Ain't Gonna Micky Mouse 'Round No More!

A Project On Eye-Tracking Systems For Eventual Mouse-Input Removal.

Group 13:

Behnhard Behn.
Jeffrey Lee
Bharat Bhat.

18-551: Final Report.
Spring 2000

SECTION I: INTRODUCTION TO THE FIELD.

I.I Project Beginnings: Reasons.

The first and foremost problem presented in using a mouse to specify computer screen activity, revolves around the fact that many people feel uncomfortable and constrained by it. Other than the perturbing feeling resulting from twiddling fingers and unsuccessful mouse pointing, there are numerous other factors. Some of these are quite serious, including handicapped individuals who have malfunctioning motor systems, or others who undergo severe pains to correlate their hand-eye movements with accuracy. There are also two other fundamental problems resulting from mouse usage. As with any peripheral input device, constant utilization often leads to degradation in the device's response and accuracy. Although reliability in the long term is not a terrible nuisance in comparison to the other considerations presented, the eradication of reliability issues aids to the aesthetic and functional value of a system. Another issue we suggest that this apparatus is capable of addressing is the constant discontent users have with portable PC pointing devices. The consensus upon this issue is unanimous: while laptop mice are maneuverable, the hassle and effort required to extract this maneuverability often results in a frustrated end-user. Finally, many people feel that auxiliary devices bring along with them a sense of mental constraint. The eye selects an object a second or more before the mouse can even be moved around to point at this object. In other words, when dealing with an interface for human-computer screen interaction, as with any mind-limb interaction, it is always a more liberating and natural experience to walk without crutches.

SECTION II: OUR SOLUTION: NON-TECHNICAL AND TECHNICAL.

II.I Non-Technical.

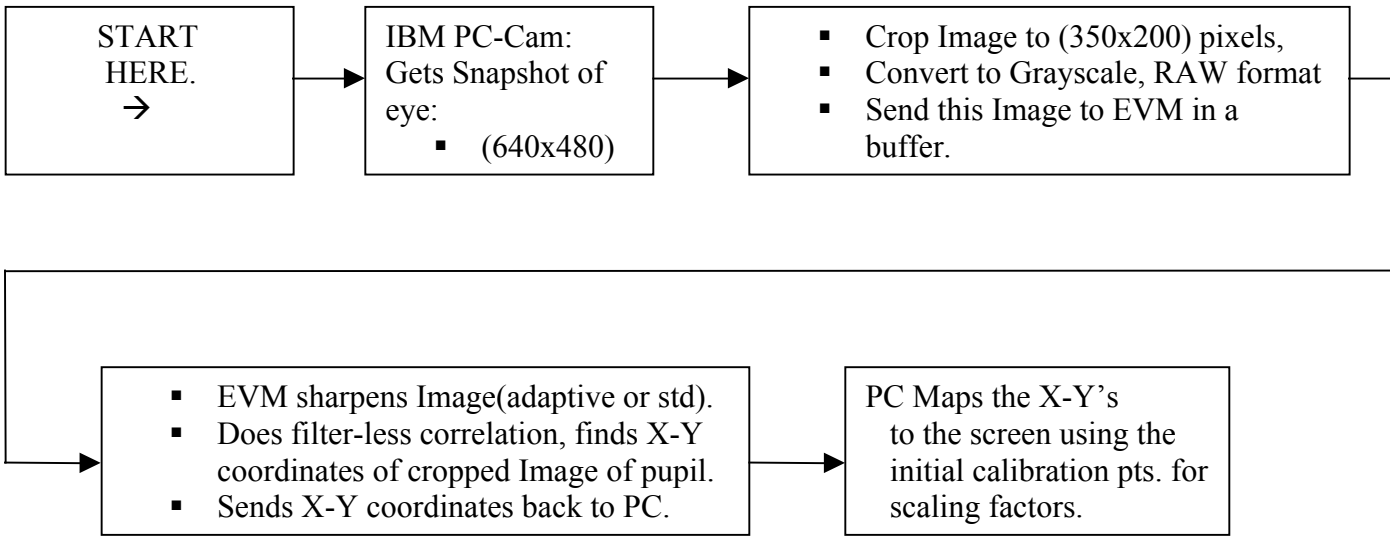
Simply stated, we replace, or enhance, the current movement tracking technology(i.e., the mouse) to an eye-tracking mechanism that rids the system of clumsy external/physical control. As insinuated in the preceding paragraph, this solution, we hope, will greatly diminish many problems associated with inconsistent or impeded hand-eye-coordination. It mitigates effects of aggravation—both psychological and physical—which many people experience when confronting computational interfaces.

There are myriad applications resulting from such a framework. The system being created is one of visual selection. The scope of this project extends into various medically related domains, but we will only pursue the realm of computer input. Upon the proper calibration of this system with our proprietary software, any menu driven device—computer, television, bank menu, et cetera— can be enjoyed to the fullest extent by the handicapped principle.

II.II Technical: General Idea.

The gist of our system algorithm consists of distinguishing the pupil from the rest of the eye using a 2-D filter-less correlation for the pupil image: sending a cropped image of the eye to in a buffer to the EVM, finding the midpoint of the pupil from the EVM correlation results, return the X-Y coordinates to the PC, and from these returned coordinates, the PC maps the virtual mouse pointer to the screen according to the scaling factor(from the initial user screen boundary calibrations to the X-Y point just returned from the EVM).

The flowchart below provides a graphical analogue to our overall system algorithm. The details of each sub-algorithm will be presented in Section V.



SECTION III: BACKGROUND: HISTORY, EXISTING RESEARCH

III.I History.

