

**Facial Tracking and Animation**  
**3<sup>rd</sup> Progress Report**

**Todd Belote**  
**Brad Busse**  
**David Brown**  
**Bryan Harris**

**Monday, April 12, 2004**

## **Introduction**

The initial system integration process has been completed. All 5 phases of Data Acquisition have been completed. The Point Initialization and Tracking algorithms have been integrated into the application framework. Testing has begun on this integration. The Audio Processing framework is still being worked out.

## **Data Acquisition**

Phases One through Four have been fully implemented and tested. Phase 5, consisting of our systems threading and queuing functionality has been implemented and is currently being tested and debugged.

Phases One through Three, represent the implementation and testing of the stand-alone functionality of our system. The final product of phase three has the functionality to Record an AVI video movie and WAV audio file via the Winnov capture card. The validation of phase three proved the hardware interfaces (video and audio) were correctly operating. Phase three also has the functionality to open a pre-recorded AVI file and emulate the events of the capture card. At a variable frame rate the FrameGrabber function extracts one video frame and passes it to Video Processing. The Final Product of Phase three has been used in the implementation and testing of the Point Initialization and Point Tracking Algorithms.

Phase four of the data Acquisition System adds the functionality to do direct data capture from the camera and pass it to the video processing. This phase is the foundation of the final functionality of our system, being able to eliminate the file save, file open middleman. To test Phase Four, the system simultaneously displayed the Active X camera window (what the camera currently is capturing), and a bitmap of what data was being passed to video processing. The system was run for an extended amount of time to prove stability of our system. After detection of some minor memory allocation errors, the system successfully ran for over 30 minutes 3 separate times without error.

Phase five of the Data Acquisition system will be the final product of the data acquisition system, and will be the system framework in which the other algorithms will be integrated with. This phase will include the data queuing system and the multithreaded framework of the system. The data queuing system will enable the system to store frames during unexpected CPU over utilization, and catch-up during times of normal operation. For this to work the average cost of all of the processing algorithms must be less than 33 milliseconds, or the video frame period. This system has been implemented and has been tested. The queuing system with an average processing cost of 20 milliseconds can catch up from 1000 milliseconds of lost CPU time. This is just an initial test; more rigorous testing is in progress to determine the bounds of our system.

Coupled with the queuing system, the multi-threaded framework is being implemented in phase 5. This is currently in implementation and test due to a major design constraint discovered during initial implementation of the thread system. The original design

consisted of spawning a new thread for every entry in the queue. Note, during normal operation the queue is not used, it is only used during unexpected delay. It was determined during testing that the system was never able to catch-up, or come out of the queue. It was determined that the method to spawn the thread took on average 56 milliseconds to complete. However, the thread/queuing system has been redesigned and currently has a 3.5 millisecond overhead average per frame.

Currently phase 5 is in final testing and implementation, it will be complete the week of April 11<sup>th</sup>. Phase 5 testing is the most rigorous because it comprises the final Data.

### **Point Initialization and Tracking**

The frame for the face has been added to the data used for Point Initialization and Tracking. The initialization algorithm still works quite well. Unfortunately the size of the frame has created a need for the face to be farther away from the camera during the capture process. This has caused some problem with keeping track of the points as they are not showing up as distinctly now as before. The tracking algorithm is currently being refined to account for the discrepancies that are showing up. It is imperative that good data be sent from Tracking to Resolution for that algorithm to work well.

### **Point Resolution**

The facial reorientation algorithm has been tested in an integrated part of the total system. The results thus far are successful, though a few issues still require resolution before the system can be said to have passed verification.

Upon completion of an effective reference frame with which data could be gathered, integration and testing of the facial reorientation algorithm could be completed. A short clip involving several head movements was recorded prior to the integration of the facial reorientation algorithm. The algorithm was then integrated into the rest of the system.

When the integrated system was tested using data recorded earlier, the results were promising. The integrated system seemed to correctly capture and reorient the data for the first few frames. After that, significant errors were noted in the data. Fortunately, upon reviewing the performance of the system, we believe that the issue is due to implementation faults, rather than being demonstrative of a flaw in the design of the system. As such, it is only a matter of tracking the bug down, which should be completed by the end of this week.

### **Audio Processing**

Sound Analysis.

The LPC algorithm has been implemented and analyzed using the Windows high performance counter functions. During the next two weeks the MFCC, Pitch, and Power algorithms will be implemented and tested also.

Status of the LPC Algorithm.

The LPC algorithm is written in C++ and working. There is currently no way to test if the values output by the algorithm are correct until we have software from an alternate source to compare them with.

Analysis of the LPC algorithm.

The LPC algorithm is being tested using a wave file containing 1 second of speech. The following table lists the time required by the LPC algorithm to process 1 second of audio data from differing frequencies.

<b>Frequency</b>	<b>Time to process 1 second of data</b>
44100 (audio CD quality)	50 milliseconds
22050	25 milliseconds
11025	17 milliseconds
8000 (telephone quality)	9 milliseconds

These times include the time spent copying data to and from the vector format required by the algorithm. It has also been found out that these copy operations take about 14 microseconds for each 33 milliseconds of audio data. This is 1.6% of the total time, which is not significant in the grand scheme.

If any action is going to be taken to reduce the amount of time required by LPC, another(faster) algorithm should be found and implemented. However, the algorithm currently being tested is already working and is quite fast but perhaps not fast enough.

These details will be more thoroughly considered after MFCC, Power and Pitch have been implemented and timed. After the running times of these algorithms are considered, we will have a better idea of our time constraints.

## **Conclusion**

The initial stage of the complete application has been put together. As expected there are quite a few bugs to work out. The next couple of weeks will be spend fixing and changing where necessary to present a fully functional application.