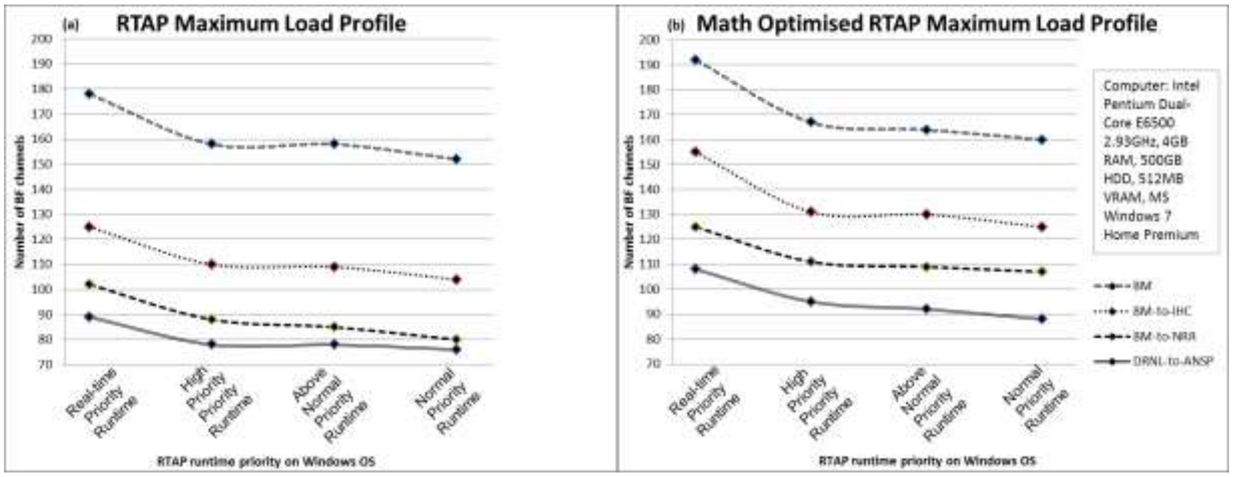

Fig. 5: RTAP runtime load profile.

4. Conclusion and Future Work

RTAP, a real-time multi-threaded Windows application of the auditory pathway with dynamic loading has been described. Due to its dynamic characteristics, the loading of multiple channels, priority-based CPU allocation and algorithm selectivity can be variably set for other computers running different CPUs. It is able to run multiple threads that are capable of harnessing parallel computing capability of a multi-core processor to compute, display and store computed data to a file.

The dynamism of RTAP allows many features to be implemented for future releases. Breaking down algorithm modules into smaller parts and encapsulating them in threads may achieve further optimization and higher loading of channels. As there are a number of configurable parameters in this model, one potential feature is to utilize scripts to load and save simulation configurations. The sine tone generator may be expanded to include more input signals such as square and sawtooth.

5. References

[1] S. Akhter and J. Roberts, Multi-Core Programming: Increasing Performance through Software Multi-threading, 1st ed. Intel Press, 2006, pp. 1–336.

[2] R. K. Singh, “Real-time Auditory Periphery (RTAP),” 2012. [Online]. Available: https://code.google.com/p/rtap/source/list.

[3] E. A. Lopez-Poveda and R. Meddis, “A Human Nonlinear Cochlear Filterbank,” J. Acoust. Soc. Am., vol. 110, no. 6, pp. 3107 – 3118, 2001.
http://dx.doi.org/10.1121/1.1416197

[4] R. Meddis, L. P. O’Mard, and E. A. Lopez-Poveda, “A Computational Algorithm for Computing Nonlinear Auditory Frequency Selectivity,” J. Acoust. Soc. Am., vol. 109, no. 6, pp. 2852 – 2861, 2001.
http://dx.doi.org/10.1121/1.1370357

[5] C. J. Sumner, E. A. Lopez-Poveda, L. P. O’Mard, and R. Meddis, “A Revised Model of the Inner-Hair Cell and Auditory-Nerve Complex,” J. Acoust. Soc. Am., vol. 111, no. 5, pp. 2178 – 2188, 2002.
http://dx.doi.org/10.1121/1.1453451

[6] R. Meddis, “Auditory-nerve First-spike Latency and Auditory Absolute Threshold: A Computer Model,” J. Acoust. Soc. Am., vol. 119, no. 1, pp. 406 – 417, 2006.
http://dx.doi.org/10.1121/1.2139628

[7] B. R. Glasberg and B. C. Moore, “Derivation of auditory filter shapes from notched-noise data.,” Hear. Res., vol. 47, no. 1–2, pp. 103–138, Aug. 1990.
http://dx.doi.org/10.1016/0378-5955(90)90170-T

[8] A. Fog, “4. Instruction tables - Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs,” 2014.

[9] N. N. Schraudolph, “A Fast, Compact Approximation of the Exponential Function,” Neural Comput., vol. 11, pp. 853–862, 1999.
http://dx.doi.org/10.1162/089976699300016467