The eye performs perhaps the most intensive direct sensory input function of all the human sense organs. When considering the impact of sensual information to the brain, sight appears as the most immediate and influential. Many people are often fascinated by where others are looking. One can be led to believe that what a person consistently chooses to observe and see determines many aspects of that person's personality. On the other end of the spectrum, eye movement also models and relegates a decision-making system. Before a person buys something in a grocery shop, he/she gazes (looks at the specified object longer than a mere glancing) at the object with his/her eyes. At the end of the 'gaze,' the person has chosen the object. The next step is to analyze the shape of the object and select the object with a movement of the hands. However, many people act restrictively and/or indolently in their selection mechanisms. Even considering those who have perfectly healthy limbs and movements, there arises the question: Why/How must I actually go and get it? Thus, it can be seen that the eventual repercussions of eye-tracking systems unequivocally stand on the scale of re-shaping our entire society.

III.II Current Research.

Pupil-tracking devices and methods have been explored extensively in the past few years. However, most of the research we found concerning this field resulted exclusively in pupil-tracking technologies. Aside from virtual reality arcade games, and some medical applications slightly removed from the flavor of our system, we did not come across many application-based projects.

Many of the existing eye-tracking systems perform matched-pattern recognition for the pupil. However, some people have employed more intricate methods in order to account for head movement, increasingly accurate non-optical eye-tracking systems, less costly implementations, and systems that allow for additional elements of decision—such as recognizing winks as commands. These more intricate methods involve Reflections (Punkerje Images), induction-coils, and electro-oculography[1]. There is no definitive consensus upon which technique works best for all applications. Each implementation method has its drawbacks and virtues. Due to the unwieldy contraptions used in non-camera based systems, we feel that an optical approach is the user-friendliest.

The only prior 18-551 project similar to ours was an Iris-Recognition system. The Iris-Recognition project, however, did not deal with real-time video processing or eye-tracking. Most of our information and inspiration for the eye-tracking portion of our project comes from the work of Arrington Research[2].

---

[1] http://www.cs.sunysb.edu/~vislab/projects/eye/Reports/report/index.html, and www.arringtonresearch.com.

[2] www.arringtonresearch.com.

SECTION IV: VIDEO FORMATTING.

We encountered many problems obtaining the appropriate image format for transferal to the EVM. The format we desired was a single 8-bit Grayscale array(or, buffer) which contained the entire cropped image with no header information(i.e., RAW). Arriving at this point was realized through the following steps:

I. Camera snapshots saved onto a file and converted to RAW Grayscale format with Adobe Photoshop. Then, the formatted file was sent to the EVM for a single correlation result. This phase existed merely to verify the integrity of our PC-EVM transfer process code(HPI transfer), and to be certain of the accuracy of our EVM correlation code.

II. The next step was to get the image into a buffer instead of into a file, and convert the buffer image format from 24-bit BITMAP to 8-bit RAW Grayscale.

    In the camera SDK, there is a parameter which stores the image data to a buffer, instead of to a file. We switched this parameter directly into the SDK code and received a 24-bitRGB buffer containing the image. Then, we read the manual to parse out and decipher the header information[3]. In the same loop, we converted all the RGB values to Grayscale by averaging the red, green, and blue values[4].

III. The final step was to crop out the 640x480 pixel image to be only 350x200 pixels. We did this step, cutting out the unnecessary parts, and sent the image to the EVM using HPI transfer.

IV. A few fortunate issues regarding formatting were:

    a. The SDK contained parameters to store the image to a buffer.
    b. The image stored in the buffer was in RGB format, which is far easier for RAW Grayscale conversion than let's say, JPEG or BMP.

[3] Used the sizeof(header_info) operator to skip the header information(this converts the image into RAW format).

[4] 24-bit BMP format stores its image info in RGB values(i.e., 8-bits for red, 8-bits for green, and 8-bits for blue). The resulting averaged value is 8 bits, denoting the darkness or lightness of the color(i.e., black and white, or Grayscale).

SECTION V: ALGORITHMS.

The main algorithms used were in the following domains:

I. Formatting procedures(already explained above).
II. Image sharpening(adaptive and standard methods).
III. Lowest Correlation-Value Finder; i.e., determination of the pupil-center.
IV. Initial Calibration.
V. Paging and PC-EVM data-transfer/communication.
VI. Implementing the Real Time model.

V.I Image Sharpening.

The sharpening algorithms were split into two types: standard and adaptive. Adaptive sharpening uses a 3x3 filter also, but it has the additional tenet of using a thresholding factor, which delimits the saving of pixels only if the sharpened value exceeds the threshold value. The methods and for both these procedures are presented in the Appendix(Section X).

V.II The Filter-less Correlation.

Originally, we were planning to do a correlation using the entire pupil as the filter(i.e., a 25x25 filter, when using a 640x480 image resolution). But, with the IR-LEDs, the magnification, and the Dichroic mirror, the image we received allowed us to substantially reduce the filter size for the correlation and still obtain exactly the same accuracy for pupil detection.

Also, initially, we used standard correlation to locate the pupil center. However, we realized later into the project that since we want the lowest pixel values from the correlation(i.e., there is no pattern or contrast-based image that needs to be matched), there is no need to do any multiplication in our matching algorithm. Instead, we can merely add-up the pixel values on a 5x5(or other size) square. The point with the lowest pixel value sum after the entire image has been processed is the mid-point of the pupil. This method saved us a noteworthy amount of time and calculations(since addition, then multiplication is far more expensive than mere addition).

Another aspect addressed in our correlation is the kernel size of the 'addition-square.' For example, if the user has an unusually bulbous eyebrow, it will be present in the cropped image, and might potentially steer correlation calculations awry. To take care of such cases, we made the 'addition-square' size parameters global DEFINE constructs which can easily be altered if the need arises. In the case of a bushy eyebrow, the 'addition-square' size can be made 9x5, thus reducing the impact of a horizontal brow. Another example might be for someone who has their eyelashes impede

correlation calculations. In this case we would use a 5x9 'addition-square.' For the most part, however, we want to maintain the square size to 5x5, as it provides the optimum speed vs. accuracy ratio.

