Synchronizing Multimodal Data Streams Acquired Using Commodity Hardware

Martial Michel, Ph.D.
Systems Plus, Inc.
One Research Court - Suite 360
Rockville, MD 20850, USA
and NIST
martial.michel@nist.gov

Vincent Stanford
National Institute of Standards and Technology (NIST)
100 Bureau Dr - MS 8940
Gaithersburg, MD 20899, USA
vincent.stanford@nist.gov

ABSTRACT

We have developed tools and techniques that allow video frame level synchronization of multiple free-running commodity video cameras, microphones, and computer nodes using non-realtime operating systems. The techniques rely on physical audiovisual synchronization pulses, statistical procedures to correlate and interpolate the multiple timestamp streams, and software tools for review to produce smoothed and drift-corrected timestamp streams in our multimodal corpora.

In this article we present those techniques and tools. Our project is open source and we are seeking collaborative developers for future work.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*; E.5 [**Files**]: Organization/strucutre; G.3 [**Probability and Statistics**]: Correlation and regression analysis; H.3 [**Information Systems and Retrieval**]: Miscellaneous

General Terms

Algorithms

Keywords

Audio/Video Synchronization; Data Streams; Timestamps; Commodity Hardware

1. INTRODUCTION

Synchronizing multimodal data streams captured using non-realtime systems is a complex task. There is clock drift in the hardware itself, and also a non regular interval between successive timestamps (jitter) due to the non realtime schedulers on the systems capturing the data. Some sources of this jitter include disk writes, DMA, context switching, and other sources of refractory delays.

Within the National Institute of Standards and Technology[1] Smart Space Project[2], in a collaboration with the Automatic Meeting Recognition Project[3], we have developed a set of tools and techniques that enables us, when recording multiple firewire cameras and microphones directly to disk, to resynchronize all sources to within one video frame of one another. This technique relies on pre and post recording audiovisual synchronization pulses, statistical correction of jitter, and the use of a set of internally developed visually oriented software tools.

In this article, we will first introduce the Smart Space Project as well as the Automatic Meeting Recognition Project, and then describe key technical details of our multimodal resynchronization tools and techniques.

2. NIST SMART DATA FLOW SYSTEM

The NIST Smart Data Flow System was designed to facilitate research in the field of pervasive computing and smart spaces by facilitating management of numerous sensors and computer nodes connected in a pervasive network infrastructure. It focuses on advanced human/computer interaction; integrated multimodal sensor networks; data flow discovery; and sensor-fusion based context-aware interfaces that can:

- Identify speakers, gestures, or persons in the video
- Transcribe and respond to what users say
- Implement realtime security mechanisms based on continuing identification of users
- Protect privacy of vital information with realtime security
- Implement accessible work spaces for users with special needs
- Recognize objects in video streams of the environment

In order to support the development of the necessary large-scale NIST standard reference data sets, the integration of many complex recognition algorithms, and sensor-fusion processing, we developed a basic infrastructure layer for multimodal laboratory data collection and experimentation[4, 5, 6]. To serve these goals we developed abstract data transport mechanisms for:

- Interconnecting sensors and systems to explore issues in distributed sensor processing

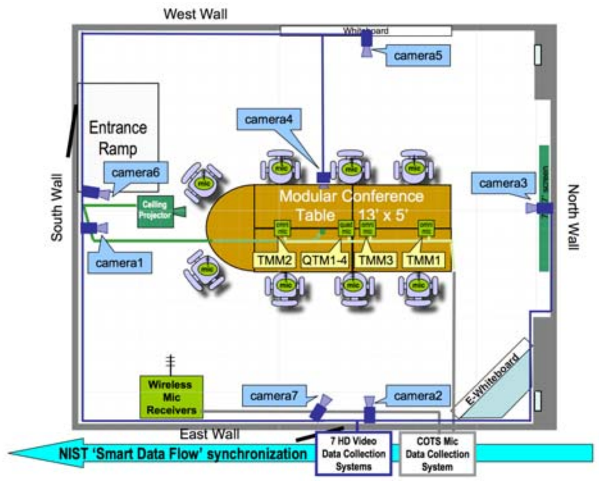


Figure 1: Plan view of the Corpus 02 Meeting Room Data Collection Laboratory

- Establishing a multimodal interface test-bed for accessible work spaces
- Providing a multi-sensor data recording environment

