An audio-video front-end for multimedia applications

Dmitry Zotkin, Ramani Duraiswami, Larry Davis
University of Maryland
College Park, MD 20742

Ismail Haritaoglu
IBM Research, Almaden
San Jose, CA 95120

Abstract

Applications such as video gaming, virtual reality, multimodal user interfaces, and video conferencing, require systems that can locate and track persons in a room through a combination of visual and audio cues, enhance the sound that they produce, and perform identification. We describe the development of a particular multimodal sensor fusion system that is portable, runs in real time and achieves these objectives. The system employs novel algorithms for acoustical source location, video-based person tracking, and overall system control, which are also described.

1 Introduction

Equipping machines with the ability to see and hear has been the stated goal of many researchers over the last few decades see (e.g., [1]). Only recently has the state of the art in the technology of acquiring information, and processing it advanced sufficiently to build practical systems that can begin to achieve this capability. Applications of such systems include user interfaces for computers, robots that sense and perceive their environment, perceptive spaces for applications in immersive virtual or augmented reality, etc.

We report on the development of a generic front-end for such applications that combines audio and video sensing and some perception. Our system is based on a combination of acoustical array(s) and a video camera equipped with pan tilt and zoom, attached to a personal computer. A suite of algorithms/software modules that perform acoustic source location, vision based person location, head position estimation, filtered beamforming, speech recognition, and person recognition have also been developed and/or implemented as part of the system. The system operates in real time, with the video operating at approximately 15 frames per second and the audio performing a corresponding number of source locations.

1.1 System Description

Our goal was for a relatively portable system that could combine the multiple requirements in one package. We thus decided to avoid highly specialized hardware but rather go with a PC with general purpose add-on boards. We are using a dual Pentium III-600 MHz system, Windows NT 4.0, and multithreaded programming to utilize both processors. All processing is done in real time with separate threads for the audio acquisition, video acquisition and camera control. The latter integrates information from audio and video trackers and uses different algorithms to control the cameras depending on the desired application (described below).

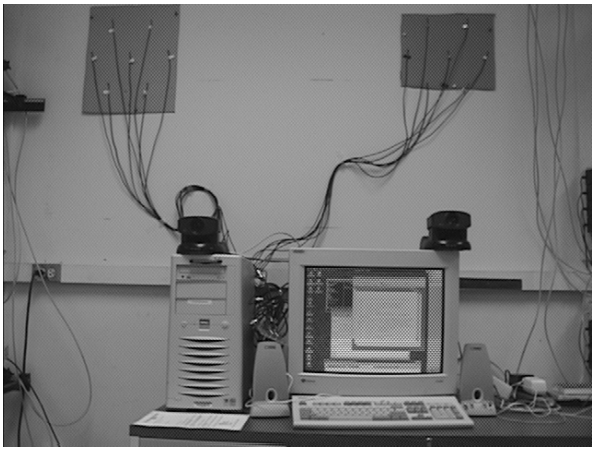

Figure 1: Experimental setup: two cameras and two arrays.

The audio capture subsystem consists of microphones arranged in a planar pattern, a set of preamplifiers and a data acquisition board. We have 14 speech-band button microphones divided into two subarrays. Within a subarray, six microphones are placed on the circumference of a circle and one microphone in the center. The diameter of a circle is approximately 30 cm. The data acquisition board used is capable of capturing 16 channels at 12 bit resolution. Each channel is sampled at 22.05 kHz. The board allows for software-controlled automatic gain control.

The video capturing subsystem is similar to the W^4 setup [3]. It consists of an active Sony EVID-30 video camera located near the microphone array. The camera can be controlled via a serial interface, allowing pan and tilt angles of the motorized head and zoom to be set [2]. The image is captured with a Matrox Meteor II frame grabbed using software based on the MIL-Lite toolkit. Capture can be done at full resolution and speed (640x480 and 30fps, respectively). However, in practice the framerate is controlled by the audio and video processing speed, and is lower ($\simeq$ 15 fps).

2 Audio Algorithms

The algorithms comprising the acoustical subsystem include noise estimation, time-delay computation, estimation of azimuth and elevation angles, estimation of the depth of the source, and final filtering to throw away spurious results due to noise or reverberation. Once the source location is estimated the sound can be beamformed to improve clarity and suppress noise. The beamformed sound can then be fed to a speech recognition system.