True Correlation using entire pupil for the filter(25x25 pixels):

25x25 = 625 calculations per pixel.
Search area of Cropped Image: 350x200 pixels = 70,000 pixels.
Total Calculations: 70, 000 x 625 = 43,750,000.
Cannot do this in Real-Time.

Our present Implementation(filter for pupil is 5x5, and the correlation is non-multiplicative):

5x5 = 25 calculations per pixel.
Search area of Cropped Image: 350x200 pixels = 70,000 pixels.
Total Calculations: 70, 000 x 25 = 1,750,000.
Makes Real-Time possible.

Reduction of calculations necessary to locate pupil: 42 million. This is 25 times faster!

The Following Chart illustrates the difference in efficiency between using varying-sized 'addition-squares:'

Size of Addition-Square('filter')	Cycles(in millions) to do the Correlation.
5x5	57.2
13x13	308.0
5x9	97.2
5x13	266.0
9x9	278.8

V.III Initial Calibration Process.

Initial calibration is the step where each particular user's screen boundaries are stored. Through the use of Visual C++ display code, the user is prompted to enter each of the four corner points. Each corner-image is sent to the EVM for the correlation results. The returned X-Y coordinate(showing the pupil position of the user looking at a screen corner) is stored on the PC-side as a screen boundary. After receiving the four points, the calibration and mapping functions have the data they require to process the X-Y coordinate to virtual mouse pointer operation. Invariably, the x-y differences between the points are only in the region of one hundred pixels, if not less. This poses accuracy problems when the point is mapped to the screen. However, as expected, good calibration results(i.e., a user with sharp contrast of pupil to eyebrow, eyelid, skin, et cetera, or a user with large eyes, or a user who does not mistakenly input an image as a corner when in fact it is not) significantly improves mapping accuracy.

V.IV PC-EVM Communication.

The image is sent to the EVM with HPI transfer. The EVM correlation results are sent back to the PC with sync_send( )s—sync_sends( ) use mailbox transfers also.

Initially, the EVM controlled the transfer—i.e., when the EVM was ready, the PC sent the data over. We put the PC in charge—i.e., the EVM in in a wait_transfer( ) until the PC is ready, the PC sends image data over(and stalls until the EVM sends results back and should be ready for an image again), and so on.

The image is stored in EVM off-chip memory when it is transferred via HPI. It is then loaded to the wkspace[ ] buffer through DMA(running as a background process while the EVM is processing something else).

V.V Implementing the Real Time model.

The initial calibration code being composed, we had to embark upon making the system loop-able in order to get it real time. This incorporated both paging and timing issues. As previously mentioned, we used the PC-side as the initiator for the transferring of data. By the time the PC receives the correlation results from the EVM, it has already sent out the next cropped image to the EVM. To implement the loop, we simply placed a while(TRUE) loop surrounding the transfer functions.

SECTION VI: SPEED ISSUES.

As briefly mentioned above, the initial method for locating the pupil of the eye was correlating a five by five pixel black valued filter with the images of the eye taken by the PC Camera. This would yield peak minimum correlation result values in the vicinity of the pupil. With the PC Camera writing 640 by 480 pixel images to the PC memory, it was decided that the entire 640x480 pixel image would not be transferred to the EVM in its entirety. An image 640x480 would take a considerable amount of time to be correlated with a 5x5 filter, and would also take too much time to be transfered to the EVM memory. With the image scaled down to 350 by 200 pixels, and the color depth reduced to 8 bits, we were able to reduce the unnecessary overhead resulting from excessive image data and excessive image color information.

The other main speed issues encountered throughout our design and implementation are noted below:

VI.I Correlation vs. Cheap Correlation.

The nature of the region we are attempting to detect(i.e., the black pupil on top of a lightened iris('lightened' due to the IR LEDs), allowed us to implement a "watered down" correlator, as opposed to a "true" correlating algorithm. The pupil of the eye is black, and on an 8 bit RAW grayscale color map (i.e. 0 to 255,) black values represent the low end of the scale. Searching for the 5 by 5 pixel area essentially consists of summing up the pixel color values of the 25 pixel area. The iteration of this "correlation" or sweep that has the smallest sum of pixel color values is the region with the highest concentration of black.

As previously mentioned in Section V, this reduced arithmetic sequence requires far less calculations than the true correlation originally set for deployment. A true correlation requires memory fetch and multiplication of the color value of each element in the filter with each color value of each corresponding overlaying pixel from the image, and then the sum of these 25 multiplications. The reduced "correlation" algorithm we employed requires only memory fetch and summation 25 times. This drastically reduced the number of cycles required by the EVM to calculate the location of the pupil.

VI.II Kernel Size Of Correlator.

The varying camera angles and individual corporal contours on occasion require a "correlation" filter size of unorthodox dimensions. Pupil detection, under certain circumstances, can benefit greatly from different search regions (filter sizes.) The size of the correlation kernel (filter size) increases the number of cycles required to complete both the true correlation and the reduced "black search." For each cell pixel added to the kernel, an additional multiplication and summation operation is required. Multiplying by the number of pixels in the image accounts for a modest increase in processing cycles. Increasing the size of the kernel is therefore quite costly in terms of processing cycles. A 5 by 5 kernel was determined to be the most reasonable compromise for it is assumed that

the largest black dot from the PC Camera snapshot will always be the pupil, and that no other object in the snapshot will have distinct black areas of 5 by 5 pixels or greater.

VI.III EVM Memory Paging.