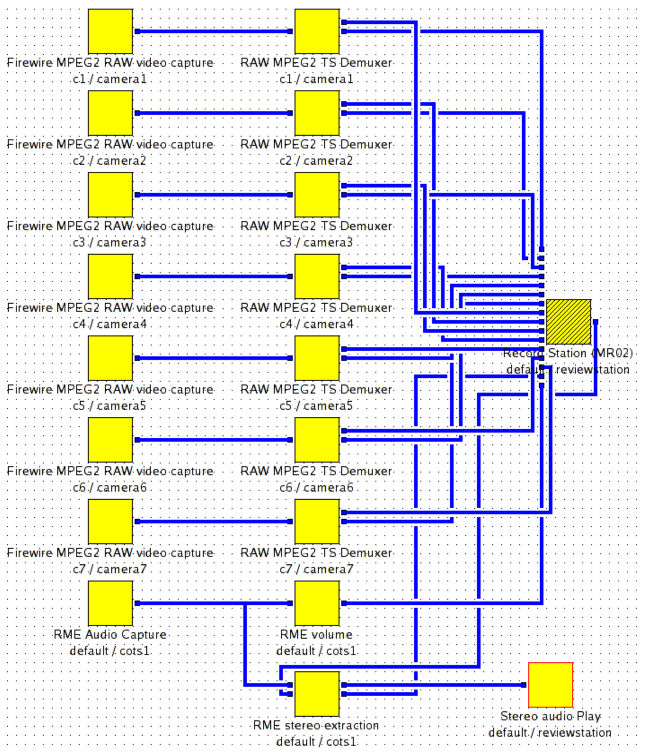
Therefore, in order to ease the development of these complex and highly integrated sensor-rich environments, we defined a system for distributed processing and network transfer of high bandwidth flows between nodes interconnected by a server crossbar.

The NIST Smart Data Flow System, first deployed in 1999, was tightly bound to the Linux operating system because of its open source code base and high performance networking capabilities. At that time it supported over four times more network data throughput than other available operating system technologies. It is the core of our data collection system used in sensor capture, command, control, and multimodal data review, for the NIST Automatic Meeting Recognition Project.

The Version 2 Smart Flow System is currently in final test and will be released later in 2006. It is much higher performance, and is written to portable interfaces that can be compiled on Linux, OS X, and Microsoft Windows. We are testing on OS X, and Linux, but not yet on MS Windows. We are interested in development partners who wish to port the system to the Win32 platforms and will work with them to help resolve any remaining architectural issues. Please contact the authors to discuss such an undertaking.

3. NIST AUTOMATIC MEETING RECOGNITION PROJECT

Significant research efforts have been undertaken, or are ongoing, in mining information from newswire services, news broadcasts, and conversational speech; and also in accessing metadata extracted in these domains. However, less focus has been placed upon the more challenging and equally important meeting domain which includes many subdomains such as: judicial proceedings, legislative proceedings, lectures, seminars, board meetings, and other less formal meetings. All of these could benefit from automatic recognition,



Firewire MPEG2 RAW video capture c1 / camera1
RAW MPEG2 TS Demuxer c1 / camera1
Firewire MPEG2 RAW video capture c2 / camera2
RAW MPEG2 TS Demuxer c2 / camera2
Firewire MPEG2 RAW video capture c3 / camera3
RAW MPEG2 TS Demuxer c3 / camera3
Firewire MPEG2 RAW video capture c4 / camera4
RAW MPEG2 TS Demuxer c4 / camera4
Record Station (MR02) default / reviewstation
Firewire MPEG2 RAW video capture c5 / camera5
RAW MPEG2 TS Demuxer c5 / camera5
Firewire MPEG2 RAW video capture c6 / camera6
RAW MPEG2 TS Demuxer c6 / camera6
Firewire MPEG2 RAW video capture c7 / camera7
RAW MPEG2 TS Demuxer c7 / camera7
RME Audio Capture default / cots1
RME volume default / cots1
RME stereo extraction default / cots1
Stereo audio Play default / reviewstation

Figure 2: NIST Meeting Room Data Collection Laboratory Capture Map showing the client nodes for camera capture, MPEG-2 demultiplexing, sound capture, and recording

