

Facial Tracking and Animation
2nd Progress Report

Todd Belote
Brad Busse
David Brown
Bryan Harris

Monday, March 29, 2004

Introduction

Two weeks ago we submitted our Critical Design Review. At that time we had performed a proof of concept for each of the subsystems in our overall design: Data Acquisition, Point Initialization, Point Tracking, and FAP Generation. Since then we have begun implementing the base functions of each subsystem, as well as started the integration process, though currently just between Data Acquisition and Point Initialization.

Data Acquisition

With the design constraints nailed down in the Critical Design Review (CDR), implementation of the Data Acquisition System has begun. Phase 1 of the Data Acquisition system has been fully implemented and validated.

Phase 1 of the Data Acquisition system contains the functionality to extract and deliver video frame data from a movie file (avi) to any needing functions. The data acquisition system will run at an adjustable periodic rate, from 25 milliseconds to one second.

The core of the validation and testing of Phase 1 of the data acquisition system was aimed at the timing of the data delivery. Reliable timing characteristics are crucial to the testing of the point tracking algorithms, which this system is designed to accommodate. The system has built-in testing and validation functionality. The system displays, on the monitor, the current data being passed out on the output buffers. This functionality enables visible confirmation that the system is running, and has good behavior. The system has built in checks for the periodicity of the system. The system checks the period on every loop and counts the number of time the period is outside of a plus or minus 10 percent of the target period. Upon completion, the system displays the number of times the period was outside of its bounds, and in which direction the error occurred (higher or lower than the target period). Details about the errors, such as, the actual period and which iteration, can be found in a delay log file.

The results gathered through a series of test runs at different periods indicate that the system is stable and will satisfy the needs of the point tacking and initialization. In a test of 21 runs, the system produced an average of .28 high errors per run, and NO low errors per run. With .28 errors per run, the system has a .08 percent error, which will not compromise the effectiveness of our system. This is acceptable because the system can guarantee a period greater than or equal to the target period, while being stable enough to check consistency.

Currently, Phase 1 of the data acquisition system is being integrated with the point-initialization system. The system will be able to find and identify points from an avi movie file.

The implementation of Phase 2 of the data acquisition system is in-work and should be completed the week of March 29th. It will incorporate capture card interfacing and video data record to file.

Point Initialization

The algorithm for initialization has been proven to work in a stand alone application. This was accomplished by using a single 640X480 image from the camera captured by the old tracking system. This image was placed in a dialog and searched for points. The process hasn't been perfected, but under certain conditions it finds and identifies the specific feature points we are looking for.

Now that Phase 1 of the Data Acquisition subsystem has been complete, the initialization is being integrated into the basic framework of the new application. The Point Initialization objects have been included into the project and their basic functionality integrated. The new framework is different in two ways. First, the data represents a 320X240 frame, not 640X480. Second, before the algorithm was running off of the image after it had been displayed to the screen. Now it is using the raw bitmap data to find the points. This change has caused some problems to occur, and therefore the algorithm is currently being refined.

Point Tracking

To review the purpose of the facial reorientation algorithm, we wish to develop an algorithm which, given an array of X and Y data coordinates, the algorithm will reorient the data to fit the standard (head-on) view. This data can be compared to the last frame's results to get the distance each marker has moved in the interim. The displacements of the data points can be passed to the facial animation algorithm to form the .fap file.


At this point, we have a C++ implementation of the facial orientation algorithm that performs according to specifications. This stage will next be integrated into the main system once the modules it depends on (data acquisition and point initialization) are in place. Once this module is integrates with the rest of the program, all that remains is to compare the data points with their last seen locations. This task is not expected to take much time. The development of the facial reorientation algorithm is proceeding on schedule, only lacking integration and a small amount of work to be complete.

FAP Generation

Background

FAPU conversion requires that each feature point be converted from a number given in pixels to a unit-less number. The output file will be an ASCII text file in the FAP format.

In order to get a unit-less number from a number of pixels, we divide by certain FAPUs — facial animation parameter units. FAPs are divided into 10 groups. For each group, the FAP is divided by one of 6 FAPUs — shown in the figure. These are also measured in pixels. When the FAP(measured in pixels) is divided by the respective FAPU(again, measured in pixels), the resulting number is normalized and is saved to the output file.

	Description	FAPU value
	IRIS Diameter in neutral face	$IRISD = IRISD0 / 1024$
	Eye Separation	$ES = ESO / 1024$
	Eye - Nose Separation	$ENS = ENS0 / 1024$
	Mouth - Nose Separation	$MNS = MNS0 / 1024$
	Mouth - Width Separation	$MW = MW0 / 1024$
	Angular Unit	$AU = 10^{-5} \text{ rad}$

Current FAPU Demo

There are two input arrays to the FAPU Demo. The first array contains 68 floating point numbers representing the x-values of each feature point. The second array is the same format as the first, but instead represents the y-values. If a feature point is to be skipped over, it will be given a sentinel value, or another array can be used as a mask for which of the 68 FAPs are in use. The current demo only prints out the FAPs in use, and does not use either of these methods.

The FAPU Demo has been completed, based on random number generators. Our group is using 22 of the 68 feature points. The file contains 1000 lines of numbers representing these 22 feature points at time intervals.

Since the information generated by the demo is based on the computer's random number generator, the FAE will likely generate a face which moves in a completely random pattern. This will not be recognizable as any form of movement that someone would make.

Conclusion

The implementation process is moving forward at a good pace. During the week of March 29th the Data Acquisition and Point Initialization subsystems should be fully integrated. This entails both finding the points initially, and finding them frame by frame. While it won't account for head rotation and translation, it will track the absolute frame position of the points. Also within the next couple of weeks Point Tracking and FAP Generation will have coded versions of the algorithms from the CDR. At this point in time they can

be integrated as well. This should be a fairly short process. Once they are integrated then testing of the entire system can begin.