Processing video with the EVM requires relatively large memory buffers to store the image data in a location readily accessible by the DSP processor. The EVM is equipped with 16 megabytes of external RAM. The external RAM is the most sensible location to store the image data. In this specific scenario, a 350 by 200 pixel image with 8 bit color depth requires 70,000 bytes of memory. Upon initialization, this memory slot is allocated and is available for storage. For every individual external memory word transfer that the processor requests, 15 clock cycles are required. With a DMA block copy, 15 cycles are used to set up the transfer, and each subsequent word transfer requires 1 or 2 cycles per word. Given the size of the on-chip executable and amount of available on-chip memory, a page size of 71 lines or 24,850 bytes. The "correlation" sweep requires 2 rows and 2 columns of valid data on the extremes of the local page, therefore on a 71 line page the correlation can only run until line 69 of the page. The next page transferred to local memory will then start on line 67, and so forth. 25 external memory accesses per pixel results in 15*25*(350-4)*(71-4) = 8,693,250 cycles. With local memory paging, the memory access processor usage reduces to 20+(2*350*71) + (25*(350-4)*(71-4)) = 629,270, providing more than an order of magnitude of improvement in the processor usage for the memory transfers.

VI.IV Image Sharpening.

Image enhancement techniques provide significant improvements in the quality of input images. Once the data is paged into local memory, optimal image enhancement can be employed while minimizing memory transfer overhead. Sharpening filters essentially require the same computational resources as the correlation. With optimal adjustment of the headset camera apparatus, very high quality data can be obtained thereby passing the necessity for image enhancement.

VI.V Software Optimization.

The EVM's arithmetic capabilities are best exploited when the high-level language shows explicit instructions for quasi-simultaneous calculations. The compiler is able to create processor specific instructions that are optimized to minimize the cycles required for the array of calculations and maximize the usage of the EVM's arithmetic modules. With the implementation of a variably sized kernel in the pupil location algorithm, unrolling of the inner loops becomes more impractical than beneficial. Performance comparisons can be made with a fixed kernel size of 5 by 5, for example. With a fixed kernel size, the two inner loops can be unrolled into a cluster of 25 multiplication and 25 sum operations. With these two loops unrolled, the EVM will be able to parallelize these arithmetic operations, increasing throughput in certain instances.

SECTION VII: TIMING.

Timing is an extremely important point in the construction of a real time application. With several components performing different tasks at different times in one fixed chronological order, a strict order of execution must be maintained to ensure robustness. Once the PC Camera has taken a snapshot and stored it in PC memory, the EVM must be in a ready to receive state. Once the memory transfer to the EVM's external memory is complete, the PC must be in a ready state to receive the x-y coordinates from the EVM. Once the EVM returns the x-y coordinates to the PC, it must reenter the ready state for the next snapshot frame from the PC Camera. This stop and go timing was achieved by using strategically placed wait statements on both the EVM and the PC. Command signaling to wake the devices out of wait states was achieved with synchronous PCI messages.

VII.I EVM Profiling.

The performance of the EVM throughout the processes of pupil tracking is a very important gauge of how far into "real-time" this device can enter. A true correlation is a mathematically intensive calculation, especially at motion video frame rates, and at motion video image resolutions. In this specific application, the images were scaled down to 350 by 200 pixels and the color depth reduced to 8 bits, thereby reducing the memory and processor requirements. The actual pupil finding technique, along with the local memory paging process, requires on the order of 57 million cycles for an entire frame. This number was obtained by placing profile points at the beginning and the end of the "correlation" function on the EVM. The memory transfer from the PC to the EVM was also timed to be on the order of 10 million cycles. A count of 67 million cycles to run through the process of obtaining the data of an image, and locating the pupil in the image, is still in the order of a real time application.

Here are the profiling results for individual and total functional blocks:

Procedure.	Cycles (in millions)	Time (sec)
Filterless Correlation(5x5 filter).	57.2	1.43
Sharpening(standard)	25.9	0.65
Adaptive Sharpening	159.2	3.98
Total Processing Time:	67.2	1.68

VII.II Frame Rate.

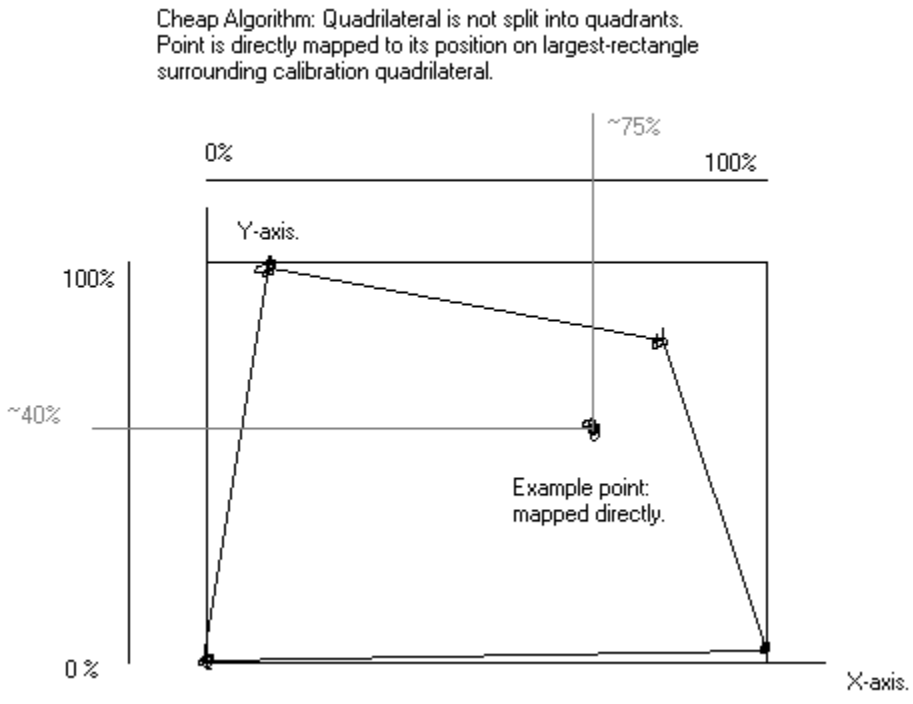
The architectural nature of this application does not allow for a maximal frame rate to be used. With the stop and go wait functions, everything is timed in a sequential order. The 67 million cycles that the EVM requires to process one frame, at the DSP clock frequency of 40 MHz, elapse a time of about 0.60 frames a second. Assuming the PC side image snapshot, display and calibration sequence do not occupy more than 1/6 of a second, a total processing frame rate of half a frame per second can be obtained. A maximal frame rate could be obtained with the use of double buffers and ping pong data transfers, thereby allowing the EVM to correlate and receive data at the same time, and the PC can snapshot and send map correlation points at the same time. This kind of architecture would require serious changes in the data transfer procedures.

