Simulation and Real-Time Implementation for Teaching 3D Sound

Kenneth John Faller II and Armando Barreto

Electrical & Computer Engineering Department
Florida International University
Miami, FL, 33174, USA

This paper describes a software demonstration program, implemented in Matlab®, which has been designed to facilitate the teaching of how 3D sound is synthesized using computerized techniques. The demonstration program simplifies the explanation of the fundamentals of 3D sound through interactive visual and auditory examples. As an extension to the Matlab® demonstration program, a real-time implementation of continuous sound spatialization was developed using the Texas Instruments TMS320C33™ DSK Starter Kit. A complete description of the implementations is given and the necessary code is shared with interested educators at a designated web site.

I. Relevance of teaching the principles of digital sound spatialization.

A large portion of our assimilation of the environment that surrounds us depends on our ability to determine the location of diverse sound sources. This fact has been exploited in the development of synthetic immersive environments, such as those created for virtual reality and computer gaming. Because of the growing importance of these technical areas, and the impact of sound spatialization on broader fields, such as human-computer interaction, it is important to provide students in technical programs with a basic knowledge of how 3D sound is created in computers. While many technical curricula do not include explicitly a space for the introduction of this type of knowledge, a freely available, pre-packaged demonstration, such as the one described here, may provide interested educators with a simple, entertaining and succinct vehicle to include this topic in their lectures.

When a sound produced by a physical source is perceived our brains are able to determine the approximate location of the source, in a 3-dimensional space, as shown in Figure 1. In this diagram we note that the "horizontal" location of the source is characterized by an azimuth angle, and the "vertical" location is defined by an elevation angle.

The exact mechanisms used by our brains to estimate the azimuth and elevation of a sound source from the sound itself are still being investigated. It is known, however, that important azimuth and elevation cues are determined by time and intensity differences with which the sound waves arrive to each ear and through the spectral changes produced by the head and the pinnae or outer ears. Further, synthetic replication of these effects can actually be used to emulate the apparent location of a sound source in space.

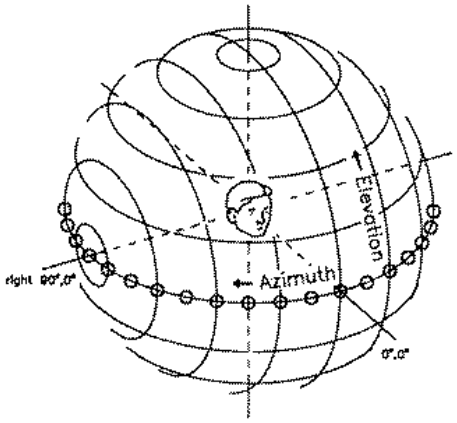

Figure 1: Diagram of spherical coordinate system (from Makous & Middlebrooks [7])

The ability to alter the apparent location of a sound source in space is of great value. Areas of importance that would benefit from the accurate synthesis of 3D sound are: human-computer interface, multimedia applications such as video games, aids for the vision impaired, virtual reality systems, "eyes-free" displays for pilots and air-traffic controllers, spatial audio for teleconferencing and shared electronic workspaces, and auditory displays of scientific or business data [1].

II. 3D Sound generation approaches

Currently, there are two approaches to 3D sound system implementation: multi-channel or two-channel. A multi-channel system consists of a number of speakers physically positioned at desired points around the listener (i.e. Dolby® 5.1 setup). Although this is an effective solution it is often

times cumbersome and expensive. The alternative is the two-channel solution. The key to this approach is the Head-Related Transfer Functions (HRTFs).

HRTFs are signal processing filters applied to sound signals to simulate 3D positional sound. They account for the modification of sound waves by the head, shoulders, torso, and the pinnae. Each location around a listener (characterized by an azimuth, an elevation and a radial distance away from the listener) has a pair of corresponding HRTFs that change the original sound in the same way as the head and torso of the listener would change the sound from its physical source (at that location) before reaching the left and right eardrums of the listener. Strictly speaking, the HRTFs for a given source location will vary from person to person depending on the listeners' physical attributes, although "generic" HRTFs measured from a manikin of average dimensions are frequently used.

