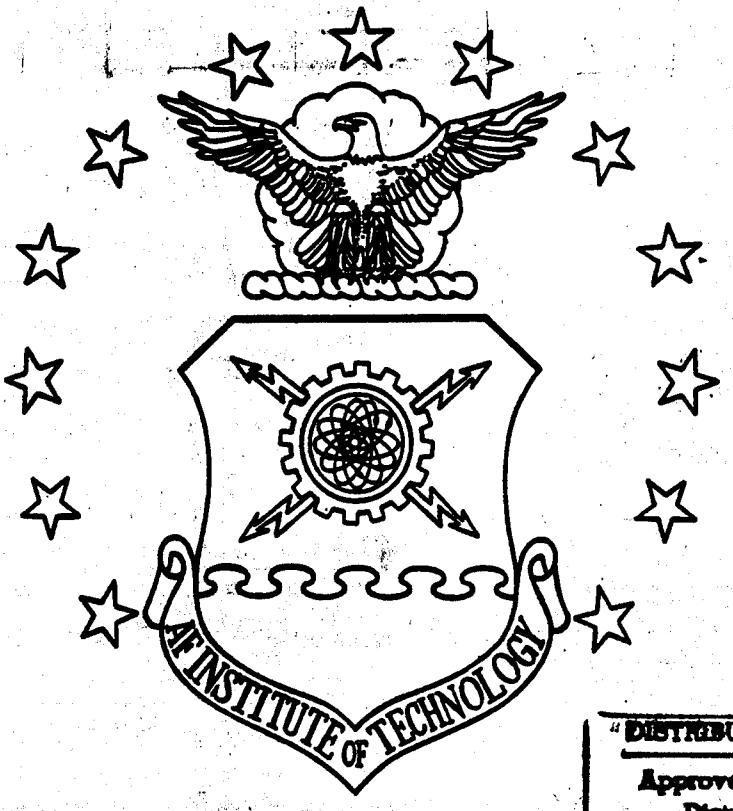
12-1995

Temporal Influence on Awareness

Don E. Hill



TEMPORAL INFLUENCE ON AWARENESS

THESIS

Don E. Hill
Captain, USAF

AFIT/GE/ENG/95D-07

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/95D-07

TEMPORAL INFLUENCE ON AWARENESS

THESIS

Don E. Hill
Captain, USAF

AFIT/GE/ENG/95D-07

AFIT/GE/ENG/95D-07

TEMPORAL INFLUENCE ON AWARENESS

THESIS

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Don E. Hill, B.S.

Captain, USAF

December 1995

Table of Contents

List of Figures

List of Tables

Acknowledgements

I wish to thank my research advisor, Dr Steve Rogers, for his expert advice and assistance. His easygoing manner and helpful pointers were much appreciated. I cannot imagine a better research advisor here at AFIT. I also wish to thank my readers, Dr Matthew Kabrisky and Dr Eugene Santos for their insightful comments and suggestions.

I wish to thank my fellow engineering classmates for making the lab an interesting place in which to work. I especially wish to thank my research partner, John Rosenstengal. His energy, sense of humor, and engineering insight helped make this effort the success that it was. I'm not sure anyone could make this process "fun", but he made this process as fun as it could possibly be.

I wish to thank Beth and Tony Ervin, my sister and brother-in-law, and Roger and Julie Hill, my brother and sister-in-law, for the much needed support and understanding they provided.

I wish to thank Sue, a very special person in my life, for the patience she demonstrated throughout this last year.

I thank my parents, Willie and Mary Hill, for believing in me even when I doubted myself.

Finally, I thank God for giving me this opportunity despite my own shortcomings.

Don E. Hill

Abstract

Grossberg's Motion Oriented Contrast Filter(MOC) was extensively analyzed(7). The output from the filter's "global motion" neuronal layer was compared to a noncausal post-processing filter developed by AFIT. Both filters were shown to incorporate a weighted, noncausal temporal range of input data in processed output. The global motion framework was then implemented using a physiologically motivated pulsed neural model – the Pulse Coupled Neural Network(PCNN). By incorporating both spatial and temporal data, the PCNN was shown to exhibit a common visual illusion, apparent motion. The existence of a physiological temporal processing range was further investigated through implementation of two multi-modal experiments which integrated visual and auditory stimulus input channels. Results from the first experiment reinforce earlier findings from literature of a temporal window for perception of simultaneous activity. (Events occurring within this window are considered simultaneous; events which span more than one window are considered temporally separate.) Data collected from the second experiment suggests future inputs from an accessory auditory stimulus impact current perception of a visual stimulus. The influence of the auditory accessory stimulus decreases as the temporal delay between visual and auditory stimulus presentation is increased up to a maximum value of approximately 40 milliseconds. These tests results suggest the existence of perceptual noncausality in the mind – awareness as a function of past, current, and future perceptual inputs.

TEMPORAL INFLUENCE ON AWARENESS

I. Introduction

1.1 Background

Apparent motion – the illusion of continuous motion when presented with a series of still frames, is well known among neuro-scientists, who seek to understand its origins, as well as Hollywood directors, who use its effects to entertain audiences(8). Due to recent advances in the study of human cognition, an increased interest in the study of apparent motion has emerged. If two spatially translated dots on an otherwise blank screen are flashed in temporal succession, individuals observing the dots will experience the sensation of apparent motion if 1)the frequency of the flashing dots is sufficiently high, and 2)the distance between the two dots is sufficiently small(8). Such an observance is significant for the following reasons: Individuals perceiving apparent motion see one dot move from the point of the first flashing dot to the point of the second flashing dot (figure 1). Since, in reality, no dots were flashed in the time between the first dot turn off and the second dot turn on, the mind had to fabricate the perception of a dot (or series of dots) in time and space (between the dots) based on both future (the second dot) and past (the first dot) inputs through the visual system. As a result, the mind appears to work in a noncausal fashion; that is, it seems to produce outputs, the perception of the in-between dots, which are not only the result of previous inputs (the first dot), but future inputs as well (the second dot).

In a related visual illusion, a brief flash of colored light followed immediately by a second flash of a different color at the same location may result in the perception of a *single* flash of a *third* color

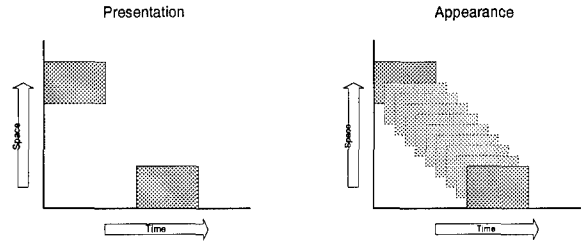


Figure 1. Apparent Motion(shaded area corresponds to stimulus on at specific space-time location)

(5)(figure 2). The perception of this single "apparent color" is affected by 1)the intensity of each color presented; 2)the relative presentation location for each color; and 3)the temporal length of each flash. Once again, the mind appears to fabricate an output based upon a temporal sequence of inputs. More importantly, perhaps, is the apparent manipulation of time. The brain somehow receives two separate temporal events and squeezes their perceived occurrence into one solitary perceived event. In effect, neurological processing appears to incorporate a temporal processing window to determine the inputs to "current" awareness. Incorporation of such a window allows the brain to preserve and maintain a stable and consistent internal representation, or "world model," of an external environment.

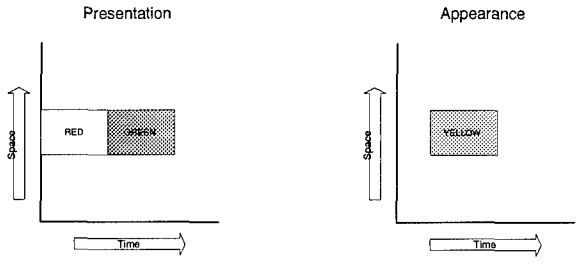


Figure 2. Apparent Color

Taken alone, the occurrences of apparent motion and color appear to be little more than neurological slights of hand. However, an understanding of why and how the illusions occur may

provide the basis for a model of higher brain activity. Specifically, an understanding of how the brain integrates the spatial and temporal aspects of data inputs might one day lead to a rudimentary model for consciousness, the Holy Grail of modern neuroscience. The potential benefits to be gained from the integration of an artificial consciousness into existing technologies provide the impetus behind the involvement of the USAF in the search for a solution to the problem of modeling apparent motion and color. If such a solution is found, then the basis for much higher level artificial thought may be formed, thereby giving new meaning to the moniker "smart machines".

1.2 Problem Statement and Scope

The temporal aspects of visual illusions will be investigated to gain insight into a general understanding of the origins of conscious processing. Analysis will be concentrated around 1)investigation of temporally-weighted processing filters developed separately by Dr. Stephen Grossberg at Boston University ((7)(12)(13)(11)) and AFIT (27); 2)simulation of this temporal filter concept using a physiologically-motivated Pulse Coupled Neural Network; and 3)implementation of a series of multi-modal physiological experiments to quantify the existence of a sliding temporal window in perceptual processing.

1.3 Thesis Organization

The following chapter will introduce the underlying concepts of Dr Stephen Grossberg's Motion Oriented Contrast filter(MOC) as they relate to the temporal processing of visual illusions(7)(12)(13)(11). The foundation for the filter, the presence of a "Global Motion Layer," will be compared to the sliding non-causal temporal window filter developed by AFIT (27) to post-process pristine and FLIR moving images. In Chapter III, a pulse based neural network architecture will be introduced

and used to simulate the temporal processing properties of a Global Motion layer. The ability of such a network to reproduce the apparent motion phenomenon will also be analyzed. Chapter IV describes the physiological motivation and implementation of two multi-modal time-dependent experiments. The first experiment demonstrates the existence of a perceptional window for "current time." The second experiment provides evidence that event occurence is temporally weighted such that past and future events may affect perception of a current event. Finally, Chapter V will discuss the conclusions which can be drawn from this work.

II. Noncausality in the Perception of Motion

2.1 Background

In 1992, AFIT developed a neural network retinal model to segment motion using spatio-temporal inputs (27). The AFIT model, heretofore referred to as the Adaptive Retinal Model, used as its basis a model of the fly's visual system developed by Ogmen and Gagne and added to this model two-dimensional adaptive input nodes (25)(27). In the course of the research, AFIT found that output data obtained from the model clearly demonstrated motion in one dimension; but this ability was severely diminished when the model was extended to two dimensions. (The additional dimension required that output be represented as a series of images – the result being a loss of the features in the response among the 256 gray levels used to represent the cell activity level (27)). In order to focus on the transient behavior of the input, (and thereby minimize the model's loss in performance), two post-processing techniques were developed: moving average subtraction and thresholding.

AFIT implemented four versions of the moving average subtraction techniques. They included the causal and non-causal level-window average and a causal and non-causal gaussian window average. An illustration of these moving averages is shown in figure 36(27). The nine images shown represent nine frames of output spanning a "window" in time centered on the image currently being processed. The moving average subtraction methods were described as follows:

> The moving average process calculates an average for each pixel and subtracts the current pixel value. In the causal moving average, the averaging window extends only into the past. No future data is needed in this process and the window can be level-windowed or gaussian in shape. The noncausal moving average uses either window type but adds as many future images as past images. This scheme requires the model to delay its output until it can calculate the current frame value (27).

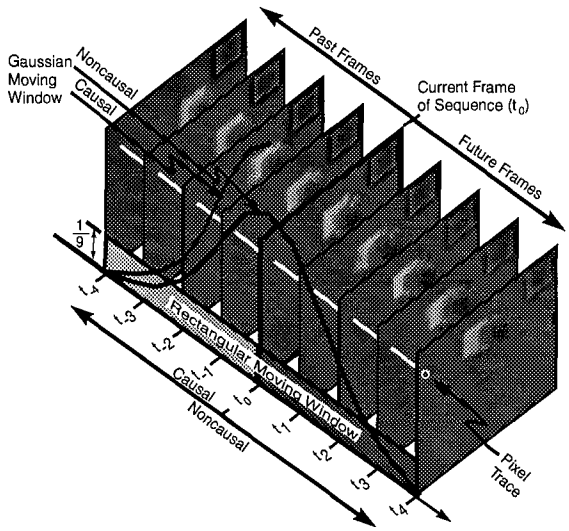


Figure 3. Adaptive Retinal Model Post-Processing (Moving Averages)

AFIT applied both techniques to the output generated from a one-dimensional simulation in which a large moving rectangle is passed over a smaller stationary rectangle. The input and output from the adaptive retinal model, prior to post processing, are shown in figures 4 (27) and 5 (27), respectively. Notice the ridge and dip in the forward wave when the two rectangles cross. The effects of the

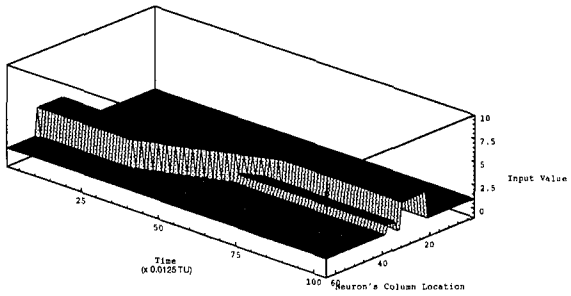


Figure 4. Adaptive Retinal Model Input

causal, level-window moving average on the retinal model output can be seen in figure 6 (27). As expected, AFIT observed a severely diminished response in the plateaus, mesas, and rippling of the