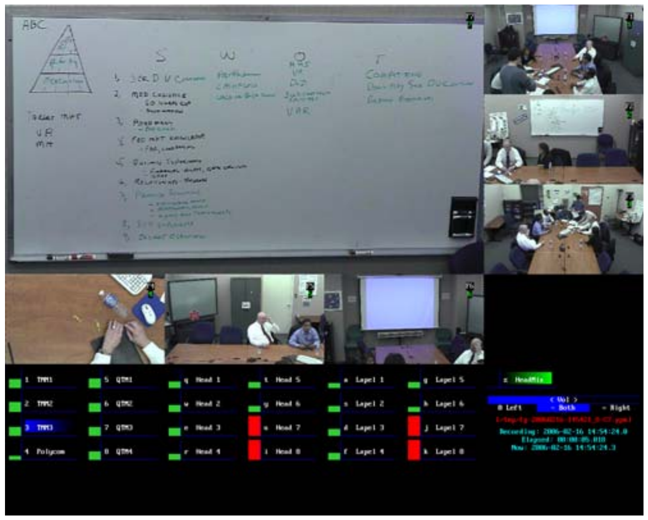


Figure 3: View of the recording operator's Capture and Replay station

understanding, summarization, and information extraction technologies, linked to online information retrieval systems. Currently, such technologies exist only as early research prototypes and require substantial continuing development.

NIST, through its Automatic Meeting Recognition Project (also referred to as the *Meeting Room Project*) has begun to make it possible for researchers to access data and metadata associated with a variety meetings; allowing them to work on understanding the logistics of meetings, the human-to-human interaction involved, and other detailed speech and video analyzes. The project supports the development of audio and video recognition technologies in the context of human meetings through an extensive data collection effort[7, 8, 9].

The Meeting Room Data Collection Laboratory allows us to record people in a common meeting environment using a whiteboard, a large-screen Windows desktop, a computer projector, and audio-conference equipment. In addition to these meeting room presentation aids, there are seven high definition cameras, six with fixed view and focus, and a seventh with auto-focus and manual pan-tilt to follow the presenter, as well as a Commercial Off The Shelf (COTS) recording system enabling us to collect up to twenty-four synchronized microphone channels. Seven of these channels are tabletop microphones, comprised of three omni-directional microphones, and a 4-channel unit with a directional microphone pointed in each compass direction. We also have a multiplexed tap into an audio teleconference line. Finally, there are 16 wireless microphones, used as either 8 head-mounted and 8 lapel-mounted, or as 16 head-mounted microphones for the meeting participants. Figure 1 presents the current layout of the room.

The Smart Data Flow map for Meeting Room capture is shown in Figure 2. The boxes represent client nodes, and links between boxes the data flows. The servers (sfd) are not represented but run one on each distributed system. The capture map features:

- Seven MPEG-2 capture client nodes, each reading from its Firewire video camera interface
- Seven MPEG-2 demultiplexers that split audio and video, and export MPEG-2 Elementary Streams (ES) to output flows of MPEG-2 I frames, and of all I, P, and B frames
- A COTS capture client to compute and display channel volume meters, as well as listen to specific channels being recorded
- The record-station client that displays the seven camera views and a volume meter for each audio channel; all under operator control

Each camera data stream is recorded to disk on its associated capture computer, at 1280 x 720, 30 fps in MPEG-2 ES, with one MPEG-2 Group Of Pictures (GOP) per 6 frames at 18Mbps, which uses about 8GB per hour. The 24 microphones collected by the COTS system are always recorded, at 48kHz and 24bits per channel, taking over 12GB per hour. Data is recorded on each of those systems using the NIST Smart Data Flow System, in special file formats (SMD) containing each individual video and audio frame, as well as Unix Epoch timestamps, as given either by the capture device driver, or capture software.

While this data is collected at each individual capture system, it is also processed for display on the review station that enables the operator to view all cameras and check the level of each captured microphone. A view of the recording station during an actual meeting as seen by the recording operator can be seen in Figure 3.