SECTION IIX: THE MAPPING ALGORITHM:

There are many problems associated with mapping a point from an arbitrarily shaped quadrilateral to its relative spot on a rectangle or square. The quadrilateral has to 'stretched' in certain parts and 'compressed' in others. This relates to the *Dimension* of matrix *Subspaces*. Any NxN matrix, when compacted to a NxM matrix(where N > M) will lose information(or, in matrix algebra terminology, it will lose *Dimensions*). The reverse is also true: a matrix with less dimensions linearly transformed to a matrix with higher dimensions will have to be 'padded' with zeroes, or other chosen constants. Thus, the mapping algorithm is not as trivial as one would initially infer. The cost involved in a sophisticated mapping algorithm, although not horrendously exacting, does indeed impede speed optimizations. For our case, we found four mapping methodologies, or algorithms, each with its own accuracy to complexity(or cost) ratio:

IIX.I The Cheap mapping algorithm[5].

This method, although the fastest and simplest, is also the least accurate. It only works when the quadrilateral resulting from the initial calibration is not excessively distorted. In most cases, however, the calibration quadrilateral forms a pseudo-rectangular shape, and the cheap mapping algorithm functions quite reasonably.

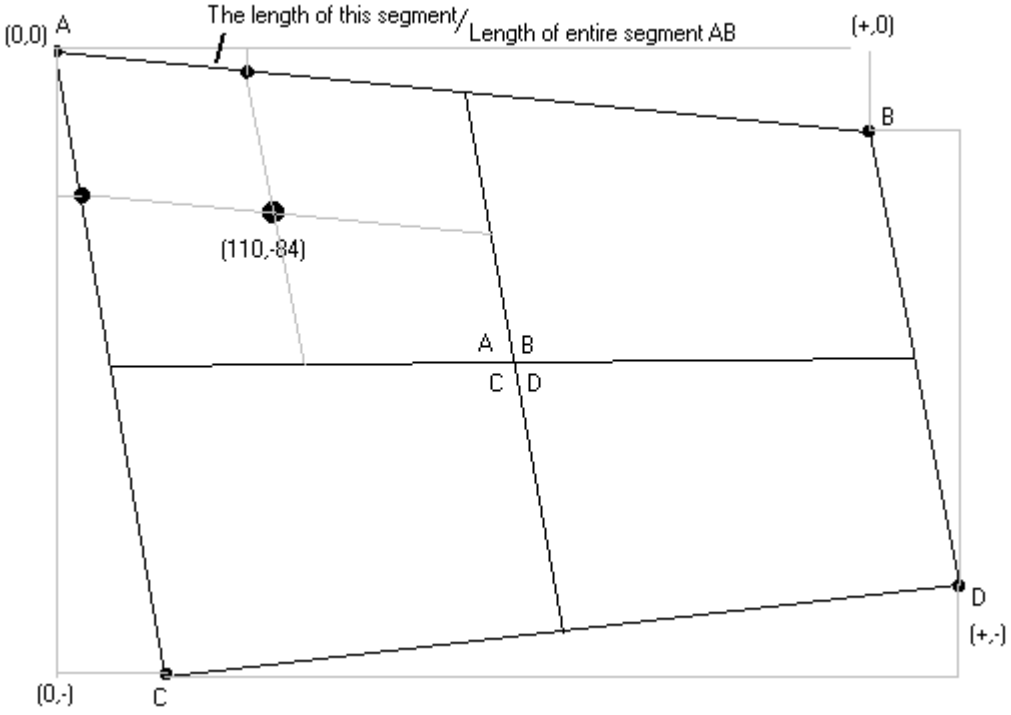


[5] The algorithm presently in use.

IIX.II 2-Tier Complex Mapping algorithm[6].

With an accurate pupil location algorithm and carefully posed camera data, the traced path of the eye, while following a rectangular perimeter, is not rectangular. In order to accurately map "correlated" pupil locations onto the rectangular screen, a method of converting coordinates from a quadrilateral plane to a coordinates on a rectangular plane is necessary. From a variety of algorithms available, including one specifically tailored for this application which did not work as expected, a geometric Cartesian approach was chosen to be most suitable. Given four calibration points, or coordinates of the pupil locations gazing at each of the corners of the screen in front of the user, a quadrilateral can be constructed. Drawing vertical and horizontal bisectors from midpoints of each edge of the quadrilateral splits the quadrilateral into 4 quadrants. Each of the four quadrants contains two edges of the original quadrilateral, and these two edges can be used to calculate the relationship of the correlated x-y coordinate on a quadrilateral plane to a rectangular plane. This is achieved through the calculation of intersection points of the two edges, and two lines parallel to the edges, each drawn out to the edge from the correlated point. This gives a distance on each edge from the origin of the edge, with this distance, normalized to the length of the edge, the location of a correlated point on the quadrilateral plane can be expressed accurately in the rectangular plane. On certain occasions, when the correlated point has coordinates that are outside the scope of the original calibration quadrilateral, the absolute coordinates must be truncated. With a unit scale, the truncation values are either 0 or 1 depending on which quadrant the out-of-bounds coordinates fall into. This method provides a simple way of correcting the EVM output coordinates when the calibration frame is not a right-angled quadrilateral. The current operation of this algorithm may incite inaccuracies with correlated coordinates on or near the boundaries of the quadrants. However, this inconsistency is relatively unimportant when compared to some of the obtuse and unorthodox calibration quadrilaterals.