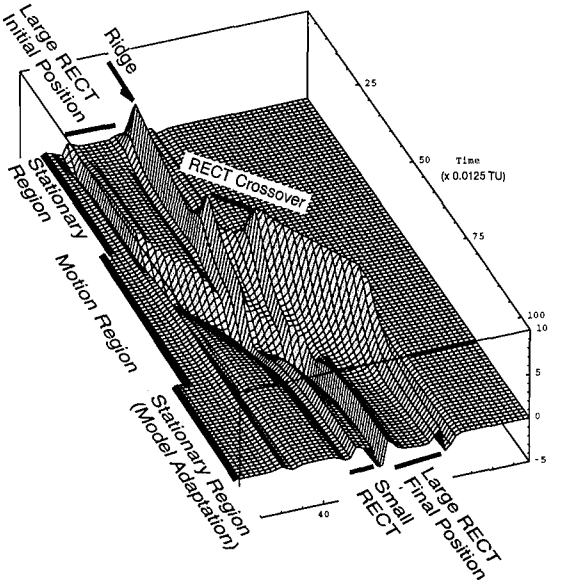


Figure 5. Adaptive Retinal Model Output

output. Unfortunately, while the overall response was decreased, the initial transient, forward, and trailing waves were enhanced. In contrast, the output from the non-causal, level window moving average (shown in figure 7 (27)) provided a much clearer indication of motion. Specifically, the trailing wave was inverted – resulting in a distinguishable plateau between the forward and trailing waves. Outside of the motion region, the background plane remained nearly plane. When the second post processing technique, thresholding, was applied to this output, the leading and trailing edges of the moving rectangle were clearly shown (figure 8 (27)). The post-processing described above referred only to causal and noncausal level-windowed versions. AFIT later applied all four versions of the moving average window to two 2D image sequences: a computer-generated pristine image sequence, and a real Forward Looking Infrared Radar(FLIR) image sequence. Examples of both image sequences are represented in figures 9(27) and 10(27), respectively. (Figure 9 also

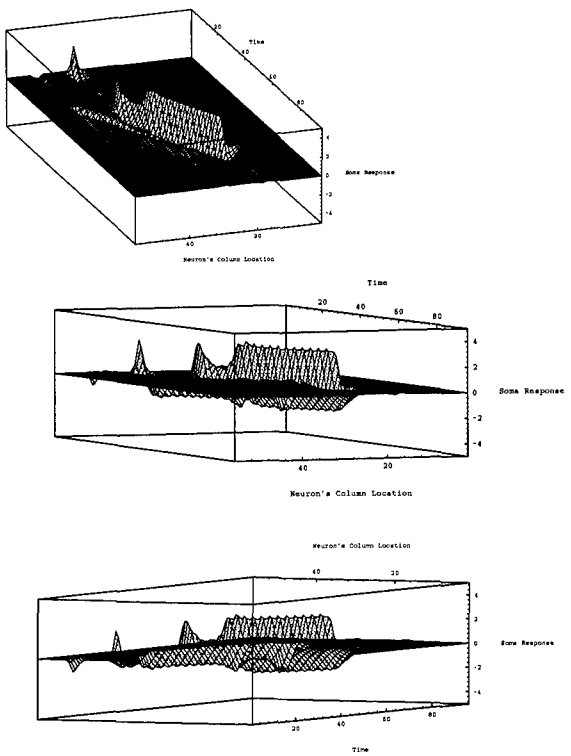


Figure 6. Post Processing: Causal, Level-Window Average (transient, leading, and trailing edges shown)

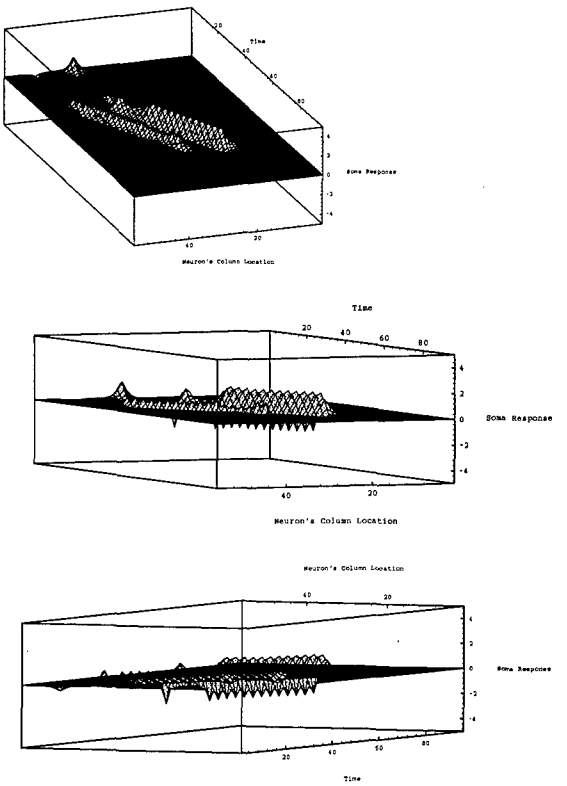


Figure 7. Post Processing: Non-Causal, Level-Window Average (transient, leading, and inverted trailing edges shown)

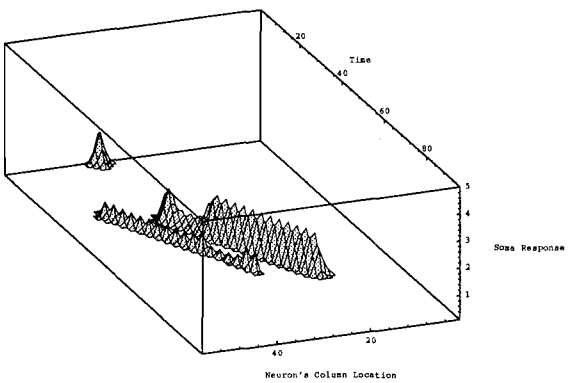


Figure 8. Post Processing: Thresholding (leading and trailing edges of moving rectangle shown)

includes example sequences for the image input and O&G model output for sake of comparison.) Using the same post-processing techniques described earlier, AFIT concluded that the non-causal, gaussian-windowed, moving average produced the best response for both the computer generated image sequence, as well as the FLIR image sequence. Specifically, AFIT observed: "The causal filter shows gradients of the output imagery based only on *past* information, while the noncausal filter adds *future* data to give *direction* information(emphasis added)(27)." From a performance standpoint, AFIT demonstrated that the integration of a non-causal window greatly enhances the calculation of form and motion in time.

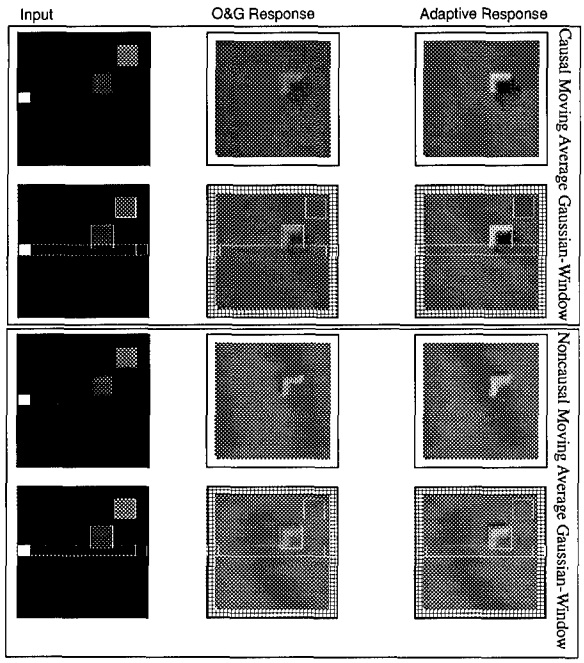


Figure 9. Computer Generated 2D Pristine Image Frames(input, two-dimensional O&G model output, and AFIT Adaptive model output shown)

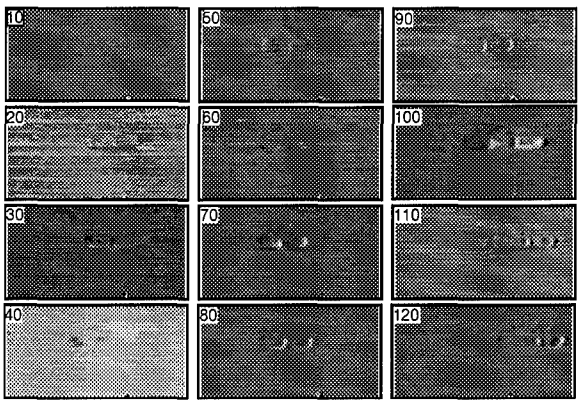


Figure 10. Forward Looking Infrared Radar(FLIR) Image Frames (output from noncausal moving average post-processing of Adaptive Model)

2.2 Noncausality in Grossberg's Oriented Contrast Filter(MOC)

Although never referred to as such, Stephen Grossberg implements a perceptual non-causal filter as part of the level 5 processing of his Motion Oriented Contrast Filter. Through implementation of spatially broad receptive fields, level 5 cells receive inputs from localized groups of lower level cells. These inputs precede "perception" of visual activity at corresponding retinal locations and can be described as "perceptually non-causal." Before a discussion of these non-causal properties can proceed, however, a brief graphical review of the filter's first four levels is presented.

2.2.1 Generating Motion Signals at Level 4 Neurons. Grossberg originally developed his Motion Oriented Contrast Filter(MOC) because the previous version, the Oriented Contrast filter(OC), was insensitive to direction of motion(7). To produce the necessary motion sensitivity, the retinal input (level 1) is used as inputs by both sustained cells (level 2) and transient cells (level 3) to produce sustained and transient responses at each neuron "position". One way to visualize such an arrangement is shown in Figures 11 and 12. Figure 11 represents the spatial relationship between level 2 sustained cells and level 1 input responses. For the sake of clarity, the

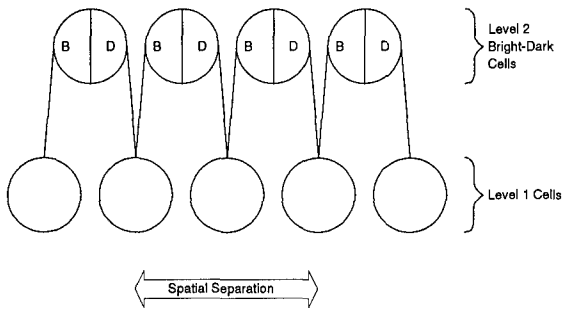


Figure 11. Interaction Between level 1 cells(retinal photo-receptors) and Bright-Dark level 2 (sustained) cells

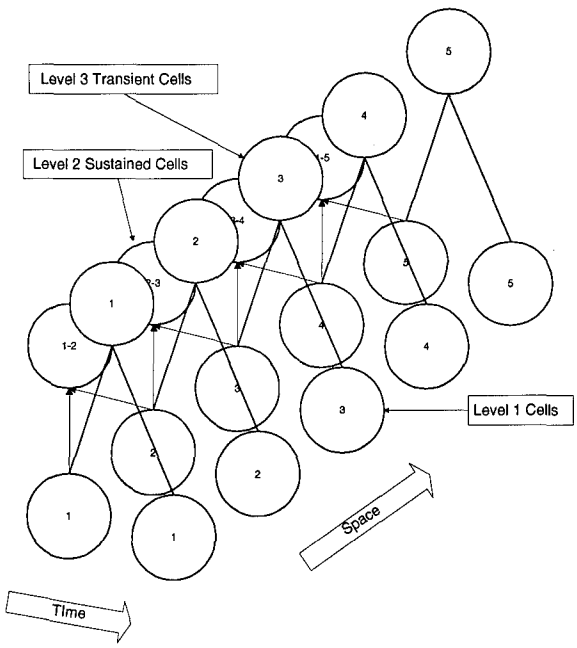


Figure 12. Interaction Between level 1 cells(retinal photo-receptors) and Transient-On level 3 cells(with space-time receptive fields)

relationship is shown in one dimension only. In addition, the level 2 cells shown represent only those cells sensitive to spatial Bright-Dark changes in luminance. To fully represent level 2 cells, an additional set of cells sensitive to spatial Dark-Bright changes in luminance must be included. A Bright-Dark Level 2 cell shown in Figure 11 will produce a response if its respective level 1 cells produce excitatory inputs for the "Bright" side of the level 2 cell and inhibitory inputs for its "Dark" side(12). Dark-Bright level 2 cells respond under complementary brightness conditions. Since the illustration shown is limited to one dimension, the result of such a setup is a network responsive to relative changes in luminance among neighboring level 1 cells(7).

Figure 12 represents the temporal relationship between level 3 transient cells and level 1 input responses. Level 3 transient on-cells (shown in the illustration) respond if the level 1 cell at time $t+1$ is greater than the activity of that cell at time t. Similarly, transient off-cells respond if the activity of their corresponding level 1 cells decrease over time. The level 3 representation shown in figure 12 is greatly simplified for ease of explanation. For an in-depth explanation of the transient stages required for accurate representation, refer to Stephen Grossberg's complete presentation of the MOC filter(reference(7)).

At this point, level 2 and level 3 cells contain information as to the potential direction of contrast, as well as transient behavior of the response. By gating the level 2 and 3 outputs with each other, a level 4 "local motion" signal is created(7). A representation of this gating is shown in figure 13.