4. SYNCHRONIZING MULTIMODAL DATA

4.1 A practical solution based on NTP, and statistical interpolation

For cost and reliability reasons we elected to use commodity hardware and non-realtime operating systems to record data from multiple devices, each interfaced to a free running commodity computer. This presents some problems in high-resolution data synchronization across the channels. One problem is to assure that all capture systems use the same external clock information to limit, as much as possible, drift between the clocks on the capture systems. We found that uncorrected system clocks on commodity motherboards could drift a minute or more in a single twenty-four hour period. To help limit the extent of the problem we are using the Network Time Protocol (NTP)[10] as an external clock source to stabilize the commodity clocks in our data collection network.

NTP concepts rely on the fact that time synchronization to a local source can be achieved by synchronizing to multiple external clocks repeatedly and estimating the network delays statistically. NTP needs a reference clock that defines the true time to operate, and all clocks are set towards that true time. Even when a network connection is temporarily unavailable, NTP can use measurements from the past to estimate current time and error. NTP will also maintain estimates for the accuracy of the local time, and maintain its synchronization to the master server. If the system has been running for about an hour, it can be considered stable, and will be able to avoid setting the clock to values in the past. It will adjust the way the computer sees time to resynchronize to the master clock, making sure no major synchronization issues can occur.

We are using NTP with a master node as the master clock located on the same physical network switch as the capture nodes to propagate time synchronization information as quickly as possible. This insures that all capture nodes are and remain within a few milliseconds of one another during capture.

4.2 The audio visual slate

In order to provide a common reference point in each individual camera and individual microphone channel, we coupled a movie slate *clap*, that generates an acoustic impulse from an impact, to the electrical contact of a photographic strobe *flash*, which triggers when the slate clap bars hit. This gives us a multimodal event:

- on the audio channels, the clap sound is easy to spot by displaying the audio waveform.
- on the video channel, the flash produces a light burst that is visible in only one video frame.

In our new recording settings, since we are recording at 1/60 of a second, it is possible for the flash not to appear in a video frame that corresponds to a *flash frame*, therefore we

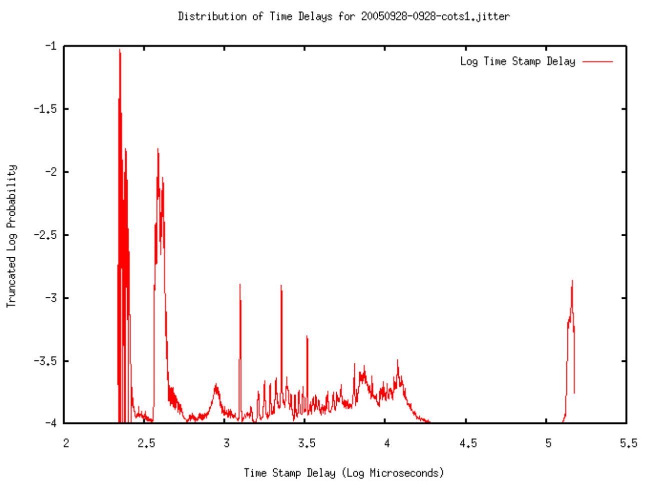


Figure 4: Distribution of timestamp jitter (log-log scale) for a COTS twenty-four microphone bank. The range spans three orders of magnitude, and shows a complex structure.

chose to do three flashes per direction seen by a camera; i.e. one camera on each wall and the ceiling mounted camera.

4.3 Using a capture device block index

The capture clients are designed to record data and the timestamp associated to each frame captured. For video, this means that each video frame should be $\frac{1}{29.97} = 33.366ms$ apart. Unfortunately, most off-the-shelf recording devices do not provide internal timestamp information. In addition, the computers used to capture the data obtained from the recording devices are simple PCs running Linux without a real time kernel. The timestamps recorded by those computers are assigned by a call to the operating system's internal time querying function after the data has been received and made available to the recording program. Successive recorded timestamps show a large random jitter on the expected interframe Δt. This jitter can be explained by several sources (in addition to the ones listed above), including: scheduler delays caused by varying load on the system, local system clock drift, disk operation, uninterruptible kernel time, and particularly for the MPEG-2 cameras: data compression time. This gives us a delay between frames that can be quite irregular, but which is fairly accurate in the long term owing to the properties of the NTP system. To regularize the individual timestamps we correct them post-capture using a simple statistical procedure.