Time Delays: Our experimental environment was an extremely noisy and reverberant room with many sources of continuous sound. This circumstance made estimating time delays quite difficult. Noise from several computers (≃20), power supplies, and network switches formed a constant low frequency reverberant hum in the room. In the end we settled upon an algorithm described in [6].

Let the signal received at the i^{th} microphone be $x_i(t)$. The fast Fourier transform of x is $X_i(\omega) =$. The generalized cross-correlation function for the ith and jth signals, $R_{ij}(\tau)$ is given by

$$R_{ij}(\tau) = \text{IFFT}(W(\omega)X_i(\omega)X_j'(\omega))[\tau] \tag{1}$$

where IFFT denotes the inverse Fourier transform, and $W(\omega)$ is a weighting function. The time delay between signals received at the i^{th} and j^{th} microphones is given by $\arg\max_\tau R_{ij}(\tau)$. As described in [6], and assuming that the noise spectrum is the same for all channels in the system, the weighting function resulting in a maximum likelihood estimator is

$$W_{ML}(\omega) = |N(\omega)|^{-2} \tag{2}$$

Source Location: Determining the source coordinates from measured time differences is an almost classical problem arising in many different fields of signal processing. For a summary of the literature we refer the interested reader to the comprehensive review in [8]. In what follows we sketch a novel algorithm for source location. Further details may be found in [4].

We have N microphones located at $\mathbf{m}_i = (x_i, y_i, z_i)$, and a source at $\mathbf{s} = (x_s, y_s, z_s)$. The speed of sound is c, and distances between the microphones and the source are χ_i, with

$$\chi_i = \sqrt{(x_i - x_s)^2 + (y_i - y_s)^2 + (z_i - z_s)^2}. \tag{3}$$

The measured time delays between microphones i and j each provide a linear relationship of the form

$$\chi_i - \chi_j = ct_{ij}. \tag{4}$$

In general for N microphones there are $C(N, 2)$ measurements of which $N-1$ are independent. Various methods for solving the above system have been developed, including maximum likelihood estimators, and various interpolation based solutions [8].

In our solution we use the fact that the problem can be decomposed into two independent sub-problems. The first involves the measured time differences (4) which are potentially contaminated by errors due to multipath and reverberation, and in knowledge of the sound speed. This subsystem has rank $N-1$. We make the definition

$$\mathbf{d} = \left[\ \chi_2 - \chi_1, \quad \cdots, \quad \chi_N - \chi_1\ \right]^t. \tag{5}$$

so that the independent set that must be estimated from the noisy measurement can taken to be $\mathbf{d}$.

We can estimate $\mathbf{d}$ by solving the rank deficient problem by imposing hard constraints that impose $R_{max} > \chi_i \geq 0$, and also bound time delays, and incorporate knowledge of the expected imprecision in the measurements to throw out outliers. These constraints have the form

$$\chi_i - \chi_j > c_{\min}(t_{ij} - \epsilon), \quad \chi_i - \chi_j < c_{\max}(t_{ij} + \epsilon), \tag{6}$$

for $t_{ij} > 0$, and with similar equations for $t_{ij} < 0$. This set of equations (4) and (6) is solved using a constrained L_1 optimization algorithm (CL1) [12]. This gives a solution where the distance to the closest microphone is zero, i.e. we arrive at a constrained L_1 norm estimator for $\mathbf{d}$ in Equation (5) above using **all** the measurements, but excluding those measurements that violate constraints. Knowing $\mathbf{d}$, in the second stage of our solution, we estimate χ_1 and the coordinates using the procedure of Smith and Abel [9].