2.2.2 Noncausality in the MOC: Calculating Level 5 Cell Inputs. As was previously stated, Grossberg's goal in designing the Motion Oriented Contrast Filter was to produce a filter whose outputs would be motion signals which could be competed in a competition-cooperation loop

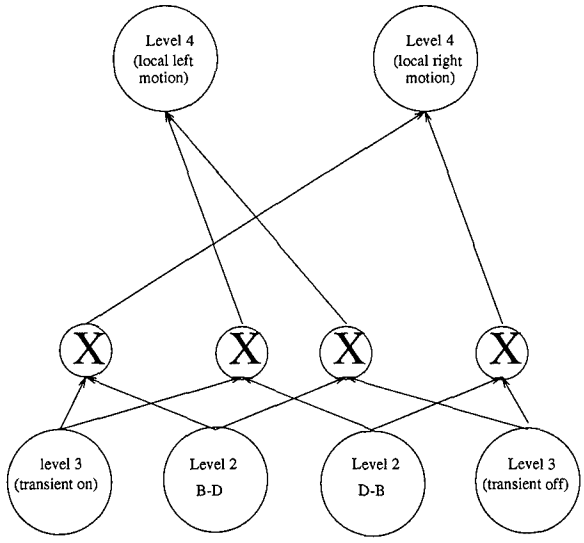


Figure 13. Gating of Level 2 Sustained Cells with Level 3 Transient Cells

analogous to the original Boundary Contour System(12). The cells are "global" in the sense that the spatial "footprint"(ie., receptive field) of level 5 cell inputs spread out across space. For the one dimensional view shown in figure 14, each level 5 cell represents a "global" position and receives inputs from level 4 cells across space at each time increment. If this example is expanded to two dimensions, the spatial footprints appear as a series of overlapping concentric circles corresponding to each global motion cell represented in the field (figure 15). The maximum "reach" of the footprint is dependent upon the inter-connectivity of the level 5 and level 4 cells. For the examples shown, the receptive field spans across 5 level 4 cells. The output of each level 5 cell is dependent upon

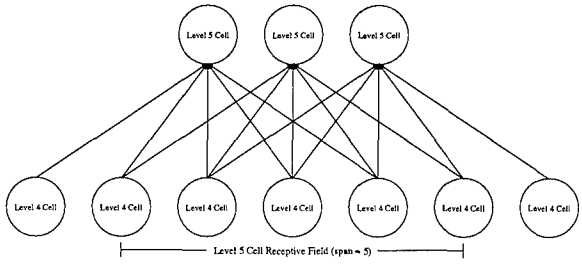


Figure 14. 1-Dimensional Representation of Level 5 Cells Receptive Fields

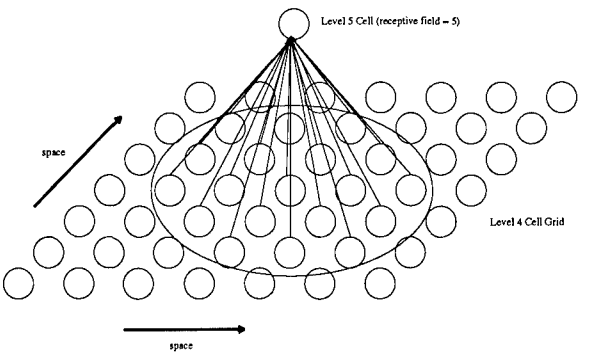


Figure 15. 2-Dimensional Representation of Level 5 Cells Receptive Fields

a summation of weighted inputs from the connected level 4 cells. Both Grossberg and AFIT use a Gaussian kernel to generate input weights. For Grossberg's MOC, this kernel is spatial; at individual time increments, inputs from different level 4 cells are multiplied by a distance-dependent Gaussian weight, and then summed together for input into a level 5 cell(7). In the previous AFIT research, the Gaussian profile was used in a temporal fashion. For each spatial position, the contribution of each temporal response is calculated by first scaling the response by a time-dependent Gaussian weight, then averaging the result obtained with other corresponding temporal responses at the same location(27). The weighting profile used here is crucial – the use of a linear or constant profile will not produce satisfactory results. The level 5 output generated using the weighted level 4 inputs is competed along with all other level 5 cells corresponding to the same motion direction in what Grossberg describes as a "winner take all" network(7). The result, then, is the excitation of one level 5 motion cell and inhibitation of those cells surrounding it.

Given the level 5 configuration described above, how does this model account for the non-causality of AFIT's post-processing model and, as a by-product, produce apparent motion? The answer to this question lies in the temporally-dependent cell activity level. Recall that the activity level (defined here as pulses/unit time) of a given cell does not rise or fall instantaneously; rather,

the activity level changes over time depending on both the current value of the level as well as inputs from other neurons. In addition, this cell activity level must cross a given threshold before producing an output. As a result, a temporal delay, $delta_t$, exists between the onset of a stimulus and the time that a given neuron "acknowledges" the stimulus input by outputting a pulse (figure 16). For purposes of this discussion, the crossing of the internal cell threshold will designate "perception" for higher-order processing. With this definition in mind, two separate time axis now exist: real time and "perceived" time. Real time denotes passage of time external to the brain; ie., the time when an event occurs. The rise and fall of the internal cell activity level is a function of real time – inputs to the level 5 cell contribute to the internal cell activity level as they enter the synapse. In contrast, perceived time refers to the passage of time experienced or acknowledged by the brain. Until the occurence of an event produces a cell activity level above the internal perceptual thresholds described earlier, no awareness of that event is experienced in the brain. As a result, the perception of a stimulus is a function of real time, t_r, plus some internal event processing time, $delta_t$, which may vary depending upon the type and number of feeding inputs. Since passage of internal perceptual time is referenced to awareness of a stimulus, the $delta_t$ processing time is nonexistent on the perceptual axis.

For a unitary input at level 5, (that is, only one level 4 neuron firing during an arbitrary window in real time), the level 5 cell will fire upon reaching its internal threshold. Here, the perception threshold is only dependent upon one level 4 input so the "perception" of this input occurs at $t_r + delta_t$. However if this single stimulus is removed and another stimulus added at a later time at a different location, a different scenario arises(figure 17). The removal of the stimulus does not simultaneously remove its effects upon the level 5 cell. Rather, cell activity in level 5 will begin to decay over time. As it does so, the activity may fall below the "perception"

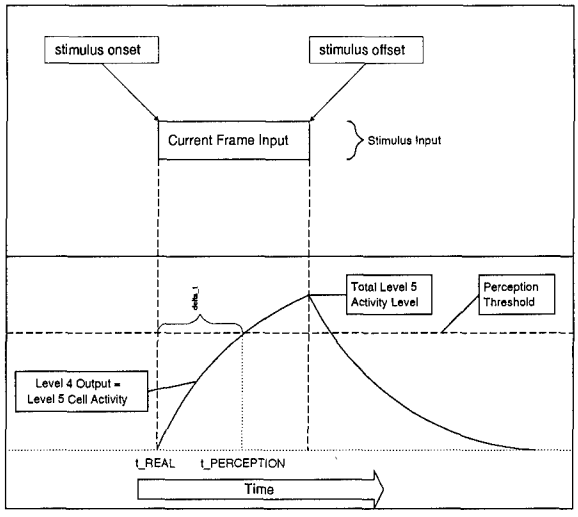


Figure 16. Representation of temporal delay, $delta_t$ between stimulus onset and perceptual onset

threshold – thereby preventing output. Note: the activity level may not be zero; it may simply be below the perception threshold. In a similar manner, cell activity from a level 4 input cannot instantaneously force a level 5 cell to output, that is, produce a signal which is "perceived" by higher processing. As was the case for the previous stimulus turning off, a new stimulus turning on causes a level 4 cell to increase its affect upon a level 5 cell over time. Once again, this input may or may not enable the level 5 cell to reach threshold. What happens, then, if the circumstance arises whereby a stimulus turning on and another turning off occur in such a way as to produce increasing and decreasing cell activity levels which are simultaneously fed into a level 5 cell? The answer is perceptual noncausality, and, if the conditions are "right", apparent motion. Until a level 5 cell exceeds its threshold, nothing is perceived to occur. Similarly, perception of a stimulus is maintained as long as the threshold is exceeded. Below this threshold, however, remnants of both "old" stimuli and new "stimuli" may (after undergoing gaussian spatial filtering) add together to produce a result which is above the perception threshold and, therefore, perceived. This result is crucial: since some of the stimuli are "old" in the sense that they occurred in the past, they are

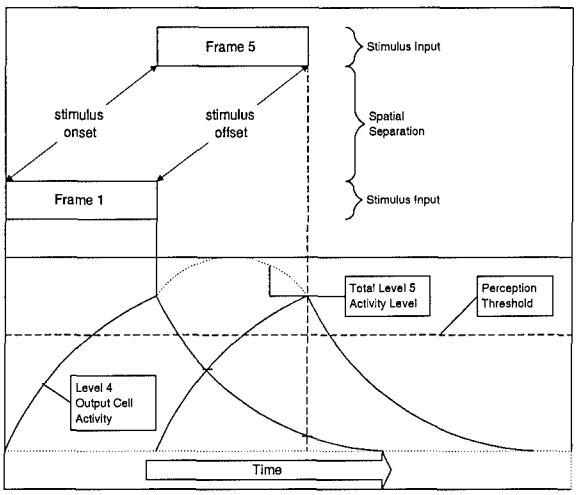


Figure 17. Temporal Overlap of 2 Stimulus Inputs at Level 5

now past their maximum cell activity level so any perception associated with these stimuli may have already occurred. In a similar fashion, level 4 cells responding to a new stimuli have not reached their full activity level and may not yet be perceived at level 5. Conceivably, the result of the scenario described above could be the following: at some time t_{p1}, a stimulus response is perceived at a level 5 cell(this is the first actual stimulus response); after some perceptual time t_{p2}, a second stimulus response occurs at a different level 5 cell location where no actual stimulus occurred(this response is apparent – no stimulus initiated it); finally, the last stimulus response occurs some perceived t_{p3} later at yet another level 5 cell location(once again, this is an actual response to a real stimulus). The combination of rising and falling activity levels produces a false response which, when perceived, appears to be a function of the earlier and later signals. Based on perception, the result is noncausal. Simulation of apparent motion using the MOC filter is shown in Figure 18. For additional examples of MOC output, see appendix B.

The interaction of rising and falling level 4 cell activity level inputs at level 5 cells can be tied back to AFIT's non-causal window (see Figure 36) by viewing the contribution of individual

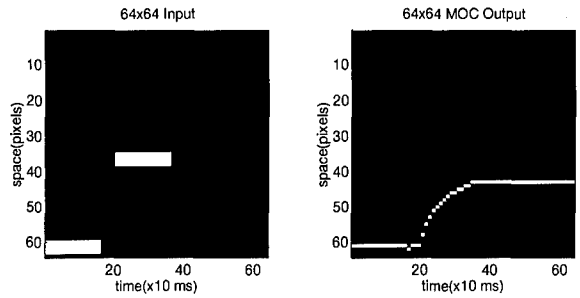


Figure 18. Simulation of Apparent Motion using Grossberg's MOC Filter

cell activity levels which are summed together at level 5 cells. A stimulus analogous to AFIT's time frame t_{-4} has a negligable effect upon the output perceived at frame t; the cell activity generated by the earlier frame is approaching zero by the time the snapshot at time t is taken, so its contribution/effect upon stimulus perception at time t is negligable. Similarly, the cell activity level of the stimulus which will be perceived at frame t_{+4} is still small at perceived "current" time t, so its contribution to level 5 cell activity (and therefore, perception) at this time is small.

To demonstrate the similarity between the Grossberg and AFIT models, a series of stimuli profiles are input into a MOC filter. The stimuli, shown in figure 19, are each presented for 160 ms (simulated), and are separated by gaps of 210, 170, 130, 90, 50, and 10 ms. For purposes of comparison the midpoint between each stimulus' offset is set equal to t_0, current time. (This is the same format used to explain the AFIT model in figure 36). Using this time frame, the offsets for each stimulus profile occur at $t_{\pm 180}$, $t_{\pm 160}$, $t_{\pm 140}$, $t_{\pm 120}$, $_{\pm 100}$, and $t_{\pm 80}$ ms, respectively.

The MOC output at t_0 for each of the six input profiles is plotted in figure 20. As was the case with AFIT's noncausal Gaussian window, the effect of a stimulus input upon level 5 cell activity decreases as its temporal proximity to t_0, current perceptual time, increases.

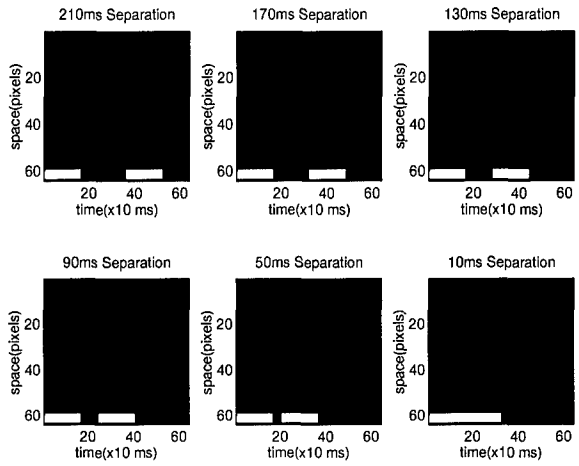


Figure 19. MOC Input Profiles

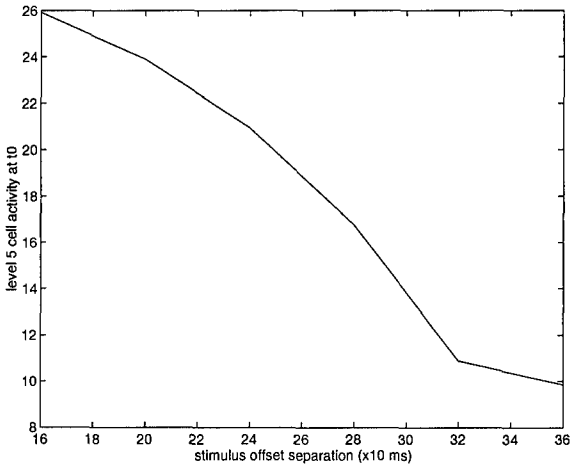


Figure 20. Effect of Temporal Stimulus Separation on Level 5 Activity at t_0

2.3 Conclusion

Neither AFIT nor Grossberg set out in their respective research efforts to investigate the presence of a perceived noncausality in motion processing. Yet, as was demonstrated in the preceding pages, the presence of a noncausal element provided a necessary link in the understanding of visual processing for both research efforts. In the case of Dr. Grossberg, the perception of noncausality forced any would-be explanation of apparent motion to account for the binding of past, present, and future inputs. For AFIT, the incorporation of non-causality in retinal model processing was due less to physiological accuracy than it was to bottom line performance: AFIT settled upon a system which worked. The work performed here has served simply to highlight the non-causal perspective for both Grossberg's work, as well as AFIT's Adaptive Retinal Model. In light of the fact that researchers, led by the likes of Dr. Benjamin Libet, have begun to uncover new evidence as to the existence of subjective temporal processing within the brain, it should come as no surprise that the best model for human visual processing is... human visual processing(22).