Figure 4 shows an example of the frequency distribution of the timestamp jitter during a COTS capture session. The irregularity stems from the fact that we can only timestamp the data when we get a data buffer from the Linux kernel. The majority of the timestamps are under the average value as the kernel masks other interrupts during disk input/output processing, so data builds up in the capture device internal buffers, and the system processes them as fast as it can during other periods.

The accuracy of the long-term average of the interframe Δt suggests that simple statistical procedures might be used to interpolate the very noisy, but relatively unbiased time estimate represented by the individual timestamps. However, simply averaging the Δt values over time will not necessarily be robust with respect to loss of blocks. We therefore use a simple linear regression procedure against the block index. Thus our model of time progression is:

$$\hat{t}_n = \hat{\Delta} \cdot \iota_n + \hat{\tau}$$

Where ι_n is the n^{th} block index and is not necessarily consecutive; e.g.: there may be gaps in the sequence if blocks were lost. The parameters Δ and τ are estimated by the elementary expedient of minimizing the sum of squares:

$$S(\Delta, \tau) = \sum_{1 \le n \le N} \left[t_n - (\Delta \cdot \iota_n + \tau)\right]^2$$

This is simply done by differentiating with respect to Δ and τ, and setting the result equal to zero, as:

$$\frac{\delta S}{\delta \Delta} = -2 \sum_{1 \le n \le N} \iota_n \cdot (t_n - (\Delta \cdot \iota_n + \tau)) = 0$$

and

$$\frac{\delta S}{\delta \tau} = -2 \sum_{1 \le n \le N} (t_n - (\Delta \cdot \iota_n + \tau)) = 0$$

So that:

$$\hat{\Delta} = \frac{\sum_{1 \le n \le N} (\iota_n - \bar{\iota})(t_n - \bar{t})}{\sum_{1 \le n \le N} (\iota_n - \bar{\iota})^2}$$

and

$$\hat{\tau} = \bar{t} - \hat{\Delta} \cdot \bar{\iota}$$

Where t_n is the n^{th} timestamp obtained from the kernel with its highly variable additive jitter noise. This allows a practical timestamp correction procedure that performs well across the time spans involved in normal meetings. We are currently developing a recursive least squares technique using the Widrow FIR filter estimation technique[11] so as to allow online correction of the timestamps and will report on that at a future date.

As a basis for the linear regression, we need to have a list of all the observed timestamps for a given capture device. Where available we also need block numbers for the data frames. If the particular hardware device does not provide a block or frame number, we must synthesize one from the frame count. The SMD files contain the actual data chunk and the timestamp information for each collected data frames. When required, another file format, using the SMM extension collects metadata information related to each collected frames (such as the the block number or frames type). It is very resource intensive and time consuming to work with all the data contained within SMD files, therefore we introduced the use of a separate data chunk index file (usualy refered to as the "index" files), designated by the SMI extension, containing only the timestamp and location in the SMD file of the corresponding data. The index file for an 8GB video file is less than 2MB, so we can manipulate it easily.

4.4 Jitter fix example

We apply the linear regression to the timestamp from the original index for each capture device to the block timestamps, and this gives us a new index containing not only jitter corrected timestamps, but also corrected start and end

INSERT **1** Example program output for a timestamp correction regression run

```
Initializing: Index check, memory allocation, ...
Checking: 20050928-0928-cots1.smi

Preprocessing, Part 1: From source index file
Analyzing: 20050928-0928-cots1.smi

Summary:
Beginning timestamp:  1,127,914,091,169,480,000 ns
End timestamp:        1,127,919,986,582,873,000 ns
Total Length:         5,895,413,393,000 ns
Seen buffer:          1,768,549
Calculated jitter:    3,333,474 ns

Preprocessing, Part 2: From memory
Method: Linear regression by method of least squares
  with x_mean = 884,274.500,000
     and y_mean = 2,947,871,028,514.230,957
  with 'y = a + b * x' knowing that 0 <= x < 1,768,550
  where a = 268,018,824.381,486 ns
     (value added to the first timestamp)
  where b = 3,333,357.469,530 ns
     (delay between two consecutive timestamps)
  and real_y = beg_ts + y
Expected beg ts:           1,127,914,091,437,498,752 ns
Expected end ts:           1,127,919,986,643,518,080 ns

Processing to Disk
File: 20050928-0928-cots1.smi
First timestamp written:   1,127,914,091,437,498,752 ns
Last timestamp written:    1,127,919,986,643,518,080 ns
```

timestamps that enhance the re-linearization of the data. Insert 1 shows the actual process involved in correcting the index file whose jitter distribution was shown in Figure 4.