Often all we are interested in is the bearing angles for control of the camera, as the zoom can be determined heuristically. We can obtain the bearing angles approximately knowing $\mathbf{d}$. Using the distances between the microphone and the source we have

$$\begin{aligned} &-2\chi_1 z_i \cos\theta - 2\chi_1 x_i \sin\theta\cos\phi - 2\chi_1 y_i \sin\theta\sin\phi \\ &= 2\chi_1 d_i + d_i^2 - \left(x_i^2 + y_i^2 + z_i^2\right) \end{aligned} \tag{7}$$

Using the plane wave approximation ($d_i/\chi_1 \ll 1$) this equation can be written as

$$x_{i+1}A + y_{i+1}B + z_{i+1}C = -d_i \quad i = 1, \ldots, N-1, \tag{8}$$

where $A = \sin\theta\cos\phi, \qquad B = \sin\theta\sin\phi, \qquad C = \cos\theta$

and solved for the bearing angles, again using CL1.

For increased robustness, in our system we process data from two subarrays independently. (Remember that two groups of microphones are placed on the conference room wall with a horizontal separation of about 1.5 m). This gives source bearing angles from two viewpoints, which can be intersected to arrive at the three-dimensional coordinates of the source.

Clustering: Since estimates of the source location will be corrupted by noise, we filter the obtained estimates. Priority is given to signals of sufficient power that appear to

be of speech origin based on frequency content. Additional filtering is based on acceptance regions [7]. At any given time, the algorithm keeps track of last M valid position estimates $X_i = (R_i, \phi_i, \theta_i)$, $i = 1 \dots M$. These estimates are searched attempting to find a "cluster". In mathematical terms, for every i we compute

$$C_i = \sum_{j=1}^{M} 1(|X_i - X_j| < \varepsilon) \tag{9}$$

where ε is an constant which defines how much estimations can differ and still considered to be in the same cluster, and $1(\arg)$ is an indicator function which evaluates to 1 when $\arg$ is true and to 0 otherwise. If for all i $C_i < \beta M$, then there is no sound source present; otherwise, the measurement X_i such that C_i is maximal is taken as the position. β is an empirical constant which defines how many estimations should fall next to each other. Values of $M = 15$ and $\beta = 0.35$ work well.

Beamforming: The beamforming algorithm used for enhancing the S/N of a detected sound source is a simple delay-and-sum beamforming. Denoting $\hat{\tau} = \min_i \tau_i$, the beamformed signal can then be expressed as

$$\hat{x}(t) = \sum_i x(t + \tau_i - \hat{\tau}), \tag{10}$$

3 Vision Algorithms

We are primarily interested in video assisted audio tracking, i.e. when the audio data has priority over the video data. When sound is detected the camera is rotated to get the source in the field of view. During periods of silence, no sound data is available and the camera tracks the last sound source, rotating as it approaches the edge of the current field of view to keep it in the center. When a new source is detected, it becomes a center of attention for the system and it is tracked using video means. For video object localization and tracking, we have three possible subsystems, each of them is suited best for a specific operating conditions.

The initial video processing was based on [3]. The goals of the active tracking stage are to track people with an active camera in a large field of regard when their movement can take them outside of the current field of view. We wish, in addition, to have the ability to change the person tracked when the sound localization system determines that there is another talker in the scene. The basic video algorithms are background model creation from mosaic images; localization, segmentation and labeling of the foreground objects in the scene; motion detection and foreground object tracking using optical flow; and zooming to get a close-up of the head of the talker. Here we give only brief overview of the algorithms involved. Details can be found in [3].

The W^4 system first constructs the mosaic of the field of regard of camera by taking several images with different pan orientations. These images are then overlapped at reference points to produce a mosaic. A background model for the mosaic is computed and later used to detect foreign objects in the scene and to track them. Also, a motion detector is implemented by subtracting three sequential images and thresholding the pixel values in the difference image. The motion detector is used when the camera position differs from the reference positions used in mosaicking and thus a background model is not available. W^4 continuously tracks the position of the biggest foreground object in its field of view with a second-order motion model and predicts the object position in subsequent frames. When the next predicted location of the tracked object is outside of the current field of view, a new camera position is computed and the camera is rotated such that the object will be in the center of the current field of view. Note that there is no discrimination between human and non-human objects in the scene. W^4 uses only grayscale input so skin color information can not be used.