III. HRTF Implementation

The demonstration program described here uses the Matlab® function "filter" to create a binaural (left & right) pair of sounds from a monaural input "wave" file. The "filter" function performs the convolution of the digitized input sound with the impulse response of the appropriate HRTF. The impulse response of a filter is the output sequence obtained from the filter when the input is a discrete impulse. In the case of an HRTF we call the impulse response a "Head Relate Impulse Response" or "HRIR". The operations performed by Matlab® when the "filter" routine is executed are as shown in Figure 2 [4]

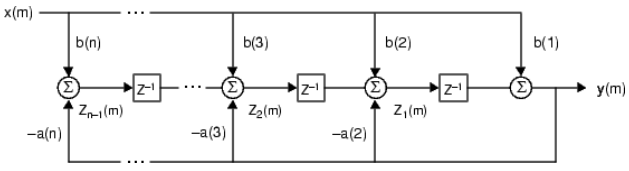

Figure 2: Direct form II transposed structure (from Matlab® [4])

or

$$y(n) = b(1)*x(n) + b(2)*x(n-1) + \ldots + b(nb+1)*x(n-nb) - a(2)*y(n-1) - \ldots - a(na+1)*y(n-na) \quad \text{Eq.(1)}$$

where n-1 is the filter order, [8].

Alternatively, the operation of filter at sample m is given by the time domain difference equations,

$$\begin{aligned} y(m) &= b(1)x(m) + z_1(m-1) \\ z_1(m) &= b(2)x(m) + z_2(m-1) - a(2)y(m) \\ \vdots &= \vdots \quad\quad \vdots \\ z_{n-2}(m) &= b(n-1)x(m) + z_{n-1}(m-1) - a(n-1)y(m) \\ z_{n-1}(m) &= b(n)x(m) - a(n)y(m) \end{aligned} \quad \text{Eq.(2)}$$

The input-output description of this filtering operation in the z-transform domain is a rational transfer function,

$$Y(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(nb+1)z^{-nb}}{1 + a(2)z^{-1} + \ldots + a(na+1)z^{-na}} X(z) \quad \text{Eq.(3)}$$

Since the HRTFs measured for the purpose of 3D sound spatialization are modeled as non-recursive filters (i.e., no internal feedback is assumed in the HRTF models), all the "a" coefficients in the above equations will be assumed to have a value of zero, which simplifies the calculations significantly. In fact, under these conditions, the "b" coefficients in the above equations are directly the numerical values that constitute the HRIRs.

IV. Re-sampling and Truncation of the MIT HRTFs

Measurement of the HRIR pairs (left and right) that correspond to a large number of sound sources around a listener requires an elaborate experimental setup. Fortunately, W. Gardner and K. Martin, from MIT have performed those measurements on a KEMAR dummy with a microphone in each ear canal and made the numerical results available on their website. The contents of their database consist of the left and right ear impulse responses from the manikin. Maximum length (ML) pseudo-random binary sequences were used to obtain the impulse responses at a sampling rate of 44.1 kHz [2].

For both the Matlab® demonstration program and the real-time implementation in the TMS320C33 DSK kit we have used Gardner and Martin's HRIRs, except that we have transformed them to correspond to the sampling rate we have used, i.e., 48.8 kHz using the resample command in Matlab® [4]. In addition to this, due to memory limitations, the coefficient files were truncated from the original 512 to 256 samples, making sure to preserve the most significant values of the HRIRs (see Figures 3 and 4). Furthermore, we have constrained the location of the emulated sound sources to just the 0^o elevation plane, with azimuths that vary in 10^o increments.

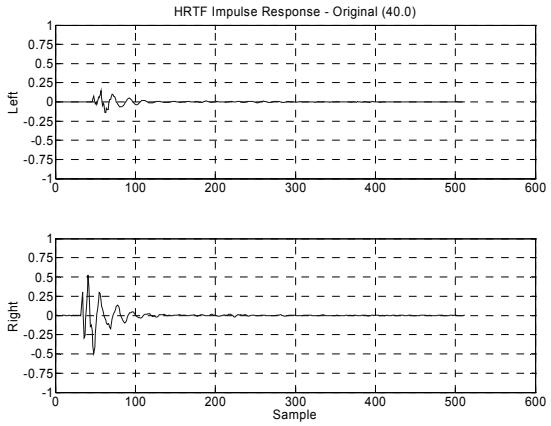

Figure 3: Example of the original impulse responses for 40º azimuth, 0º elevation