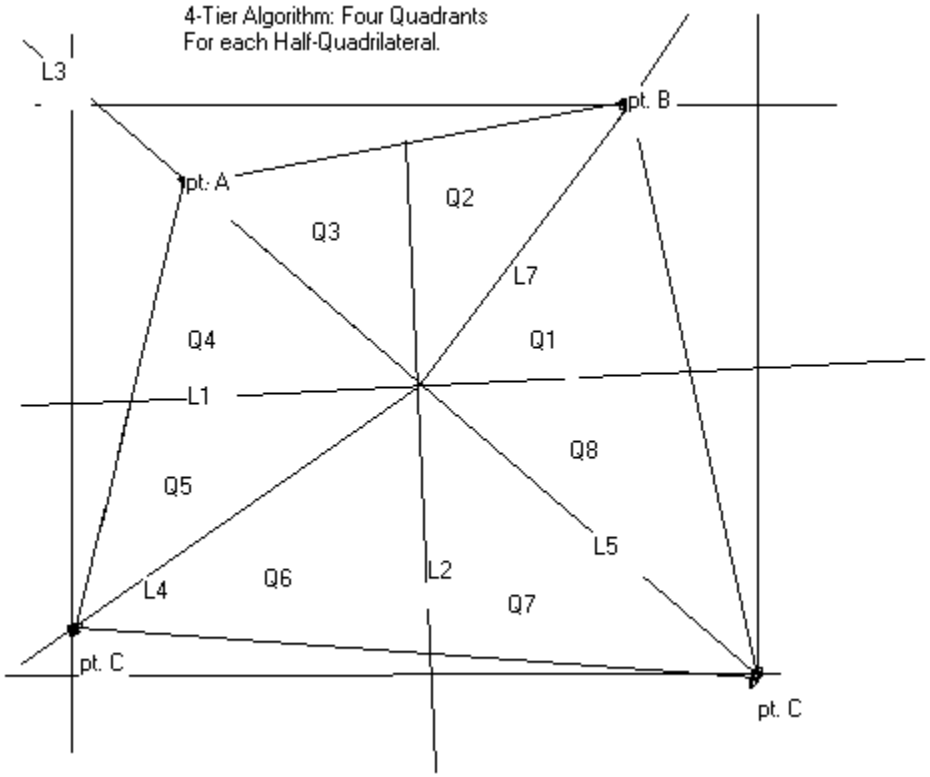
[6] This was the algorithm initially implemented in the code. However, upon finding many of the same mapping problems associated with the Cheap mapping algorithm, we decided to opt for a faster method.



2-Tier Mapping Algorithm: Half-Quadrilateral Split into two quadrants.

IIX.III 4-Tier Complex Mapping algorithm[7]

This method follows the basic tenets of the 2-Tier Complex Mapping. However, it purports to fully subsume all possible regions of the quadrilateral. That is, even with the most erratically shaped quadrilateral, this algorithm will map it to the correct point on the correlative rectangle. The high degree of accuracy is directly related to the number of times the quadrilateral is split with regard to a coordinate system[8].



IIX.IV Mapping Research Results: Wolberg's Algorithm.

We found an algorithm on the web which claimed to do exactly what we desired. However, even after careful code implementation of this algorithm, it gave us some obviously incorrect results for certain cases. We opted not to use it in the end. The algorithm is included in the Appendix.

---

[7] The 4-Tier Mapping algorithm, we believed, was too complex to implement; i.e., we were certain there existed a more efficient and simpler algorithm to map a point on a quadrilateral to its correlative point on a rectangle.

[8] That is, it is also more costly in time and complexity in implementation.

Section IX: Future Improvements.

Our current pupil tracking implementation provides users with a basic working model. In the future, many modifications can be added to make the system more robust and user-friendly.

IX.I Software Improvements.

The Visual C++ GUI provides basic setup and display features that allows people to calibrate and see where they are looking at in relation to the screen. Improvements can be made by making this display invisible to the user; i.e., ultimately replacing the entire GUI with a single mouse pointer representing where the user is looking.

Additionally, our current project calculates points based on the initial calibration points. When the user moves or turns his/her head in any direction, the initial calibration points are no longer valid. They will see that the virtual mouse pointer no longer correctly references what they are looking at. In order to compensate for this, a few things can be done. One idea is to have an emitter placed on top of the monitor and a sensor placed on the head unit that will constantly update the distance and angle of the head from the monitor. With this information, mathematical equations can be derived to constantly update where the virtual mouse pointer should be placed.

IX.II Hardware Improvements.

The head unit constructed for this project provides the necessary features to grab good input images to be used as data. Aesthetic value and comfort are the most important features of the head unit that need to be addressed. By creating a custom molded plastic unit, size and weight can be drastically reduced. It also provides the option for various styles and colors. Proper housing for the IR LEDS can also be created, instead of the free hanging wire assembly currently implemented.

The digital video camera is the primary culprit when dealing with weight. Currently, it is the heaviest device attached to the helmet. In addition, the USB cord that connects it to the computer is bothersome and often discourages users. Furthermore, camera resolution is one of the limiting factors in how well data can be represented. A 640x480 resolution only captures eye movement of about 100x70 pixels. When this is upscaled to fit a 1200x800 screen resolution, accuracy is inherently lost. As technology advances within the next few years it will be possible to use smaller, higher resolution video cameras, with wireless adapters to transmit data to the PC.