The second video processing subsystem is directed at detecting human faces. It uses the full-color input and consists of a several algorithms to perform skin color detection, human face pattern recognition, and camera control (pan/tilt/zoom) to keep the person tracked in the frame. First, the skin color is detected in the grabbed image using an improved skin color detection algorithm. This algorithm accounts for the nonlinearity of the CCD response and uses different reference ratios in skin color detection criteria in different intensity ranges. More specifically, let r, g, b be the red, green and blue intensities of a pixel. We compute the centers of acceptance regions for green-to-red ratio and for blue-to-red ratio, respectively, by linear interpolation on r on three points:

$$\begin{aligned} r &= 0.22 \Rightarrow gr_s = 0.80, br_s = 0.58 \\ r &= 0.68 \Rightarrow gr_s = 0.71, br_s = 0.57 \\ r &= 0.86 \Rightarrow gr_s = 0.83, br_s = 0.69. \end{aligned} \tag{11}$$

The skin color detection criteria set is as follows:

$$r_{min} < r < r_{max}, \tag{12}$$

$$gr_s - \epsilon < g/r < gr_s + \epsilon, \quad br_s - \epsilon < b/r < br_s + \epsilon, \tag{13}$$

and $\epsilon = 0.12, r_{min} = 0.1, r_{max} = 0.9$. All three inequalities should be satisfied for the pixel to be detected as a skin color pixel. The first inequality throws away pixels that are too dim or too bright since they are often detected as a skin color while they are not due to the non-linearity of the camera CCD. Of course, for different cameras different interpolations of gr_s, br_s on r and different values of r_{min}, r_{max} may need to be employed.

Then, the image consisting of a skin color pixels is divided into blocks of size 8x8 pixels. Every block is labeled white if the number of the skin color pixels in it is more than a half

of a number of a non-skin color pixels, and black otherwise. A connected components algorithm is then executed on the map consisting of these blocks. The resulting "skin color blobs" are treated as a starting points for a face detection algorithm.

The face detection algorithm is based on a simple template matching technique. The template matching score (correlation function) is computed for every pixel in a skin color blob in the image by shifting the template over the image. The pixel that gave rise to the maximum correlation value is a best match, and the match value is stored. To account for different sizes of faces in the image, the matching is done successively with variable template size, and the best normalized match (scaled by the area of template since the correlation is proportional to that) over all templates is taken as the best match for this blob. The template is pre-computed for several template sizes. In addition, the size of the skin color blob is used to bound the search by three or four template sizes that are most likely to match the skin color blob of that particular size.

Then, every skin color blob is tested if it is really an image of a face. A series of tests is executed; first, the normalized match value should be greater than some threshold, determined empirically. The maximum for the value is 1, and the threshold used in the system is 0.65. Then, let us denote by $\sigma(y)$ the following function:

$$\sigma(y) = \frac{1}{\hat{I}^2(y)} \sum_{x=x_c-b/4}^{x_c+b/4} (I(x,y) - \hat{I}(y))^2, \qquad (14)$$

$$\hat{I}(y) = \frac{2}{b} \sum_{x=x_c-b/4}^{x_c+b/4} I(x,y), \qquad (15)$$

where (x_c, y_c) is the center of the matched template, b is the template size and $I(x,y)$ is just the brightness of the pixel at (x,y). Thus, $\hat{I}(y)$ is the average intensity of a row of pixels of length $b/2$ centered at x_c and $\sigma(y)$ is the intensity deviation over the same row.

Since the template matching is often inexact and shifted a few pixels above or below the actual face image, the image area for

$$y = [y_c - b * 0.3, y_c + b * 0.1] \qquad (16)$$

is scanned for the row of pixels with the highest value of $\sigma(y)$. Let $\sigma(y_e) = \max_y \sigma(y)$. This row is likely to be where eyes of the person are located. Then, compute $y_1 = y_e - b*0.225, y_2 = y_e + b*0.125$. If the image in question is indeed the face, the row of pixels at y_1 is likely to the on the forehead of the person, and the row of pixels at y_2 would be below the eyes but above the lips. In both cases, one can expect that the average intensity of pixels $\hat{I}(y)$ at y_1 and y_2 is greater than the average intensity of pixels at y_s; furthermore, both $\sigma(y_1)$ and $\sigma(y_2)$ can be expected to be significantly less than $\sigma(y_s)$. Thus, if the following criteria set is satisfied, the algorithm concludes that the skin color blob is indeed the image of a human face:

$$\hat{I}(y_j) > \hat{I}(y_e), \quad \frac{\sigma(y_e)}{\sigma(y_j)} > 3, \quad j = 1,2 \qquad (17)$$