III. Simulation of Apparent Motion in a Pulse Coupled Neural Network

3.1 Background

As discussed previously in Chapter 2, AFIT developed an adaptive retinal model to segment simulated and real FLIR moving images (27). During the course of data post-processing, optimal segmentation occurred as the result of a non-causal moving average window. That is, the model produced its best output if post-processing utilized both past and "future" time slices when calculating the current output seen by the model. In a related development, Dr Stephen Grossberg, as part of a continuing effort to develop a biological foundation for the phenomenon of apparent motion, introduced a Motion Oriented Contrast filter(MOC) for use in a motion Boundary Contour System (7). In general, the MOC filter extended Grossberg's concept of competition/cooperation among orientation signals to local motion signals. Each local motion cell was the result of successive layers of oriented sustained and transient cells whose internal activity levels both increased and decreased according to the following generic differential equation:

$$\frac{dx}{dt} = -Ax + (B - x)I \tag{1}$$

where A is the passive decay rate parameter, B is the maximum activity level of the cell, and I is the resulting summation of weighted inputs. The resulting local motion signal responses were then fed into a grid of corresponding global motion cells (each receptive feeding field spanned the spatial dimensions of the lower cell layers – hence the term "global"). The strength of each feeding input was determined by the spatial distance from the global motion cell to each lower local motion cell. In what Grossberg described as a winner-take-all competitive network, the global motion cell possessing the maximum input population was determined to be the "winner"(7).

Viewed temporally, this competitive framework produced localized responses in the global motion cell layer which corresponded to movement of a stimulus across the cell grid. Grossberg and Rudd demonstrated that a temporal overlap of decaying and growing inputs from spatially separated local motion cells could produce a global motion cell with maximum response between the two stimulus inputs.

The foundation of Grossberg's MOC filter, a "Global-Motion Cell Layer" whose inputs are separated in both time and space, is integral to the formulation of a theory which might be used to explain a wide range of perceptual illusions. However, any demonstration of this theory is contingent upon the underlying structure of the neurological unit used in its implementation. Throughout his research, Dr. Grossberg demonstrates MOC operation using a simplified neuron model whose inputs and outputs are continuous. Internal cell activity is a product of an internal "leaky integrator" whose inputs are continuous outputs from lower level cells. Once the internal cell activity level exceeds a given threshold, the neuron initiates an output signal (action potential) which is proportional to the aforementioned cell activity level. Under this model, then, cell output may increase or decrease depending upon the state of its inputs.

Such a model would prove adequate if the "global motion" layer described by Grossberg was thought to exist in the retina. There, neurons in the first three layers (cones/rods, horizontal cells, and bipolar cells) have been shown to exhibit a continuous output characteristic(31). However, Grossberg's research points to area MT (V5) of the visual cortex as the most probable location for the global interactions described in his work(13). In this region of the brain, research data has shown that neuron output is not continuous; rather, it is a series of pulses. When the internal cell activity level exceeds its respective threshold, the result is the release of neurotransmitter chemicals

which produce the characteristic action potential spike output. The action potential threshold is a function of the amount of neurotransmitter present in the synapse. As such, a "firing" depletes this resevoir and raises the activation potential threshold. After each firing, the synapse undergoes a refractory period, during which time the higher activation threshold allows time for the level of neurotransmitters in the synapse to be replenished. The net result is a single pulse (if the neuron inputs persist long enough such that their weighted sum exceeds the activation potential threshold) or a pulse train (if the neuron inputs continue to contribute to the internal cell activity level such that the activation potential threshold is repeatedly exceeded after successive synaptic refractory periods).

In fact, research performed by Gray, Konig, Engel and Singer, as well as Eckhorn et al. has shown that neurons in the cat visual cortex produce oscillatory responses in the range 40-60 Hz (9) (6). A meaningful examination of Grossberg's Global Motion Layer theory requires, then, that it be implemented using a pulse based neurological model. One such model which exhibits this characteristic and, in addition, possesses the capability to synchronize 'similar' pulsed output, is the Pulse Coupled Neural Network (PCNN).

3.2 PCNN Description

The Pulse Coupled Neural Network was conceived as a "..minimal model to explain the experimentally observed synchronous feature dependent activity of neural assemblies over large cortical distances in the cat cortex"(15). The model is composed of two types of functional synapses: feeding synapses, junctions found in the direct stimulus-driven signal path; and linking synapses, auxiliary signals from neighboring neurons which modulate the feeding synaptic inputs (6). The resulting modulated feeding input is fed into the third main component of the model neuron, the

pulse generator (or spike encoder (6)). All three components are modeled as leaky integrators (6). A graphical representation of a generic PCNN neuron is presented in Figure 21(2).

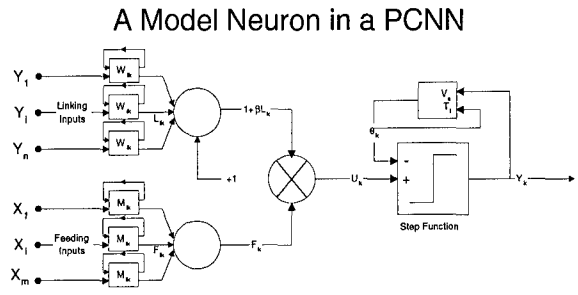


Figure 21. Pulse Coupled Neural Network

3.3 Model Operation

When an input pulse is received via a feeding or linking synaptic junction, the leaky integrator (which represents the internal activity level at that junction) is charged such that a rise in the output amplitude is realized. Since the input is a pulse and not a continuous signal, the sharp rise is followed immediately by an exponential decay of the signal (determined by a leakage time constant)(6). The sum of all feeding leaking integrator inputs is modulated by a combination of a "linking strength factor" and the sum total of all linking leaking integrator inputs. The resulting product represents the "membrane voltage" or internal activity level of the neuron(6). This voltage is compared against a variable internal threshold also represented as a leaky integrator. If the membrane voltage exceeds the internal threshold, an output spike is formed. When this event occurs, the internal threshold is recharged to an arbitrarily high value such that the membrane voltage cannot exceed the internal threshold during and immediately after the pulse output(6). During the interval in which the internal threshold remains above the membrane voltage, the neuron experiences a refractory period

– no pulse output can be generated. Once the threshold has decayed to a level at or below the membrane voltage, the pulse output sequence is repeated.

3.4 Model Implementation

Note: The equations which follow are representative derivations of work performed by Eckhorn, Johnson, Rosenstengel, and Broussard. They are provided here as a minimal foundation for a digital implementation of the Pulse Coupled Neural Network. For an in-depth analysis of the equations, please refer to the accompanying references(2)(26).

The output of a PCNN neuron can be represented as:

$$Y_j(t) = \begin{cases} 1 & \text{if } U_j(t) \geq O_j(t) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

(26) (2).Here, a pulse is produced only when the internal membrane voltage, $U_j(t)$, is greater than or equal to the internal threshold, $O_j(t)$.

The internal threshold, $O_j(t)$, is modeled as a leaky integrator with time constant τ and amplitude gain V^s(26) (2), and is a function of a threshold offset, $O_j^o(t)$, the previous threshold value, $O_j(t-1)$, and the neuron output, $Y_j(t)$:

$$O_j(t) = O_j^o(t) + O_j(t-1)e^{\frac{-1}{\tau^s}} + V^s Y_j(t) \tag{3}$$

(26) (2) The internal membrane voltage, $U_j(t)$, is a function of the total feeding input, $F_j(t)$, modulated by the product of the total linking input, $L_j(t)$ and a linking strength coefficient, β, such that:

$$U_j(t) = F_j(t)(1 + \beta L_j(t)) \tag{4}$$

(26)(2) The feeding input, $F_j(t)$, is a function of a single (optional) analog input, $I_j(t)$, and the weighted sum of synaptic inputs from other neurons(26):

$$F_j(t) = I_j(t) + \sum F_{kj}(t) \tag{5}$$

where $F_{kj}(t)$ is the synapse between the k^{th} feeding neuron and the j^{th} current neuron.

Each feeding input synapse, $F_{kj}(t)$, is modeled as a leaky integrator with τ^F time constant and V^F gain amplitude:

$$F_{kj}(t) = F_{kj}(t-1)e^{\frac{-1}{\tau^F}} + V^F X_j(t) w_{jk}^F \tag{6}$$

Here, $X_j(t)$ is the output of the j^{th} synaptic input and w_{jk}^F is the spatially determined Gaussian weight.

Similarly, the linking input, $L_j(t)$, is a function of the weighted sum of synaptic linking inputs from other neurons:

$$L_j(t) = \sum L_{kj}(t) \tag{7}$$

where $L_{kj}(t)$ is the synapse between the k^{th} linking neuron and the j^{th} current neuron.

Each linking input synapse, $L_{kj}(t)$, is modeled as a leaky integrator with τ^L time constant and V^L gain amplitude:

$$L_{kj}(t) = L_{kj}(t-1)e^{\frac{-1}{\tau^L}} + V^L Y_j(t) w_{jk}^L \tag{8}$$

Here, $Y_j(t)$ is the output of the j^{th} synaptic input of the current neuronal layer and w_{jk}^L is the spatially determined Gaussian weight.

3.5 Simulation of Neuronal Activity

One of the underlying foundations for the PCNN network is its ability to transform the luminant intensity of a spatial representation into a pulse-based temporal representation. To demonstrate this transformation, three point source inputs (figure 22) are entered into a PCNN with the operating parameters shown in Table 1. The first input, with luminance intensity $I = 0.2$, is

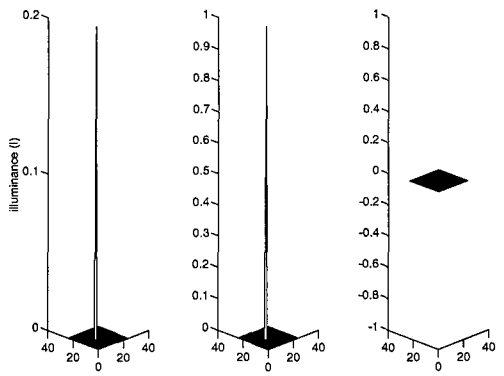


Figure 22. PCNN Inputs: a)I=.2 b)I=1 c)I=0

entered into the network at time 0 (iteration 1). At iteration 25, this stimulus input is removed and replaced with the second input, with a luminance intensity $I = 1.0$. Finally, at iteration 50, the second input is removed so that no stimulus is present for the duration of the simulation(input 3). The results of this ordered presentation of stimuli upon a PCNN input layer are shown in Figure 23. The information signifying the increased intensity seen at iteration 25 is conveyed through an increase in the pulse-rate output. Similarly, the removal of the same stimulus at iteration 50 produces a sharp decline in pulse-rate, followed shortly thereafter by an absence of any output.