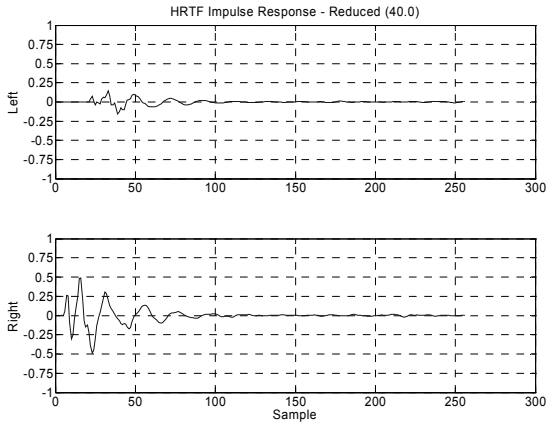

Figure 4: Example of the reduced impulse responses for 40º azimuth, 0º elevation

Display of the "reduced" HRIRs, as in Figure 4, helped confirmed that no essential segments of the impulse responses were destroyed by the shortening process. Additionally, the magnitude response of the original and the "reduced" HRIRs were found to be very similar, as displayed in Figure 5, for example.

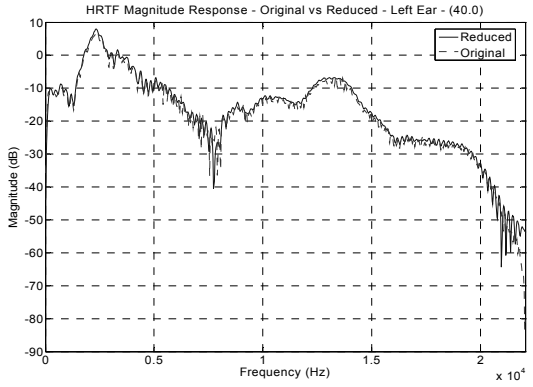

Figure 5: Example of Original vs. Reduced Magnitude response for 40º azimuth, 0º elevation – left ear

V. Matlab® Demonstration Program for Teaching 3D Sound Synthesis

Our 3D sound instructional program was developed in Matlab® because of its superior filtering and graphical user interface (GUI) capabilities. The GUI allows users to experience HRTFs audibly and visually without requiring in-depth knowledge of sound spatialization (see Figure 6).

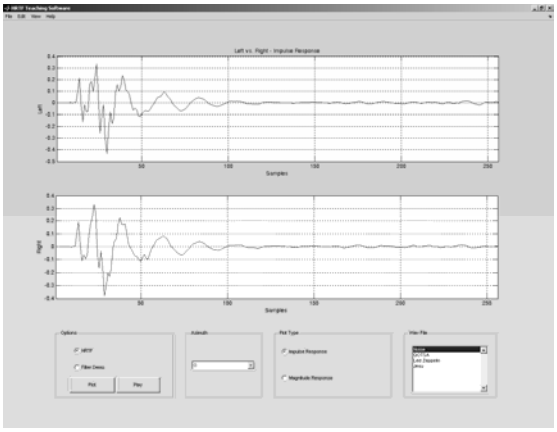

Figure 6: Screen shot of the GUI for the HRTF teaching software

The user is able to select between two modes: HRTF or Filter Demo. The selection is made in the section labeled "Options."

The Filter Demo mode can be used to explain to a student what the program is really doing, in simple and familiar terms. This mode processes one of the available sound samples ("Wave file") with a low pass filter to generate the Right output channel (delivered to the user through the right headphone) and with a high pass filter to generate the Left output channel (delivered to the user through the left headphone). When the student "Plays" the sound through the filters he/she will realize the transformation imposed on the same input sound, to generate the two output channels of the binaural sound. The student also has the ability to plot the characterization of both channels, as either impulse response sequences or as magnitude response plots.

In HRTF mode, the user must choose the desired azimuth from a drop down list in the section labeled "Azimuth." In the section labeled "Plot Type," they must choose between magnitude or impulse response plot types. In addition to this, four audio samples have been provided. The user can select from the four by clicking on the name of the desired sample in the section labeled "Wav File." Once the

appropriate selections are made, the user can either view the results of the selections by clicking the [Plot] button or use the selected HRIRs to filter and [Play] the chosen audio sample.