A simple lip area detection algorithm is also implemented. The assumption is that when person is speaking, his/her lips move, so the size of the lip area is changing. The lip area is mostly black on a skin color image, so the center of the lip area is first detected by matching a thin black fixed-size rectangle against the skin color image around the middle of the bottom half of a face. If the lip area size shows significant variations between a few successive video frames, the conclusion can be made that the person being imaged is now talking.

The third recently implemented localization and tracking subsystem uses an improved condensation tracker that combines a head model with a color histogram. This tracker is very good for robust tracking of heads in close to the camera situations such as videoconferencing or game playing and is described in [14, 13]).

4 Camera control algorithm

The camera control algorithm should take into account both audio and video information, which could be either consistent or contradictory. In case of contradictory information being supplied the priority could be either given to audio or video information or determined dynamically from previous activity. For the current implementation, the set of rules for the camera control are as follows.

If sound is detected and the source coordinates are determined to be valid, the camera turns in the direction of the source. In addition, the zoom of the camera is set proportional to the computed distance from the camera to the source (note that the source coordinates three-dimensional) so that the picture of the distant source is enlarged. Sound has priority over video for switching the camera to a new speaker. However, the camera doesn't move and doesn't change the zoom if the sound source is close to the center of a currently imaged area which includes a recognized facial pattern. In this case there is presumably no change in the speaker.

In absence of sound, the camera control algorithm keeps track of the face of the previous speaker and as long as it is recognized turns the camera to bring the face to the image center, and changes the zoom setting so that the size of the face template is reasonably large. If multiple faces are in view, the control algorithm can be adjusted to zoom on the main speaker only or to keep all of them in the field of regard. If no face pattern is found in the field of view for significant time, the camera is zoomed out in several steps in an attempt to re-acquire the target. One important issue encountered in the development is that the zooming should be limited to approximately half of the camera zoom range.

If the lens is zoomed past this point, usually the skin color picture becomes corrupted because of camera white balance auto-recalibration.

The current location that the camera is pointing at is passed to the beamforming algorithm to compute the appropriate set of delays for that point. Thus, when sound is detected, the audio system can instantly focus at the sound location. When the camera tracks the motion of a person, the algorithm updates the location of the sound source, keeping the beamformer focused on a speaker.

5 Software Description

We highlight some of the technical details of the developed real-time data processing software. The interface to the data acquisition board is implemented using the libraries provided by the manufacturers. A buffer of size 60 MB is allocated for the acoustic data and acquisition is started in the background. Samples are transmitted directly from the data acquisition board to the memory using DMA and PCI bus-mastering, bypassing the CPU, so that continuous acquisition does not load the main processor. Acquisition is automatically restarted when the buffer is exhausted (it takes about 70 seconds to fill up the buffer). Also, two threads are spawned that perform parallel FFTs and complex multiplication. The Fourier transform routine used is taken from the package *FFTW* [10].

The main processing loop repeatedly retrieves the current position of the acquisition stream via an interface call to the board library, copies and de-multiplexes data from the main buffer to internal FFT data storage, sends signals to the computational threads that data is available and waits for completion of parallel computations. Non-busy wait is used via the Windows NT events mechanism. The computational threads perform cross-correlations between different input channels. After the time delays are found, the position estimator and final position filter are called and the estimated position can be retrieved. This position is used by the camera control thread if the system is used for active audio tracking. There is no strict timing in the main program loop – as soon as one iteration is completed the next one begins operating on the new and latest frame of data. Thus, some data may be skipped if the main loop takes too long. We give priority to audio processing, and the video frame rate is adjusted on-the-fly so that no audio data is lost.