If all capture devices had no internal delays following this step, provided no loss of blocks, the captured data would be perfectly synchronized. Unfortunately, hardware capture devices have various internal delays for which we must compensate. For example, the video cameras have an internal audio/video compression delay of approximately 350ms.

4.5 Finding the audiovisual synchronization points

In order to find the point of synchronization between audio and video channels we use a modified video production slate with a strobe flash attached that is triggered by the arm used to make the clap sound. This gives us a stable reference point that is simultaneous in both acoustic and video domains. To use this information, we have to find both the audio clap in the audio signal and the flash in a single frame of the video signal, and align the two signal streams. To this end we developed a resynchronization tool that displays the video frame and the corresponding audio waveform for the specified audio and video channels. This software uses indices from the video and audio SMD files, as a frame-by-frame capable viewer and can jump in time as well as change the delay between the audio and video timestamps.

Knowing that audio data is collected at 48Khz, and written to disk with a timestamp every 160 samples, recorded audio frame are 3.333ms apart. Because we only have one audio source and the audio frames are under 4ms in duration, we have about 10 audio frames per video frame. We synchronize all cameras to the microphone located at the center of the conference table –the microphone which is closest to the source of the clap sound. The synchronization

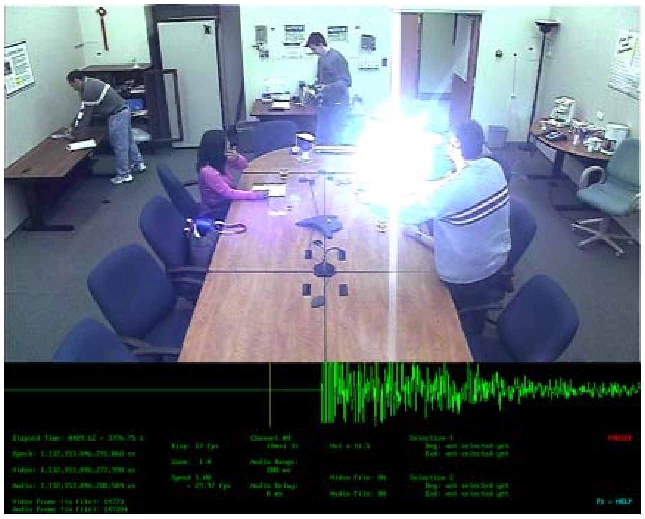


Figure 5: Resynchronization tool showing the video *flash* and audio *clap* from indices after resynchronization process. The Synchronization tool allows the operator to determine delay between audio and video by allowing the audio delay to be varied until the clap impulse is visible in the flash frame

process then proceeds in four steps:

- Locate the acoustic claps in the audio, at the beginning and end of the recording.
- Locate the frames with visible strobe flashes in each video camera recording.
- Change the delay of the audio channel until the onset of the clap impulse synchronizes with the flash frame.
- Average the delay on multiple leading flashes, and validate by checking that the value does not differ by more than one frame for slate claps at the end of the recording.

Once the delay has been found and verified using the trailing flashes, time-synchronized indices are created applying the device-specific delays to their respective index files. Figure 5 shows the audio/video synchronization tool.

4.6 Trimming the Indices

Once the audio and the video are synchronized, we replay the synchronized audio and video at the beginning and end of the meeting, and select the logical beginning and end times. Those times gives us our "trim" times. This step is not required for the synchronization process. It is done to make it easier for an operator to review a recorded meeting using the meeting's real start and end times, without including the pre- and post-recording operations. The data contained within this subset is the core meeting information, and is necessary for the extraction of data prior to distribution.

Trimmed indices, also represented as SMI files, are the result of this process. They contain all the index entries from the original index file (timestamp and data chunk location inside the SMD file), for a subset of the original index file, here, from the selected meeting beginning timestamp to the selected meeting end timestamp. Trimmed indices allow the use of specific parts of the full meeting without having to create new multiple gigabyte files. They are very useful for generating the final distribution corpora. The tools used to extract the content of the raw audio and video data go through the distribution pipeline using only the data content specified by the trimmed indices. Our work in this area is an open source project and we invite the interested reader to contact us about collaborative development of additional tools, or simply to obtain the tools we have described.