3.5.1 Multi-Layer Interactions in the PCNN. The capability to add both feeding and/or linking inputs with an almost infinite number of weighting schemes to the basic underlying design

Table 1. PCNN Parameters

Parameter	*Value*
R_L0	0
$\beta 0$	0
R_F0	0
τ_L	5
V_L	5
V_F	5
θ_0^0	.1
θ_0^1	10
τ_θ	.5
V_θ	10
R_L1	0
R_F1	0
$\beta 1$	0

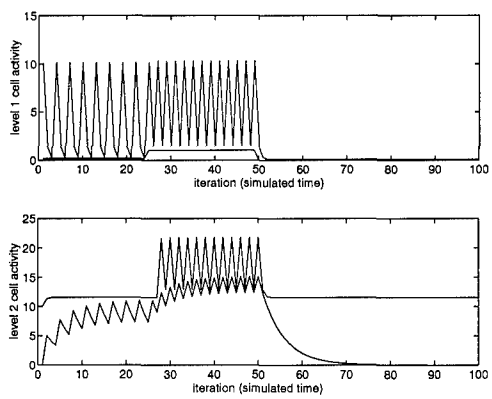


Figure 23. PCNN Representation of Intensity Variation

allows the PCNN model to function as a "universal filter" in regards to information processing. To illustrate this point, an input pattern of uniform intensity (shown in Figure 24) is fed into a 2-layer PCNN network. Initially, both the feeding and linking input radii are set equal to zero so that

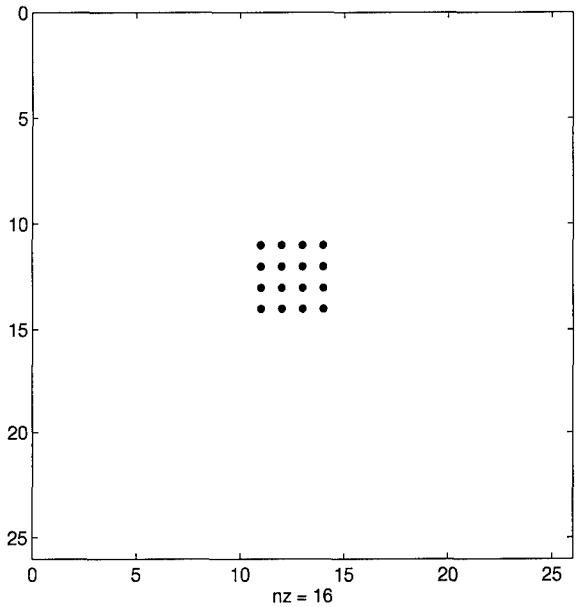


Figure 24. PCNN Solid Input Pattern

the upper layer neurons receive only pulsed outputs from the lower level neurons which correspond to the same spatial position in the network. The result, as shown in Figure 25, is a (repeated) simultaneous pulse output of those neurons with lower level inputs. The one significant difference between pulsed outputs from each layer is the inter-layer pulse delay, caused by the presence of a higher internal threshold, θ, in the upper layer. This delay is illustrated in Figure 26. If the feeding radius at level 2 is increased to 2 while the linking radius remains 0, the pulsed output from the lower layer no longer mirrors the lower input layer (Figure 27). Uniform pulsed output is now replaced by a series of pulses which reflect the degree of feeding inputs from lower level cells. If, conversely, the linking radius at level 2 is increased to 2 while setting the feeding radius to 0,

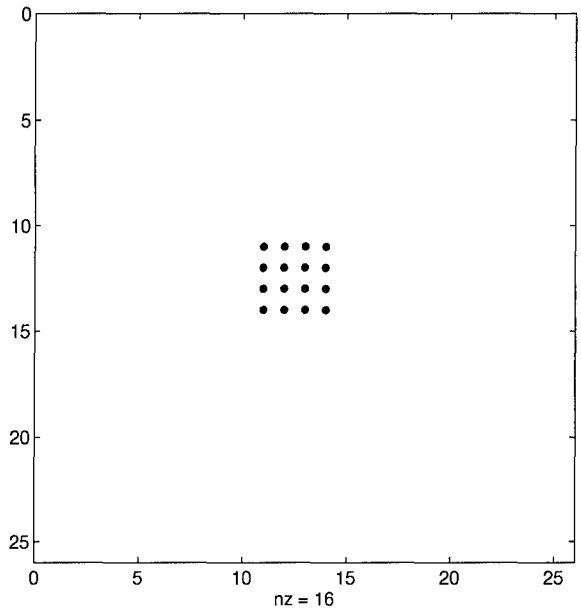


Figure 25. PCNN Level 2 Output (feeding and linking radius = 0)

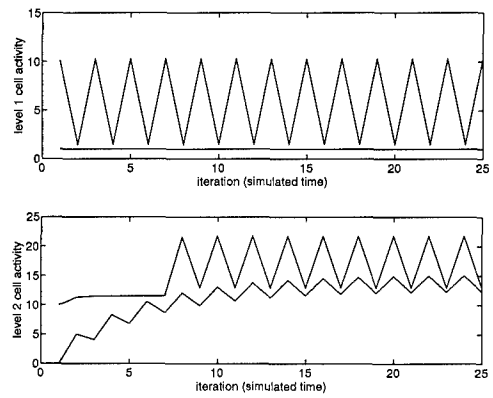


Figure 26. Delayed Pulse Propagation at Level 2 (level 2 neuron location (12,11))

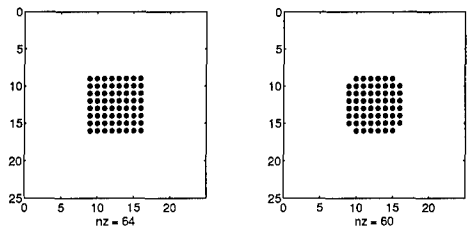


Figure 27. PCNN Level 2 Outputs(feeding radius = 2, linking radius = 0)

pulsed output at layer 2 changes once again. To illustrate this point, an input image containing two separate luminance intensities is input into a PCNN (see Figure 28). Without linking, the outputs from level one alternate between the two intensity levels, as can be seen in Figure 29. However, when linking is added to level 1, the two separate image intensities are close enough such that they "bind" together and pulse simultaneously. The result, then, is an output identical to the input in Figure 28.

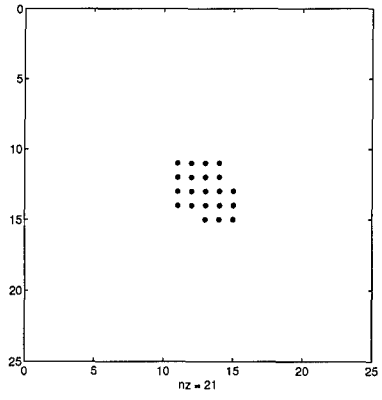


Figure 28. PCNN Input Pattern with multiple luminance intensity

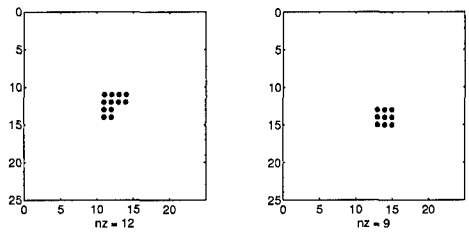


Figure 29. PCNN Output Patterns at level 1: luminance intensity = a)1 b).4 (no linking)

For all previous simulations, the weighting for all feeding and linking inputs has followed a positive gaussian profile. If a lateral inhibition weighting profile such as the following:

$$\begin{matrix} -.125 & -.125 & -.125 \\ -.125 & 1 & -.125 \\ -.125 & -.125 & -.125 \end{matrix}$$

is applied to level 2 neurons, the resulting interactions produce a series of "edge detectors", If the solid input from Figure 24 is fed into such a configuration, the output is a series of pulses corresponding to the pattern's outline, as shown in Figure 30.

3.5.2 Simulation of Apparent Motion. As was discussed earlier, Grossberg's "Global Motion Layer" operates under a "winner take all" scenario which utilizes a comparison of individual cell activity levels at discrete time increments. However, as was demonstrated in Figure 25, internal activity for a given cell is transferred to other cells through a series of output pulses. As a result, any demonstration of apparent motion using the PCNN model must utilize pulse rate information in its implementation.

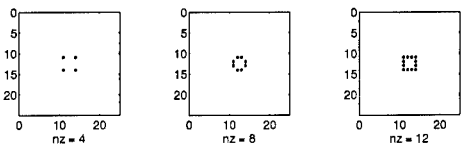


Figure 30. PCNN Level 2 Output(feeding radius = 1, linking radius = 0, lateral inhibition weighting profile)

To demonstrate apparent motion, two point source inputs (shown in Figure 31) are presented at time iterations $i = 1$ and $i = 50$ with the associated model parameters presented in Table 2. In this scenario, each input in the second PCNN layer has a receptive field equal to 9 such that its feeding radius is set equal to 4. For the simulation runs shown here, no linking inputs are used. A sample of the output pulses at iteration intervals i=20, 60, and 75, is shown in Figure 32. For a complete listing of the level 2 apparent motion output pulses, see Appendix D. As can be

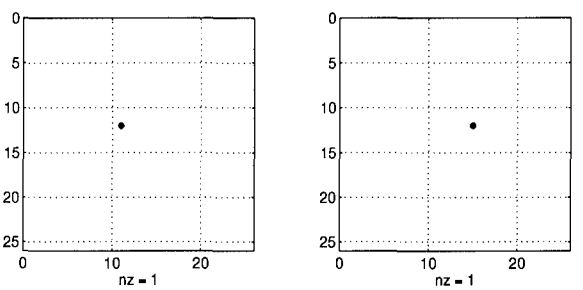


Figure 31. Apparent Motion Point Source Inputs: a)i=0 b)i=50

seen in the successive pulse outputs from level 2 neurons, output is not limited to those neurons which correspond to stimulated neurons at the input level. Instead, pulse output is present up

Table 2. Apparent Motion Simulation Parameters

Parameter	*Value*
R_L0	0
$\beta 0$	0
R_F0	0
τ_L0	5
τ_F0	.0005
V_L0	5
V_F0	1
θ_0^0	.1
θ_0^1	30
$\tau_\theta 0$	.5
$\tau_\theta 1$	1
$V_\theta 0$	10
$V_\theta 1$	50
R_L1	0
R_F1	4
$\beta 1$	0

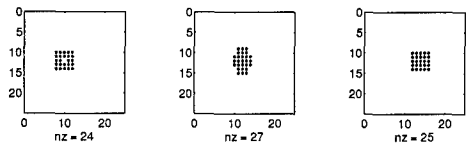


Figure 32. PCNN Apparent Motion Output: a)i=20 b)i=60 c)i=75

to a distance of 4 neurons. (This distance corresponds to the feeding radius for level 2 neurons.) The difference in the output is, as mentioned earlier, exhibited in the pulse rate of each neuron. Temporally, the center of pulsing activity at the output level (level 2) moves from the location of the first input to the location of the second input. Since pulse activity is centered briefly over lower level neurons whch are never stimulated, the simulation is interpreted to exhibit apparent motion.

3.6 Conclusion

In Chapter 2, the presence of a temporal delay was shown to play a major role in perceptual processing in the brain. Specifically, the delay was shown to allow both pre-attentive and post-attentive stimuli to have an influence upon perceived stimuli. The physiological basis for how this temporal delay might occur in a neurological framework was largely ignored. However, the Pulse Coupled Neural Network developed by Eckhorn et. al., and implemented here in Chapter 3, allows the physiological foundation for temporal delay to be re-addressed. Specifically, the PCNN has been shown to translate stimulus inputs into pulse-based temporal representations. As a result, variations in input frequency and amplitude translate to unique pulsed-code output. More important, however, is the ability of the PCNN to simulate the build-up of internal activity in individual neurons through the feeding and linking inputs of other neurons. It is this build-up, prior to pulsed output, which allows the PCNN to produce pulsed output from neurons which received no direct stimulus input (ie., apparent motion). Through modification and implementation of the model neuron, the PCNN has been shown to provide a physiological motivation for the temporal influence upon stimuli perception. In Chapter 4 which follows, the degree to which this temporal influence is present in both uni-modal and multi-modal sensory processing will be examined.

IV. MultiModal Noncausality

4.1 Background: Neurophysiological Timing in the Brain

Michael S. Gazzaniga, commenting upon the nature of the conscious experience in the existence of unique human mental capacities, once remarked,"..our awareness, our consciousness of these capacities, is nothing more or less than a feeling about them"(8) If such a premise is accepted, current neurological and psychological research can offer little more than conjecture as to the source and implementation of these "feelings" within the brain. However, the study of conscious/unconscious experience need not be shelved just yet. The "quality"(16) of consciousness may remain beyond the reach of current scientific capabilities, but its presence (or conversely, absence) has been documented through both physiological and psychological experimentation. Such research, principally the work of Dr Benjamin Libet and his associates, has provided a physiological foundation for the existence of a neuro-time factor in association with the presence/absence of conscious experience.

In 1963, Dr Libet presented results detailing the presence of a minimum threshold for conscious experience(21)(19)(18)(23)(22). The conscious threshold was determined by stimulating the somatosensory cortex with pulse trains at the liminal (threshold) intensity necessary for any sensation response(21). Evoked responses were measured for a wide variety of pulse stimuli. However, subjects experienced conscious sensations only when the applied pulse train durations equaled or exceeded approximately 500ms.(23) (Note: Libet stated that the 500ms temporal requirement was not rigid; stronger stimuli with shorter durations were also shown to be effective(22)). Single, long-duration pulses with increased amplitudes were unable to produce the conscious sensation(19). The significance of this result was threefold: 1)the presence of evoked potentials without the associated

conscious sensation provided a possible explanation for the existence of "subliminal perception" as well as the phenomenon of "blindsight" as described by Weiskrantz(18)(30); 2)the requirement for a 500ms time duration predicated the existence of a neuronal delay above and beyond the delays previously thought to exist due to pulse transmission and accumulation(18); 3)the need for a minimum duration of pulse activity added weight to the theory that the combination of the two factors (time, pulse activity) provided the coding for neural conscious activity(22). The Pulse Coupled Neural Network proposed by Eicken et al. and described in detail in Chapter 2, provides a promising realization for such a coding scheme. Based upon data collected from the striate cortex of a cat, the network encodes information through neuron pulse rates(6). (As an added benefit, the internal cell activity level for individual neurons can be shown to grow or decay over time as a product of feeding and/or linking inputs. As a result, conscious sensation can be shown to exist (ie., fire) only after unconscious/conscious feeding and linking inputs have produced the necessary internal cell activity level.)

Dr. Libet's discovery of a minimum conscious threshold and an associated pulse train duration led to an increased study of the impact of the neural time factor upon time perception. Specifically, he found that there was "..no appreciable delay for the subjective timing of a normally arriving sensory input" (20). Subsequent experiments utilizing relative ordering of subjective timing enabled Libet to conclude that the delay preceeding consciousness of activity was followed by an automatic subjective referral of the experience backwards in time to the approximate point when the stimulus originally occurred – the subjective present.(20)(4). According to Libet, a fast sensory message from the original cortical response provided the temporal anchor point(20). The result, then, was a response which subjectively occurred without delay. The diagram used by Libet to explain this phenomenon is reprinted in Figure 33(22).