Interface to an automaitc speech recognition (ASR) engine (the OGI CLSU toolkit [11]) is implemented via TCP sockets. The ASR engine comes with the a TCL programming shell with speech recognition commands built-in. Currently, we use a limited-vocabulary recognizer, and the results are simply printed on the screen in the TCL shell window. The beamformed wave is accumulated in the special buffer; it is downsampled it on-the-fly to 8 kHz as required by the ASR engine. When the speech segment is over, the data is sent to the recognizer using pre-established TCP connection. The beginning and the end of the speech are detected using a power threshold in the speech frequency band.

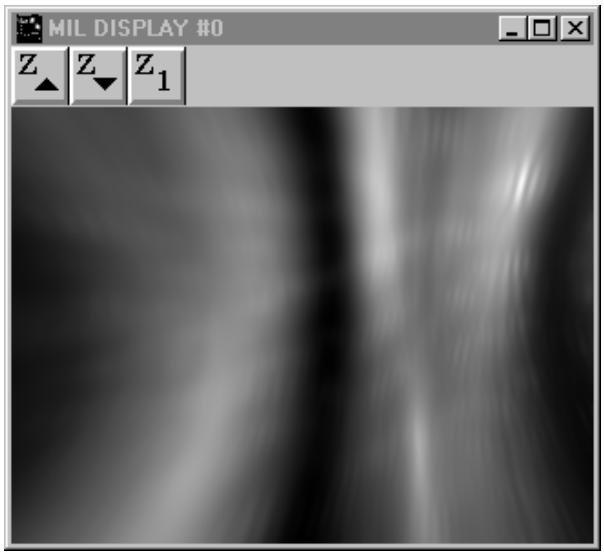

Figure 2: Sample energy map (4 sources)

6 Performance

Accuracy of the audio localization is limited by the audio sampling frequency and the fact that the estimations of time delays between channels are integer values. The typical intermicrophone distance is about 0.2 meters, which corresponds to the delay of approximately 13 samples at the sampling rate of 22.05 kHz. The average error in bearing estimation (determined by simulation) is limited by approximately 0.02 radians, which is a very good accuracy for the typical usage scenario. As for video data processing, the face template matching performed well for several people that tested the system; the system also does not misrecognize images of hands or skin color patterns on other objects as face templates. The only problem encountered is that if the forehead of a person is covered by hair, the detection does not perform well since the skin color region is assumed to be elliptical in shape.

The speed of the audio data processing alone is approximately 33 source position estimations per second. This time is mainly taken up by 7 direct FFTs, $C_2^7 = 21$ complex multiplications and 21 inverse FFTs per subarray. This performance corresponds to the data buffer size of 2048 samples, which is approximately 92 ms worth of data. The reaction time of the system in case video processing disabled is also quite fast. Four or five subsequent well-clustered source location estimations are sufficient to robustly detect sound source presence, which is equivalent to 120-150 ms. The multiple source detection algorithm is significantly slower; it takes about a second to create and resolve a multiple-source spatial energy map. A sample map for four sources

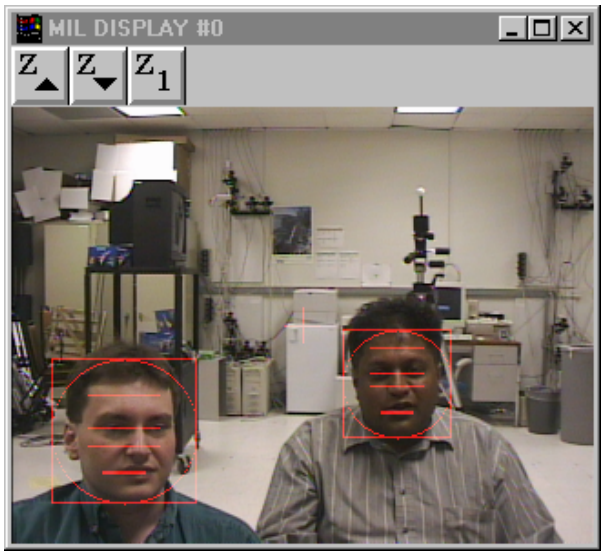

Figure 3: Multiple talker identification

is shown in Figure 2; the brightness of a pixel corresponds to the energy in the signal beamformed for this location. The resolution of map is increased here; the map used in on-line processing is much coarser.