When the [Plot] button is clicked, the function invoked *plotBtn_Callback* is invoked (Figure 7). This function reads in the necessary coefficient files for the left and right ear using *cof2vec* (Figure 8) then, depending on the user's selections, it plots either the impulse or magnitude response using functions *plotImpulseResp* or *plotMagResp* (Figures 9 and 10 respectively).

If the user clicks the [Play] button then the function *playBtn_Callback* is invoked (Figure 11). In an identical fashion to the function *plotBtn_Callback*, the necessary coefficient files for the left and right ear are read into vectors using the *cof2vec* function (Figure 8). Next, the selected audio sample is read in using the Matlab® command wavread or, if noise is selected, the "rand" command is used to create a pseudo random noise signal [4]. Once the coefficient files and the audio sample have been read in, the audio sample is filtered using the function *hrtfFilter* (Figure 12). Finally, using the Matlab® command "sound," the filtered audio sample is played to the user.

The complete set of Matlab® files needed to run the demonstration program is accessible from http://dsplab.eng.fiu.edu/3DSND_DEMO/.

```
% --- Executes on button press in plotBtn.
function plotBtn_Callback(hObject, eventdata, handles)
   % hObject    handle to plotBtn (see GCBO)
   % eventdata  reserved - to be defined in the future
   % handles    struct with handles and user data (see GUIDATA)
   azimuthVal = get(handles.azimuthList, 'String');
   azimuthIndex = get(handles.azimuthList, 'Value');

   if(get(handles.hrtfRad, 'Value'))
      HR = cof2vec(strcat('../Coeff
Files/H',cell2mat(azimuthVal(azimuthIndex)),'R.cof'));
      HL = cof2vec(strcat('../Coeff
Files/H',cell2mat(azimuthVal(azimuthIndex)),'L.cof'));
   else
      HR = cof2vec('../Coeff Files/AVG.cof');
      HL = cof2vec('../Coeff Files/DIFF.cof');
   end

   if(get(handles.impResponseRad, 'Value'))
      plotImpulseResp(HL,HR,handles.leftPlot,handles.rightPlot);
   else
      plotMagResp(HL,HR,handles.leftPlot,handles.rightPlot);
   end
```

Figure 7: Function called when the plot button is clicked (main.m)

```
function cofFin = cof2vec(fileName)
   % Reads in coefficient file
   cof = textread(fileName,'%s','delimiter','\t,','whitespace','');
   cof = cof(2:size(cof,1)-2);
   N = size(cof,1);
   cof(N-4) = cof(N-9);
   index = 1;
   for i=1:N
      if(mod(i,5) == 1)
         cofTemp = cell2mat(cof(i));
         cofStr = strrep(cofTemp,'.float ','');
         cofFin(index) = str2num(cofStr);
         index = index + 1;
      elseif(mod(i,5) > 1)
         cofStr = cell2mat(cof(i));
         cofFin(index) = str2num(cofStr);
         index = index + 1;
      end
   end
```

Figure 8: Function called to read in coefficient files and returns them as a vector (cof2vec.m)

```
function plotImpulseResp(HL,HR,axesLeft,axesRight)
   axes(axesLeft);
   cla;
   grid on;
   xlim(axesLeft,[1 size(HL,2)]);
   title(axesLeft,'Left vs. Right - Impulse Response');
   ylabel(axesLeft,'Left');
   xlabel(axesLeft,'Samples');
   box on;
   %% Create plot
   hold on;
   plot(HL,'Parent',axesLeft);
   hold off;
   axes(axesRight);
   cla;
   grid on;
   xlim(axesRight,[1 size(HR,2)]);
   xlabel(axesRight,'Samples');
   ylabel(axesRight,'Right');
   box on;
   %% Create plot
   hold on;
   plot(HR,'Parent',axesRight);
   hold off;
```

Figure 9: Function that plots the impulse response (plotImpulseResp.m)