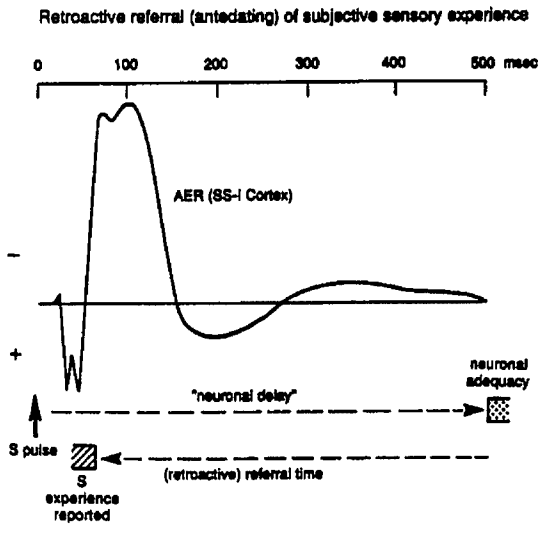


Figure 33. Subjective Backwards Referral in Time

4.2 Implications of Subjective Timing in Conscious Perception

4.2.1 Uni-modal Stimuli(uni-modal temporal succession). In the course of his research, Dr. Libet hinted at the possibility that unconscious experiences, constrained by a shorter latency for development, might have the ability to participate in and thereby affect cerebral activities prior to conscious awareness of these activities(18). Under such a scenario, two temporally separate conscious somatosensory stimuli could provide simultaneous inputs to higher level cerebral processing. Evidence for just such a situation was demonstrated back in 1970 by Richard and Roslyn Warren(29). As part of a larger research effort into the dynamics of audio perception, Warren and Warren found that the perception of words presented earlier in a sentence could be determined by the context of the latter half of the sentence(29). From this outcome, both scientists surmised that the ability to achieve correct phonemic restoration was the result of "...storing the incomplete information until the necessary context [was] supplied so that the required phoneme [could] be synthesized(29)." Acceptance of a "subjective backwards referral" operation produces a slightly

different interpretation of the Warrens' experimental results. If conscious awareness is delayed some time after occurrence of the audio stimuli, then the conscious perception of the stimuli may be influenced by other stimuli which occur during the delay. If the conscious awareness of the stimuli is then subjectively moved backwards in time, then the result would be conscious perception of a stimulus which differs from the original sensory stimulus input.

4.2.2 Multi-Modal Stimuli(multi-modal temporal succession). During the course of his research, Dr Libet noted that the subjective "backwards referral in time" capability provides a corrective mechanism for the temporal differences observed in the processing of simultaneous multi-sensory inputs(18). Without such a capability, stimuli-dependent transmission and processing delays would lead to a temporal sequence of conscious sensations(figure 34). By subjectively referring the conscious perception of the sensations back to their originating times(ie., the original cortical responses), a sense of simultaneous stimulation is achieved(Figure 35). What, then, is

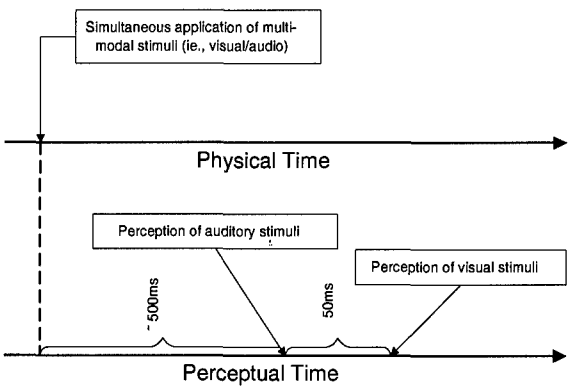


Figure 34. With-Out Subjective Backwards Referral in Time

the nature of stimuli perception if two stimuli are applied with some temporal delay? Research into this area has demonstrated the inability of test subjects to discriminate the order of unimodal stimuli applied in succession with very small temporal delays (on the order of tens of milliseconds)

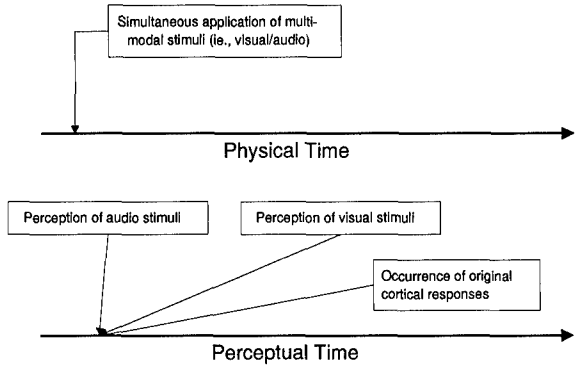


Figure 35. With Subjective Backwards Referral in Time

(29). In affect, the stimuli are perceived as having occurred in a simultaneous fashion. Such a response would seem to support the existence of a temporal perceptual window – multiple stimuli occuring anywhere within this window are processed in a perceptual sense as simultaneous. The maximum allowable difference in time between two (or more) stimuli, which would still result in simultaneous perception, would then define the temporal tolerance of this perceptual window. A possible graphical representation of such a window is represented in Figure 36. The implications for

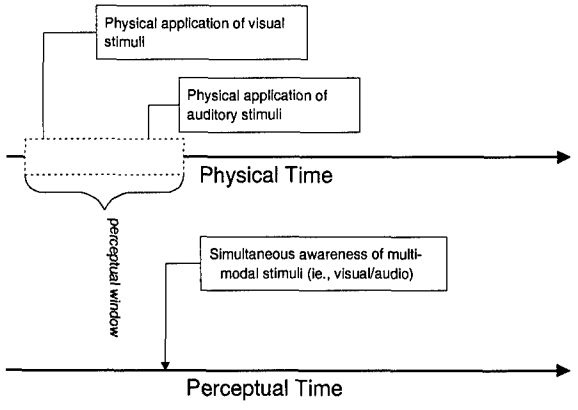


Figure 36. With Subjective Backwards Referral in Time

the existence of a perceptual window are tremendous. Information processing which utilized such information would, in effect, operate in a non-causal manner (with respect to current perception).

One such processing function, multi-modal sensory integration, has been shown to exist in the thalamus(nucleus dorsolateralis posterior thalami) of the pigeon (17),as well as the cortex of the cat anterior ectosylvian sulcus (28). Neurons within these areas respond to unique modal combinations. The presence of a sliding temporal window would alleviate the need for exact syncronization of multisensory inputs into these neurons and at the same time, greatly enhance the robustness of the overall system. To test such a hypothesis, a series of multi-modal experiments have been devised:

4.3 MultiModal Experiments

4.3.1 Materials and Methods. In the procedures to be described below, six AFIT engineering students were used as subjects in Experiments I, II, and II. All six students undertook each experiment individually, one sitting per experiment. Visual and auditory stimuli were provided to each subject via a monitor and headphones connected to a Zenith 486 computer using a 33Mhz processor with sound card installed. The subjects were instructed to sit perpendicular to the screen, approximately 8 inches from it. The visual display seen by each student was composed of a 640x480 pixel framed background, a 20x20 pixel black cross-hair in the center of the frame, and three graphical buttons in the lower right-hand side of the frame labeled "go", "left", and "right", respectively(Figure 37). Wearing headphones, subjects were instructed to fixate on the black cross-hair and select "go" with the attached "mouse." After a two second delay, two black 5x5 pixel boxes appeared on each side of the screen, separated by a spatial distance of 500 pixels(8.25 inches) and a temporal delay (in milliseconds) preset prior to commencement of the test. The spatial separation of visual stimuli was used as a means to force each test subject to use his/her peripheral vision when viewing the two black boxes. During an investigation of the

spatial distribution of visual attention, Mangun and Hillyard observed the existence of a gradient of attention which corresponded to the measured distance from focused attention to the location of the stimulus(24). This test observation was used as a means to sufficiently reduce attentional awareness of the two visual stimuli such that the test subject would, if the temporal delay was short enough, experience difficulty in selecting the stimuli which appeared first. In conjunction with the visual display, test subjects would hear one of three possible audio responses: 1)silence, 2)audio tones presented in each ear corresponding to the order of appearance of each visual block, or 3)audio tones in each ear presented in reverse order from the visual stimuli order of the black boxes. The degree of modal temporal synchronization , (ie., the amount of overlap of visual and audio stimuli in time) was determined by a second delay parameter set prior to initialization of each test. Upon conclusion of the visual/audio stimuli presentation, each subject was instructed to make a forced choice (left or right) as to which block appeared on the screen first. This scenario was repeated (in each experiment) for each student 60 times.

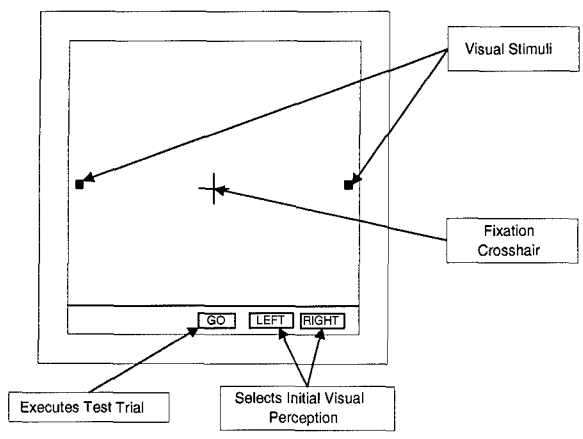


Figure 37. Visual Interface Seen by Test Subjects

Synchronization of visual and auditory stimuli was achieved using a software program written in the C program language implementing a series of application independent (API) Windows

routines. Utilization of the windows environment greatly simplified the graphical interface required for the visual portion of the experiment. In addition, such an environment allowed for real-time variable modification within the program itself.

Upon initialization of the Windows program, a calibration subroutine was called which established the number of loops executed per millisecond by the processor. This parameter was subsequently used throughout the program to set the number of "do nothing" loops necessary for a prescribed delay of x milliseconds. For example, a call to the CalibrateTimer routine might return a loops-per-millisecond value of 12,303. A delay of 30ms would then correspond to:

$$number\ of\ delay\ loops = 12,303\ Loops - per - millisecond\ \times\ 30ms = 369090 \qquad (9)$$

This calibrated value was then multiplied by two values: the desired millisecond-delay between the appearance of each black box, and the desired millisecond delay between visual and audio stimuli.

When the "go" button was selected, one of three possible integer values (0, 1, or 2) was selected using a built-in pseudo-random number generator. The random number generator itself was initialized or "seeded" using a generic Windows "randomize" routine called during the callibration sequence described above. If a "0" was generated, an experimental trial without audio tone was selected. If a "1" was generated, an experimental trial in which both visual and audio stimuli occured in the same order was selected. Finally, if a "2" was randomly generated, an experimental trial in which visual and audio stimuli occurred in temporally-reversed order was selected. No matter which case was randomly selected, the initial occurrence of visual and audio stimuli (ie., left or right) was stored – along with the selection made by the test subject. Upon conclusion of each test (60 trials), a summary of all pertinent data was made available for analysis.

4.3.2 Verification of Test Set-Up: Due to the need for accurate temporal measurements within the multi-modal experiment, as well as the nature of the hardware used in its implementation, a test set-up verification procedure was conceived and carried out prior to actual experimentation. Specifically, the actual delay (in milliseconds) between onset of visual stimuli and auditory stimuli was measured and compared at regular intervals to the delay calculated within the software. The mean and standard deviation of the delay measured at each interval is provided in Table 3. (See appendix F for a complete description of the test setup verification procedure and the data obtained from this procedure). As can be seen in the table, the variation in recorded measurements across

Table 3. Mean and Standard Deviation of Measured Delay Intervals

	Calculated Delay							
Measurement	50	40	30	20	10	0	-10	-20
mean	39.2	44.7	30.0	31.7	13.3	-3	-17.2	-43.2
standard deviation	7.2	7.8	8.6	8.4	6.1	5.7	8.1	17.5

visual-auditory delay intervals is significant. Conceivably, this wide dispersal could render several of the planned test intervals redundant if the differences between perspective neighboring mean values are found to be statistically insignificant. To test this possibility, a pair-wise Test of Significance was performed among all delay intervals using a level of significance of 5%. The results obtained from this test, presented in Table 29 of Appendix F, found all neighboring interval means to be statistically significant – except the mean values for test intervals at 20 and 30 ms. As a result, test data collected at both intervals must be considered as one set of output.

As can be seen in Figure 38, the difference between the expected and measured delay increases as a function of the length of the delay itself. Such a trend can be attributed to the use of the Windows graphical interface. Windows is an event-driven environment; as such, the system

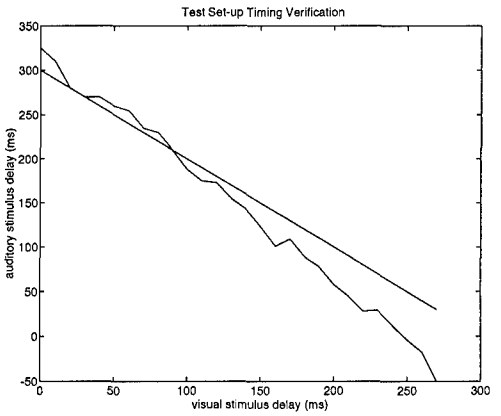


Figure 38. Test Setup Timing: Measured vs Expected Modal Delays (in ms)

periodically "borrows" control of the processor to poll for system messages. Since this polling is ongoing, increased use of the processor (in terms of time) leads to an increased "chunk" of processor time devoted to message checking.