When the video processing is added to the system, the processing rate is reduced. The decrease depends on the complexity of the picture acquired by the camera. The typical frame rate when there is one person in the frame is approximately 15 frames per second. (The audio localization is also done at the same rate). The screenshot of a processed video subsystem image is shown in Figure 3. The circular mark denotes a match for the face template. The square mark means that the skin color blob in question is indeed identified as a face. Also, three horizontal lines across the face image show, respectively, the forehead, the eyes and the nose. The thin rectangle in the bottom part of an image is the detected lip area. The camera in this frame was switching between persons in the frame as one of them stopped and the other started speaking.

We performed some experiments to evaluate the system speech enhancement performance. In one experiment, we used an electric drill sound as noise and increased the sound level until a sharp drop in ASR performance occurs for the beamformed and plain sounds. According to this experiment, the array gain is about 7 dB. In another experiment, the setup included two sound sources, one of which was a computer speaker playing music and the other is a human speaking digits. The sound source positions were determined by the system and two waveforms were computed, one with the beamformer focused on the first source and the other one for the second. The difference is obvious on the waveform picture and can be heard clearly while playing these sounds. A rough estimation of the gain was also done by using the sound power indicator of the wave player software; it is also approximately 7 dB.

7 Conclusions

This work could be used as the basis for a general user interface for a freely moving user in a room equipped with the present system. The user could then control a remote computer using a combination of voiced and gestured commands. Further improvements and enhancements of the system are ongoing.

Acknowledgments

Support from the W.M. Keck Foundation, ATR MIC Labs, and ONR via Contract N00014951021 is gratefully acknowledged.

References

[1] E.E. David Jr., O. G. Selfridge (1962). “Eyes and ears for computers,” Proc. IRE, Vol. 50, 1093-1101.

[2] R. Ismail Hariatoglu (1999), *Real-time systems for detecting and tracking people and monitoring their activities* Ph.D. thesis, Department of Computer Science, University of Maryland, College Park.

[3] I.Haritaoglu, D.Harwood, L.Davis, (1998) “W4: Who, When, Where, What: A Real Time System for Detecting and Tracking People”, Proc. 3rd Intl. Conf. on Automatic Face and Gesture Recognition, Nara, Japan.

[4] R. Duraiswami, D. Zotkin, L.S. Davis (1999), Exact solutions for the problem of source location from measured time differences of arrival. J. Acoust. Soc. Am., Vol. 106, No. 4(2), 2277.

[5] R. Duraiswami, D. Zotkin, E. A. Borovikov, L.S. Davis (2000), Active source location and beamforming. J. Acoust. Soc. Am., Vol. 108, No. 8(6), 2890.

[6] D.Feitelson, A.Weil (1996). ”A robust method for speech signal time-delay estimation in reverberant rooms”, *Proc. ICASSP-96, Atlanta, GA.*

[7] D.Sturim, M.Brandstein, H.Silverman. (1997) “Tracking multiple talkers using microphone-array measurements”, *Proc. ICASSP-97, $Munich$,Germany.*

[8] M.S. Brandstein (1995), *A Framework for Speech Source Localization Using Sensor Arrays*, Ph.D. thesis, Brown University.

[9] J.Smith, J.Abel, (1987) “Closed-form least-squares source location estimation from range-difference measurements”, IEEE Trans. on Acoustics, Speech and Signal Processing, ASSP-35(12), 1661-1669.

[10] Information on FFTW is at http://www.fftw.org/.

[11] The Oregon Graduate Institute Center for Spoken Language Understanding Toolkit information is available at http://cslu.cse.ogi.edu/toolkit/.

[12] I. Barrodale and F.D.K. Roberts, (1973) “An improved algorithm for discrete l_1 linear approximation,” *SIAM J. Numer. Anal.*, **10**, 839-848,

[13] V. Philomin, R. Duraiswami, L.S. Davis (2000). Quasi-random sampling for Condensation. Proceedings of the European Conference of Computer Vision, Dublin, Ireland 2000.

[14] D. Zotkin, R. Duraiswami, V. Philomin, L. S. Davis (2000). Smart videoconferencing, in Proc. IEEE International Conference on Multimedia and Expo, New York City, NY August 2000.