VI. Real-Time Implementation for Teaching 3D Sound

A real-time implementation of the same concepts used for the Matlab® demonstration program was developed using the Texas Instruments TMS320C33 DSK Starter Kit (~ $150) and Borland 4 C++. The TMS320C33 DSK contains a Burr-Brown PCM3003 Stereo Audio Codec, which allows for dual-channel processing of an audio signal. This allows users to experience HRTFs through a real-time system. Furthermore, this implementation can create a spatialized version of any monaural sound fed into it, for as long as necessary. The user can increment or decrement the azimuth used for spatialization by 20° using the left or right arrow keys (Figure 13).

```
function plotMagResp(HL,HR,axesLeft,axesRight)
   axes(axesLeft);
   cla;
   [HL,FL] = freqz(HL,1,1000,48800);
   hold on;
   s.plot   = 'mag';      % Plot magnitude only
   s.xunits = 'khz';      % Label the freq. units correctly
   s.yunits = 'dB';       % Plot the magnitude squared
   freqzplot(HL,FL,s);
   title(axesLeft,'Left vs. Right - Magnitude Response');
   ylabel(axesLeft,'Left Magnitude (dB)');
   hold off;

   axes(axesRight);
   cla;
   [HR,FR] = freqz(HR,1,1000,48800);
   hold on;
   s.plot   = 'mag';      % Plot magnitude only
   s.xunits = 'khz';      % Label the freq. units correctly
   s.yunits = 'dB';       % Plot the magnitude squared
   freqzplot(HR,FR,s);
   ylabel(axesRight,'Right Magnitude (dB)');
   hold off;
```

Figure 10: Function that plots the magnitude response (plotMagResp.m)

```
function playBtn_Callback(hObject, eventdata, handles)
   % hObject    handle to playBtn (see GCBO)
   % eventdata  reserved - to be defined in a future version of
MATLAB®
   % handles    structure with handles and user data (see
GUIDATA)
   azimuthVal = get(handles.azimuthList, 'String');
   azimuthIndex = get(handles.azimuthList, 'Value');

   if(get(handles.hrtfRad, 'Value'))
      HR = cof2vec(strcat('../Coeff
Files/H',cell2mat(azimuthVal(azimuthIndex)),'R.cof'));
      HL = cof2vec(strcat('../Coeff
Files/H',cell2mat(azimuthVal(azimuthIndex)),'L.cof'));
   else
      HR = cof2vec('../Coeff Files/AVG.cof');
      HL = cof2vec('../Coeff Files/DIFF.cof');
   end

   wavIndex = get(handles.wavListBox, 'Value');
   wav = 0;
   switch wavIndex
      case 1
         % Create 5 second noise sample for 48.8 KHz sampling
rate from 1 to -1 pp
         min = -1;
         max = 1;
         wav = min + (max-min) * rand(48800*5,1);
      case 2
         wav = wavread('../wav/queens.wav',488000);
      case 3
         wav = wavread('../wav/led.wav',488000);
      case 4
         wav = wavread('../wav/jesu.wav',488000);
   end

   % Filter hrtfRad's
   [FHL FHR] = hrtfFilter(HL,HR,wav);

   sound([FHL FHR],48800);
```

Figure 11: Function called when the play button is clicked (main.m)

```
function [FHL FHR] = hrtfFilter(hrtfL,hrtfR,wav)
   % Filter Left and Right channels
   FHL = filter(hrtfL,1,wav);
   FHR = filter(hrtfR,1,wav);
```

Figure 12: Function that filters a signal using the specified HRTF (hrtfFilter.m)

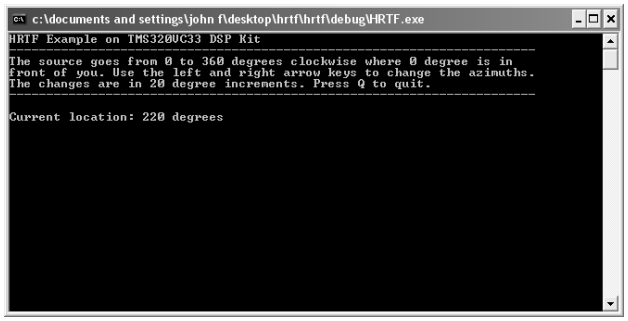

Figure 13: Screen shot of user interface for real-time implementation

The implementation of the HRTF in this system is similar to the software-based solution. The following sequence of code is located in the interrupt service routine of the assembly file:

```
FIR             FLOAT   R1,R1
                STF     R1,*AR4++%
                LDF     0.0,R1
                LDF     0.0,R2
                RPTS    LENGTH1-1
                MPYF3   *AR3--,*AR4++%,R1
||              ADDF3   R1,R2,R2
                ADDF3   R1,R2,R2
                FIX     R2,R1
                RETS
```

Figure 14: FIR subroutine is assembly file for TMS320C33 DSK (hrtf.asm)

This sequence of instructions implements equation 1 to convolve the input signal with the HRTF-based FIR filter. The complete set of programs needed to make the real-time 3D sound implementation run on the TMS320C33 DSK is available at http://dsplab.eng.fiu.edu/3DSND_DEMO/.