5. CONCLUSIONS

We have described a set of tools and techniques that was developed to facilitate post-recording synchronization of multimodal data on non-realtime operating systems and commodity video and audio hardware. From audiovisual synchronization pulses, using linear regression techniques, and performing hardware delay correction, we offer a complete method to resynchronize this type of data. The resulting re-synchronization is accurate to within 33ms (ie 1 video frame).

The resynchronization tool presented was developed using the SMD file format. SMD files record one timestamp per data frame, which allows us not only to re-linearize captured data, but also when using the frame-per-frame resynchronization tool to be able to match an audio clap with a video flash.

We are investigating means of replacing the flash with a series of timed LEDs, which would give us sub frame information and allow for an even more precise synchronization.

In the future, we will be including both the "block" count information in case of data loss, and also will introduce an on-the-fly smoothing of timestamps, where jitter should become very minimal within minutes of starting a recording.

We also intend to record the table's center microphone on the audio channel of each of the cameras. Using the MPEG-2 *Presentation Time Stamp* information, it is possible to have an accurate audio video synchronization within the MPEG-2 stream. Then, comparing the extracted MPEG-2 audio to the recorded high quality source, we will be able to automatically isolate the audio claps, and find the corresponding video frame.

6. DISCLAIMER & LICENSE STATEMENTS

7. REFERENCES

[1] National Institute of Standards and Technology. `http://www.nist.gov/`

[2] Smart Space Project. `http://www.nist.gov/smartspace/`

[3] Automatic Meeting Recognition Project. `http://www.nist.gov/speech/test_beds/mr_proj/`

[4] M. Michel, V. Stanford and O. Galibert (2005). Network Transfer of Control Data: An Application of the NIST Smart Data Flow. Proceedings of CCCT 2003. Extended version published in 2005 in the Journal of Systemics, Cybernetics and Informatics (Volume 2, Number 6).

[5] V. Stanford, J. Garofolo, O. Galibert, M. Michel and Christophe Laprun (2003). The NIST Smart Space and Meeting Room projects: Signals, Acquisition, Annotation, and Metrics. Proceedings of ICASSP 2003.

[6] R. Xu, G. Mei, Z. Ren, C. Kwan, J. Aube and C. Rochet and V. Stanford (2006). Towards User Sensitive Interfaces: Autodirective Speech Acquisition for Speaker Identification and Speech Recognition Using Phased Arrays. Springer Lecture Notes On Computer Science AI Subseries, 2006 Yang Cai, Ph.D. Ed.

[7] J. Garofolo, C. Laprun, M. Michel, V. Stanford and Elham Tabassi (2004). The NIST Meeting Room Pilot Corpus. International Conference on Language Resources and Evaluation (LREC'04)'s Speech Corpora and Annotation/Processing Tools.

[8] J. Garofolo, M. Michel, V. Stanford, E. Tabassi, J. Fiscus, C. Laprun, N. Pratz and J. Lard (2004). NIST Meeting Pilot Corpus Speech (ISBN 1-58563-302-x). `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2004S09`

[9] J. Garofolo, M. Michel, V. Stanford, E. Tabassi, J. Fiscus, C. Laprun, N. Pratz, J. Lard and S. Strassel (2004). NIST Meeting Pilot Corpus Transcripts and Metadata (ISBN 1-58563-303-8). `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2004T13`

[10] Network Time Protocol. `http://www.ntp.org/`

[11] B. Widrow and E. Walach (1996). Adaptive Inverse Control. Englewood Cliffs, NJ: Prentice-Hall.

[12] Rich Transcription 2002 Meeting Recognition Evaluation, documentation. `http://www.nist.gov/speech/tests/rt/rt2002/`

[13] Rich Transcription 2002 STT and Metadata Extraction results, presentations, RT-02 Workshop. `http://www.nist.gov/speech/tests/rt/rt2002/presentations/index.htm`

[14] Rich Transcription 2004 Spring Meeting Recognition Evaluation, documentation. `http://www.nist.gov/speech/tests/rt/rt2004/spring/`

[15] Systems Plus, Inc.. `http://www.sysplus.com/`