Section X: Appendix.

X.I Sharpening:

Source: http://www.cee.hw.ac.uk/hipr/html/unsharp.html

Unsharp Masking

A common way of implementing the unsharp mask is by using the negative Laplacian operator to extract the highpass information directly. See Figure 5.

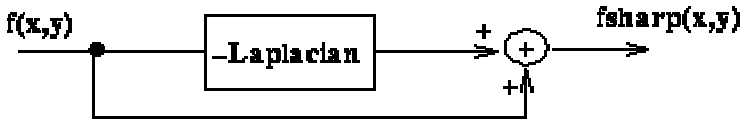


Figure 5 Spatial sharpening, an alternative definition.

Typical Laplacian filters can be described using the following 3x3 filters. For our project, we chose to implement the filter on the far left of Figure 6.

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

1	-2	1
-2	4	-2
1	-2	1

Figure 6 Three discrete approximations to the Laplacian filter.

X.II Adaptive Unsharp Masking

This filter can be re-written as $^{1/16}$ times the sum of the eight edge sensitive masks shown in Figure 9.

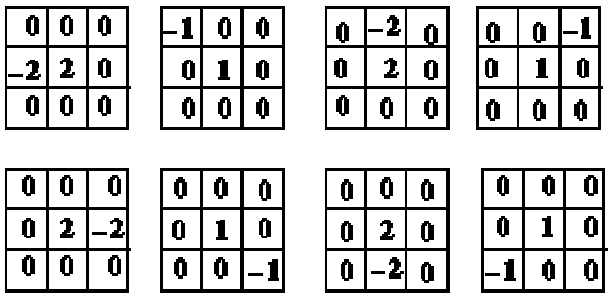


Figure 9 Sharpening filter re-defined as eight edge directional masks

Adaptive filtering using these masks can be performed by filtering the image with each mask, in turn, and then summing those outputs which exceed a threshold. As a final step, this result is added to the original image. (See Figure 10.)

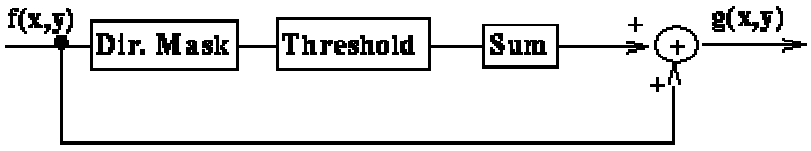


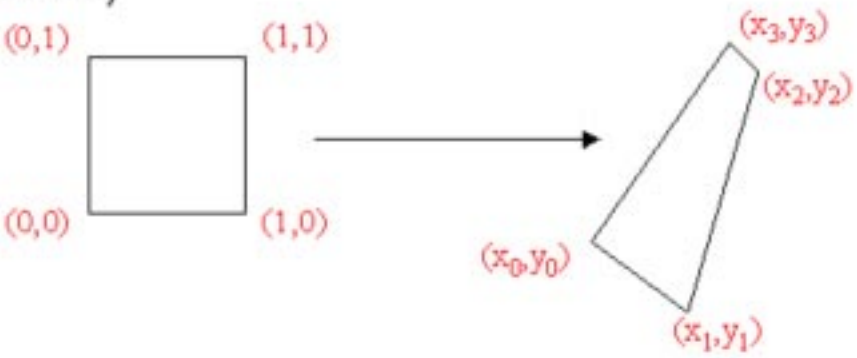
Figure 10 Adaptive Sharpening.

This use of a threshold makes the filter adaptive in the sense that it overcomes the directionality of any single mask by combining the results of filtering with a selection of masks -- each of which is tuned to an edge directionality inherent in the image.

X.III Wolberg's Algorithm: Explanatory slides:

The Projective Mapping: Case #1

Complete solution for the square-to-quadrilateral case (Wolberg, 1990)



Step 1: Define the following:

$$\Delta x_1 = x_1 - x_2 \qquad \Delta y_1 = y_1 - y_2$$
$$\Delta x_2 = x_3 - x_2 \qquad \Delta y_2 = y_3 - y_2$$
$$\Delta x_3 = x_0 - x_1 + x_2 - x_3 \qquad \Delta y_3 = y_0 - y_1 + y_2 - y_3$$

Step 2a: Solution if $\Delta x_3 = 0$ and $\Delta y_3 = 0$:

(Affine case, where square maps to parallelogram)

$$a_{11} = x_1 - x_0 \qquad a_{21} = x_2 - x_1 \qquad a_{31} = x_0$$
$$a_{12} = y_1 - y_0 \qquad a_{22} = y_2 - y_1 \qquad a_{32} = y_0$$
$$a_{13} = 0 \qquad a_{23} = 0$$

Step 2b: Solution if $\Delta x_3 \neq 0$ or $\Delta y_3 \neq 0$:

(Projective case, where square maps to a quadrilateral that is not a parallelogram)