VII. Results

The use of both the Matlab® demonstration program and the real-time implementation on the TMS320C33 DSK has been used to successfully demonstrate 3D sound to engineering students. Overall, students have been able to perceive the intended virtual positioning of the sound being processed and its movement as the azimuth selection is changed.

In some instances, students listening to the spatialized sounds reported confusion on the location of the sound sources, in particular, front and back reversals. For example, a sound source that is filtered such that it should be perceived to be located at 20° azimuth would be perceived to be at 160° azimuth (refer to Figure 1 for spherical coordinate system). This is a common symptom of generalized HRTFs and is referred to as the "Cone of Confusion" (see Figure 15).

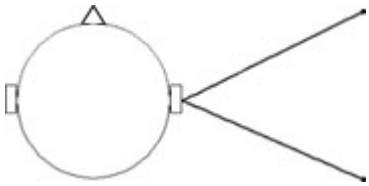

Figure 15: Cone of Confusion, top head view

The primary source of this phenomenon is the use of generalized HRTFs. The generalized HRTFs, as mentioned earlier, are measured using a KEMAR dummy head with microphones in it [2]. To maximize the accuracy of sound localization, the geometrical features of each individual user should be involved in the definition of the HRTFs used by the system when that particular user is listening to the spatialized sounds. Unfortunately, this requires specialized equipment that is not easily available to many of the potential users of the system.

VIII. Conclusion

This paper demonstrates HRTFs in both simulated and real-time environments. A detailed description is provided of the software and real-time based systems proposed to demonstrate 3D sound using Head Related Transfer Functions (HRTF). Fully commented versions of the coefficient files and Matlab® files for the software based implementation are available for download at http://dsplab.eng.fiu.edu/3DSND_DEMO/. This web site also contains assembler files, the DSK executable, the C source code, and the DOS executable for the real-time implementation on the TMS320C33 DSK.

We hope that the setup that has been described in this paper and the implementations freely available to students and educators will enable those interested to understand and experience 3D sound. Furthermore, these implementations can be used to demonstrate the practical value of many fundamental concepts of an introductory course of signals and systems or DSP, such as: transfer function, impulse response, convolution, frequency response, etc. In addition, it is hoped that witnessing these demonstration may entice students to pursue an interest in signal processing and, specifically, in sound spatialization and virtual 3D Sound.

IX. References

[1] CIPIC Interface Laboratory: Spatial Sound. [Online]. Available: http://interface.cipic.ucdavis.edu/

[2] Gardner, W. G., and Martin, K. D. (1994). HRTF measurements of a KEMAR dummy head microphone. MIT Media Lab Machine Listening Group. Available via the web at http://sound.media.mit.edu/KEMAR.html.

[3] Chassaing, R., "Digital Signal Processing: Laboratory Experiments Using C and the TMS320C31 DSK", Wiley Interscience, 1999.

[4] Matlab®, version 7 release 14, MathWorks Inc., http://www.mathworks.com.

[5] Kendall, G. S., "3-D sound primer: Directional hearing and stereo reproduction," Comput. Music J., vol. 19, pp. 23–46, Winter 1995.

[6] Barreto, A. B., Yen, K. K., and Aguilar, C. D., "PC-based DSP Training Station", Proc. ASEE 1998 Annual Conference, Seattle, WA, June1998. Session 1220: Digital Signal Processing, (CD-ROM format).

[7] Middlebrooks, J.C., Makous, J.C. and Green, D.M., "Directional Sensitivity of Sound-Pressure Levels in the Human Ear Canal," Journal of the Acoustical Society of America, vol. 86(1), pp. 89-108.

[8] Oppenheim, A. V. and R.W. Schafer. Discrete-Time Signal Processing, Englewood Cliffs, NJ: Prentice-Hall, 1989, pp. 311-312.

X. Acknowledgements

This work was partially sponsored by NSF grants IIS-0308155, HRD-0317692 and CNS-0426125.