4.4 Experiment I

4.4.1 Purpose: The purpose of Experiment I was to verify equal selection probability for "left" and "right" choices when presented with visual stimuli which occur simultaneously.

4.4.2 Method: The delay between left and right auditory stimuli was set equal to zero so that the test subject would hear only one auditory tone in each ear simultaneously. In addition, the delay between the onset of the 1st and 2nd visual stimuli (black boxes) was set equal to zero. Finally, the delay between visual and auditory stimuli was set equal to zero, which produced a delay of plus/minus 10ms between the onset of the multi-modal stimuli. A spatial-temporal representation of this setup is shown in Figure 39.

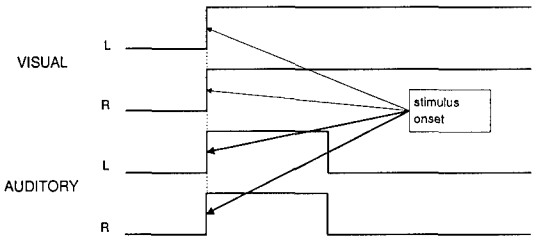


Figure 39. Experiment I: visual and auditory stimuli presented simultaneously; visual-auditory delay=0ms, visual-visual delay=0ms

4.4.3 Results: Psycho-sensory test data obtained from six test subjects are presented in Table 4. A pair-wise mean significance test using a confidence factor of 5% was applied to the test data to determine whether or not the observed mean values were statistically the same. The results from this test are presented in Table 5. Rejection of the "Null hypothesis" is interpreted to denote the existence of a statistical difference between the data pair being compared. (Note: The results shown have been adjusted to take into account the "Alpha Creep" observed to occur when comparisons among multiple data means are made(14)). As can be seen in the data, test subjects

Table 4. Experiment I: visual-visual delay = 0ms, visual-auditory delay = 0ms

	Percentage Correct(out of 30)	
Test Subject	*guessed right*	*guessed left*
1	50	50
2	53	46
3	43	56
4	36	63
5	33	66
6	46	53
mean	43.9	56.1
standard deviation	7.7	7.7

chose "left" slightly more often than "right." However, due to the wide variation in test data, the difference in means is deemed statistically insignificant.

Table 5. Pair-wise Test of Significance, visual-visual delay = 0ms, visual-auditory delay = 0ms

Configuration 1	*Configuration 2*	*T*	*Reject NULL Hypothesis?*
guessed right	guessed left	-1.9285	NO

4.5 Experiment II

4.5.1 Purpose: The purpose of Experiment II was to investigate and bound the temporal perception of visual stimuli both with and without the addition of auditory stimuli.

4.5.2 Method: The delay between visual and auditory stimuli was set equal to zero, which produced a delay of plus/minus 10ms between the onset of the multi-modal stimuli. The delay between the onset of the 1st and 2nd visual stimuli (black boxes) was then increased, in 10ms increments, from 0ms (ie., simultaneous) upward to 40ms. Spatial-temporal representations of each possible stimulus presentation is shown in Figures 40 through 42.

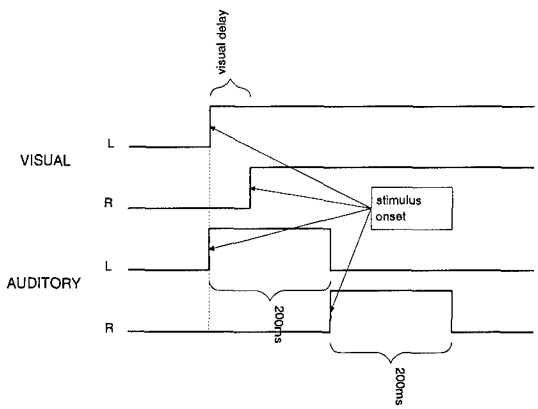


Figure 40. Experiment II: visual and auditory stimuli presented in order; visual-auditory delay=0ms, visual-visual delay=variable

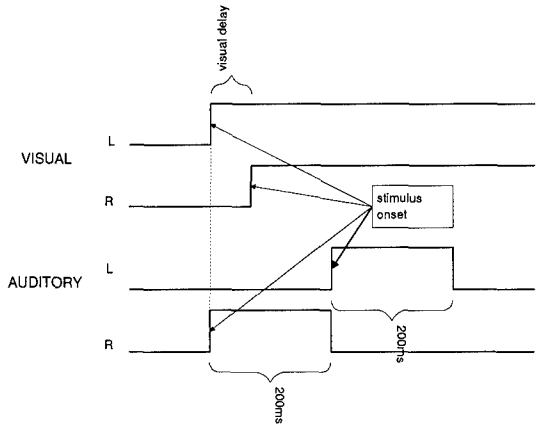


Figure 41. Experiment II: visual and auditory stimuli presented out of order; visual-auditory delay=0ms, visual-visual delay=variable

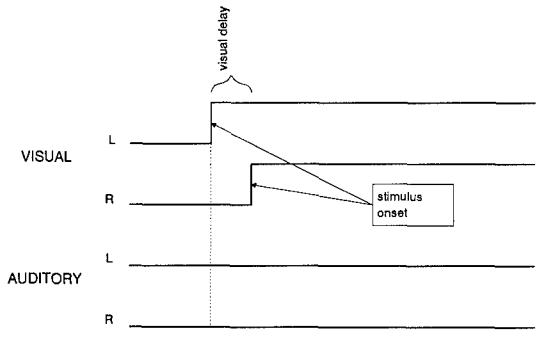


Figure 42. Experiment II: visual stimuli presented without accessory auditory stimuli; visual-visual delay=variable

4.5.3 Results: Psycho-sensory test data obtained from six test subjects at visual stimulus delays of 0, 10, 20, and 30 milliseconds, respectively, are presented (along with corresponding means and standard deviations) in Tables 6 through 9.

Table 6. Experiment II: visual delay=0ms

	Percentage Correct(out of 20)		
Test Subject	*silent*	*in-order*	*out-of-order*
1	55	70	10
2	40	70	25
3	50	90	15
4	45	85	5
5	55	80	30
6	50	50	35
mean	49.2	74.2	20.0
standard deviation	5.9	14.3	11.8

Table 7. Experiment II: visual delay=20ms

	Percentage Correct(out of 20)		
Test Subject	*silent*	*in-order*	*out-of-order*
1	70	85	30
2	95	100	20
3	75	95	20
4	45	65	30
5	75	95	30
6	55	85	25
mean	69.2	87.6	25.9
standard deviation	17.4	12.6	4.9

A pair-wise mean significance test using a confidence factor of 5% was applied to each set of data to determine the validity of separation observed in the data mean values. The results, presented in Tables 10 through 13, confirm the existence of cross-modal effects upon perception.

To illustrate the extent of these effects over time, the median correct percentages for all three

Table 8. Experiment II: visual delay=30ms

	Percentage Correct(out of 20)		
Test Subject	*silent*	*in-order*	*out-of-order*
1	75	95	55
2	95	100	50
3	50	90	25
4	50	75	40
4	50	75	40
5	95	90	50
6	100	95	85
mean	77.5	90.8	50.8
standard deviation	23.0	8.6	19.9

Table 9. Experiment II: visual delay=40ms

	Percentage Correct(out of 20)		
Test Subject	*silent*	*in-order*	*out-of-order*
1	85	100	85
2	80	90	65
3	80	95	85
4	95	100	50
5	90	100	60
6	100	95	65
mean	88.3	96.7	68.3
standard deviation	8.2	4.1	14.0

Table 10. Pair-wise Test of Significance, visual delay = 0ms

Configuration 1	*Configuration 2*	*T*	*Reject NULL Hypothesis?*
silent	in order	-3.9528	YES
silent	out of order	5.5760	YES
in order	out of order	5.6362	YES

Table 11. Pair-wise Test of Significance, visual delay = 20ms

Configuration 1	*Configuration 2*	*T*	*Reject NULL Hypothesis?*
silent	in order	-5.5	YES
silent	out of order	5.1387	YES
in order	out of order	9.4285	YES

Table 12. Pair-wise Test of Significance, visual delay = 30ms

Configuration 1	*Configuration 2*	*T*	*Reject NULL Hypothesis?*
silent	in order	-1.807	NO
silent	out of order	4.3386	YES
in order	out of order	5.3936	YES

Table 13. Pair-wise Test of Significance, visual delay = 40ms

Configuration 1	*Configuration 2*	*T*	*Reject NULL Hypothesis?*
silent	in order	-5.8797	YES
silent	out of order	1.8279	NO
in order	out of order	3.7251	YES

test scenarios were plotted at the 0, 10, 20, and 30 millisecond delay intervals, respectively. The resulting graph is presented in Figure 43. As shown in the graph, the impact of auditory stimuli

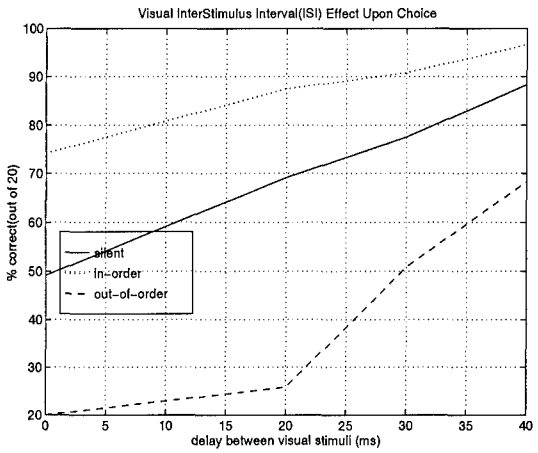


Figure 43. Experiment II: Effects of Inter Stimulus Interval(ISI) for visual stimuli applied in the presence of auditory stimuli

upon visual perception is diminished as the delay between the onset of visual stimuli is increased. The same effect is experienced when viewing a movie or television program whose audio signal is out-of-sync with the accompanying video signal; the descrepancy is often not apparent until the temporal gap between audio and visual presentation is extensive. This result appears to confirm previous studies on Intersensory Discrepancy: as the information from one sense becomes more discrepant from the information from the second sense, the influence of the former on the latter decreases (1). When viewed from the same perspective as Dr. Libet's experiments, the diminished effect can be attributed to temporal activity which "spills over" a given perceptual window of simultaneous activity.

The effects of accessory auditory stimuli have all been causal throughout Experiment II; the onset of auditory stimuli is never delayed to occur after the onset of the first visual stimuli.

To determine the effects of delayed auditory stimuli onset in relation to visual stimuli onset, the following experiment has been devised.

4.6 Experiment III

4.6.1 Purpose: The purpose of Experiment III was to investigate and bound the effects of delayed auditory stimuli upon perception of visual stimuli.

4.6.2 Method: The delay between the onset of the 1st and 2nd visual stimuli (black boxes) was set equal to 20ms. (This value was shown in Experiment II to produce the greatest variation in regards to correct choices made without auditory inputs and auditory inputs presented in reverse order from the visual stimuli). The delay between onset of the first visual stimulus and auditory tone was then increased, in 10ms increments, from 0ms (ie., simultaneous onset) upwards to 60ms. Spatial-temporal representation of each possible stimulus presentation is shown in Figures 44 through 46.

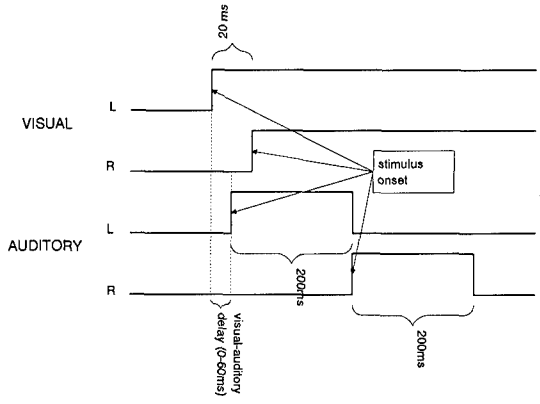


Figure 44. Experiment III: visual and auditory stimuli presented in order; visual-auditory delay=variable, visual-visual delay=20ms

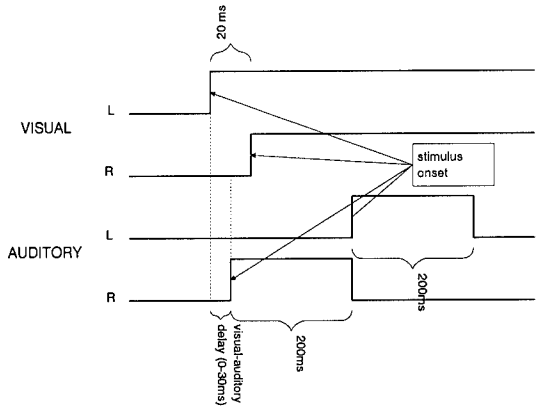


Figure 45. Experiment III: visual and auditory stimuli presented out of order; visual-auditory delay=variable, visual-visual delay=20ms

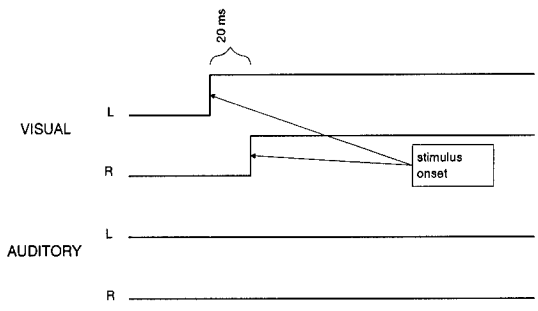


Figure 46. Experiment III: visual stimuli presented without accessory auditory stimuli; visual-visual delay=20ms

4.6.3 Results: Psycho-sensory test data obtained from six test subjects with visual-auditory stimulus onset delays of 0, 10, 20, 30, 40, 50, and 60 milliseconds, respectively, are presented (along with corresponding means and standard deviations) in Tables 14 through 20. A

Table 14. Experiment III: visual audio onset delay=0ms

	Percentage Correct(out of 20)		
Test Subject	*silent*	*in-order*	*out-of-order*
1	70	85	30
2	95	100	20
3	75	95	20
4	45	65	30
5	75	95	30
6	55	85	25
mean	69.2	87.6	25.9
standard deviation	17.4	12.6	4.9

Table 15. Experiment III: visual-audio onset delay=10ms

	Percentage Correct(out of 20)		
Test Subject	*silent*	*in-order*	*out-of-order*
1	90	100	30
2	75	100	10
3	80	55	35
4	55	85	60
5	70	80	40
6	60	75	55
mean	71.4	80.0	39.3
standard deviation	11.8	16.8	16.7

pair-wise mean significance test using a confidence factor of 5% was applied to each set of data to determine the validity of separation observed in the date mean values. The results for each delay interval can be seen in Tables 21 through 27.