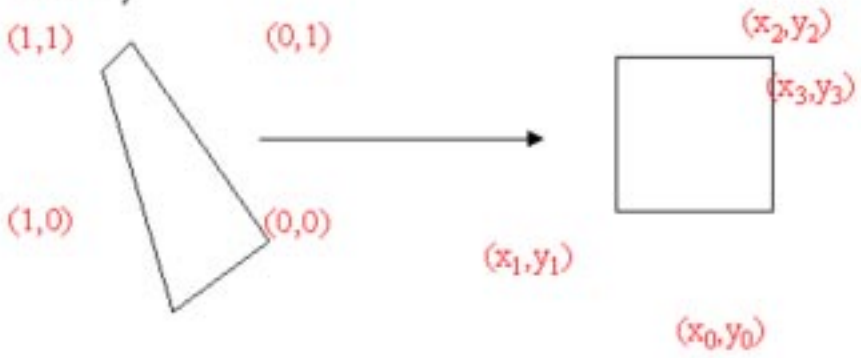
$$a_{13} = \frac{\begin{vmatrix} \Delta x_3 & \Delta x_2 \\ \Delta y_3 & \Delta y_2 \end{vmatrix}}{\begin{vmatrix} \Delta x_1 & \Delta x_2 \\ \Delta y_1 & \Delta y_2 \end{vmatrix}} \qquad a_{23} = \frac{\begin{vmatrix} \Delta x_1 & \Delta x_3 \\ \Delta y_1 & \Delta y_3 \end{vmatrix}}{\begin{vmatrix} \Delta x_1 & \Delta x_2 \\ \Delta y_1 & \Delta y_2 \end{vmatrix}}$$

$$a_{11} = x_1 - x_0 + a_{13}x_1 \qquad a_{21} = x_3 - x_0 + a_{23}x_3$$
$$a_{12} = y_1 - y_0 + a_{13}y_1 \qquad a_{22} = y_3 - y_0 + a_{23}y_3$$
$$a_{31} = x_0 \qquad a_{32} = y_0$$

recall that $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$

The Projective Mapping: Case #2

Complete solution for the quadrilateral-to-square case (Wolberg, 1990)



Step 1: Compute the square-to-quadrilateral parameters as specified in Case #1:

$a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, a_{31}, a_{32}$

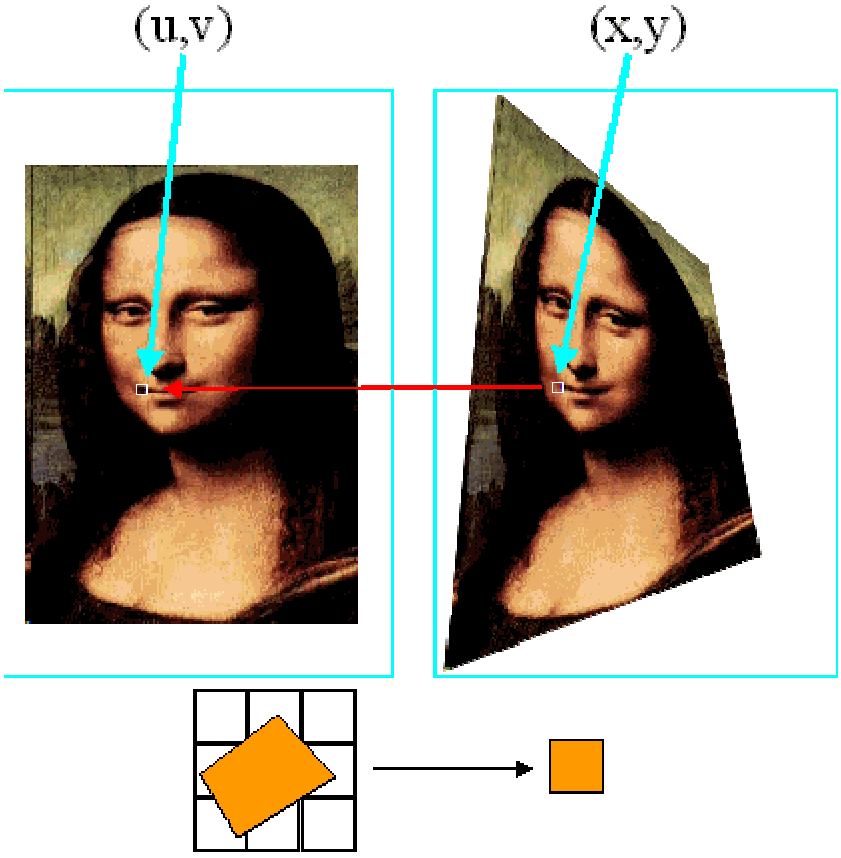
Step 2: Compute the intermediate parameters A'_{11}-A'_{33} as follows:

$$A'_{11} = a_{22} - a_{23}a_{32} \qquad A'_{12} = a_{13}a_{32} - a_{12} \qquad A'_{13} = a_{12}a_{23} - a_{13}a_{22}$$
$$A'_{21} = a_{23}a_{31} - a_{21} \qquad A'_{22} = a_{11} - a_{13}a_{31} \qquad A'_{23} = a_{13}a_{21} - a_{11}a_{23}$$
$$A'_{31} = a_{21}a_{32} - a_{22}a_{31} \qquad A'_{32} = a_{12}a_{31} - a_{11}a_{32} \qquad A'_{33} = a_{11}a_{22} - a_{12}a_{21}$$

Step 3: Compute the 8 quad-to-square parameters:

$$A_{ij} = \frac{A'_{ij}}{A'_{33}}$$

General Linear Backward Mapping



destination pixel color
computed as weighted
combination of source pixel colors

Backward mapping algorithm

```
for y = ymin to ymax
  for x = xmin to xmax
    u = (ax+by+c)/(gx+hy+1)
    v = (dx+ey+f)/(gx+hy+1)
    copy pixel at source (u,v)
      to destination (x,y)
```

References.

SUNY Stony Brook – Visualization Lab – Eye Tracking
http://www.cs.sunysb.edu/~vislab/projects/eye/

Shriver Center – Eye Tracking Laboratory
http://www.shriver.org/Research/Psychological/EyeTracking/index.htm

Vision Control Systems
http://www.visioncs.com

Arrington Research
http://www.arringtonresearch.com

TI website.
http://www.ti.com

Camera and SDK
http://www.xirlink.com

Calibration Code
http://www.cs.rochester.edu/u/www/u/kyros/courses/CS290B/Lectures/lecture16/sld018.htm