Table 16. Experiment III: visual-audio onset delay=20ms

	Percentage Correct(out of 20)		
Test Subject	*silent*	*in-order*	*out-of-order*
1	60	75	60
2	65	90	55
3	75	90	30
4	75	75	65
5	50	55	35
6	80	100	60
7	80	95	40
mean	69.3	82.6	49.3
standard deviation	11.3	15.5	14.0

Table 17. Experiment III: visual-audio onset delay=30ms

	Percentage Correct(out of 20)		
Test Subject	*silent*	*in-order*	*out-of-order*
1	75	100	20
2	85	100	30
3	65	90	35
4	55	50	50
5	55	70	55
6	65	75	80
mean	67.5	81.7	45
standard deviation	10.8	18.1	21.4

Table 18. Experiment III: visual-audio onset delay=40ms

	Percentage Correct(out of 20)		
Test Subject	*silent*	*in-order*	*out-of-order*
1	70	95	50
2	65	70	50
3	50	70	40
4	80	100	40
5	85	100	0
6	95	85	55
mean	74.2	86.7	39.2
standard deviation	15.9	14.0	20.1

Table 19. Experiment III: visual-audio onset delay=50ms

	Percentage Correct(out of 20)		
Test Subject	*silent*	*in-order*	*out-of-order*
1	85	95	70
2	90	95	70
3	80	100	15
4	45	40	55
5	50	55	45
6	75	45	20
mean	70.8	71.7	45.8
standard deviation	18.8	27.9	24.0

Table 20. Experiment III: visual-audio onset delay=60ms

	Percentage Correct(out of 20)		
Test Subject	*silent*	*in-order*	*out-of-order*
1	80	80	70
2	75	90	75
3	70	95	35
4	70	65	50
5	45	40	65
6	65	50	40
mean	67.5	70.0	55.8
standard deviation	12.1	22.1	16.6

Table 21. Pairwise Test of Significance, visual-auditory delay = 0ms, visual delay = 20ms

Configuration 1	*Configuration 2*	*T*	*Reject NULL Hypothesis?*
silent	in order	-5.5	YES
silent	out of order	5.1387	YES
in order	out of order	9.4285	YES

Table 22. Pairwise Test of Significance, visual-auditory delay = 10ms, visual delay = 20ms

Configuration 1	*Configuration 2*	*T*	*Reject NULL Hypothesis?*
silent	in order	-1.2163	NO
silent	out of order	3.217	NO
in order	out of order	3.8	YES

Table 23. Pairwise Test of Significance, visual-auditory delay = 20ms, visual delay = 20ms

Configuration 1	*Configuration 2*	*T*	*Reject NULL Hypothesis?*
silent	in order	-4.2136	YES
silent	out of order	3.1909	NO
in order	out of order	4.5508	YES

Table 24. Pairwise Test of Significance, visual-auditory delay = 30ms, visual delay = 20ms

Configuration 1	*Configuration 2*	*T*	*Reject NULL Hypothesis?*
silent	in order	-3.1	NO
silent	out of order	1.9	NO
in order	out of order	2.48	NO

Table 25. Pairwise Test of Significance, visual-auditory delay = 40ms, visual delay = 20ms

Configuration 1	*Configuration 2*	*T*	*Reject NULL Hypothesis?*
silent	in order	-2.4	NO
silent	out of order	3.1	NO
in order	out of order	4.0	YES

Table 26. Pairwise Test of Significance, visual-auditory delay = 50ms, visual delay = 20ms

Configuration 1	*Configuration 2*	*T*	*Reject NULL Hypothesis?*
silent	in order	-0.1	NO
silent	out of order	2.1	NO
in order	out of order	1.9	NO

Table 27. Pairwise Test of Significance, visual-auditory delay = 60ms, visual delay = 20ms

Configuration 1	*Configuration 2*	*T*	*Reject NULL Hypothesis?*
silent	in order	-0.4	NO
silent	out of order	1.5	NO
in order	out of order	1.3	NO

As shown in the data, the means associated with each test presentation can be considered statistically significant until the delay between initial visual and auditory stimuli reaches 50 ms. At this point, the large variation in test data reduces the significance of the observed difference in means. With these limitations in mind, a graph illustrating the effects of temporal delay between visual and auditory stimuli onset is presented in Figure 47. The sharp drop observed at the 40 ms delay interval is due largely to one test subject's unusually high susceptability to the effects of out-of-order stimuli presentation. (refer back to Table 18)

As a result of the variability of inter-modal intevals observed during test setup verification, the measurements made throughout Experiment III can not be used to determine exact temporal boundaries for the relationship between visual and auditory stimuli. However, the data is useful in that it establishes a notable trend: accessory auditory stimuli have maximum impact upon visual choice if applied simultaneously (ie., inter-modal stimulus onset delay = 0). This effect is diminished but **still evident** as the delay between visual and auditory stimuli is increased. As

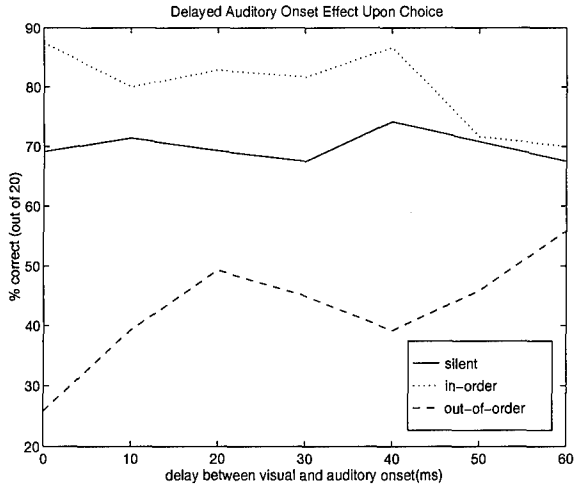


Figure 47. Experiment III: Effects of Inter-Modal Stimulus Interval(IMSI) (for visual-auditory stimuli onset) upon visual choice response

illustrated in Figure 47, the decrease in auditory influence is gradual. Specifically, even when the auditory stimulus is initiated 40 ms **after** the first visual stimulus, the auditory stimulus can alter the perception of the visual stimulus. This observed result suggests that perceived future inputs can impact current perception. Such data once again reinforces the idea that the perceptual "now" is not instantaneous, but encompasses a "window" in a neurological time frame. The temporal dimensions of this window establish the temporal resolution of the mind.

V. Conclusions

Throughtout this research effort, an emphasis has been placed upon the investigation of the neuronal-level propogation of visual stimulus information and the temporal effects associated with this propagation. During the course of this investigation, a non-causal link was shown to exist between Grossberg's Motion Oriented Contrast Filter and AFIT's Adaptive Retinal Model. This link was further developed through the simulation of Apparent Motion using a two-level Pulse Coupled Neural Network. (Simulation of the visual illusion was achieved by modifying the PCNN to implement a "global layer" analogous to the level 5 cells of a MOC filter). Finally, the investigation into temporal effects upon awareness was broadened to include a study of the influence of multi-modal stimulus inputs. Observations and conclusions drawn from this effort include the following:

1. The presence of a neuronal framework whereby lower-layer neurons feed into higher layers with progressively larger feeding and/or linking footprints leads to the propagation of temporal delays in the realization of outputs at the higher levels. In addition, the presence of internal cell activity (at each cell) governed by activity levels akin to a leaky integrator(ie., existence of a finite build-up and decay time constant) provides each cell with its own rudimentary short term "memory." When this cell activity memory is coupled with "global" neuronal connections, the result is a filtered representation of the external world. From this perspective, the apparent motion phenomenon might be considered a "quirk" in the operation of the filter.

2. The Pulse Coupled Neural Network can be configured to demonstrate a "pulsed" representation of the apparent motion phenomenon. This presentation provides a unique perspective for possible interpretation of the visual illusion.

3. Perceptual and physical time are not necessarily the same. In fact, the global interactions described above are governed by the time constants of activity levels internal to each neuron at every level. As a result, the precise temporal information inherent in the arrival of a stimulus input is lost during the propagation of that information through the brain. The brain's solution to this problem appears to be the use of a perceptual window – events (stimuli) occurring close enough in physical time fall within the perceptual tolerance of the perceptual window. The resulting perception is simultaneous occurrence. Knowledge of the existence of a perceptual window is important from a Human Factors standpoint in regards to multi-sensory integration. Specifically, any human-machine interface implementing multi-sensory stimulation must be designed so as to minimize the impact of one sense modality upon another. Failure to do so may result in an unintended perceptual awareness of incorrect stimuli.

4. The existence of the perceptual window described above gives rise to perhaps the most significant (and most exciting) finding: noncausality in multi-modal perception. Experiment III of Chapter 4 demonstrated the ability of one modal stimulus to affect the **future** perception of another modal stimulus. Such an ability may allow the brain to maintain a seamless, consistent, and stable internal "world model" of its external environment. The "price" for this construction of such a model: the occurrence of stimulus input illusions... *al la* apparent motion.

Appendix A. Grossberg's Static Boundary Contour System

A.1 Background

Back in 1984 Grossberg et al. introduced the Static Boundary Contour System (BCS) model. The model was developed to simulate the visual system's ability to detect, complete, and regularize boundary segmentations in response to a variety of retinal images(7) The model consists of two major subdivisions: a non-linear oriented-sensitive filter (OC filter), and a cooperative-competitive feedback network, called the CC loop(see figure 48. A brief description of each division follows.

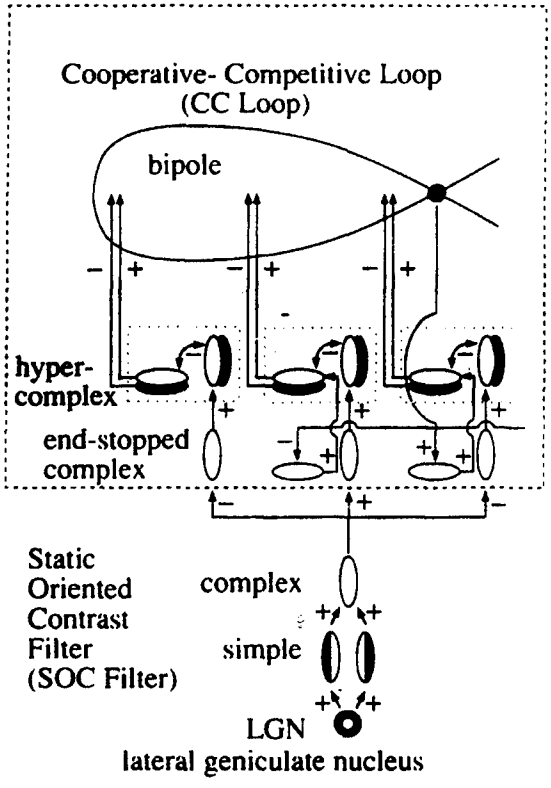


Figure 48. Static Boundary Contrast System

A.2 Oriented Contrast Filter

The first stage of the Oriented Contrast Filter models the shunting on-center off-surround interactions at the retinal and LGN levels.(figure 49) As such, the model is able to account for variable illumination and, at the same time, amplify local contrast in an input image. The outputs

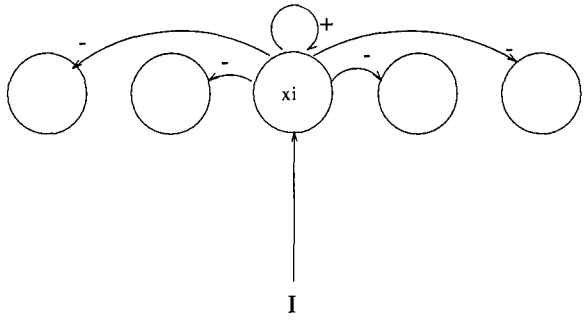


Figure 49. Shunting On-Center Off-Surround Interactions

of these model LGN cells are then input to pairs of like-oriented simple cells that are sensitive to direction of contrast (stage 2).(7) This step models the "orientational hypercolumns" discussed by Hubert and Weisel(3). For this model, the columns consist of oriented receptive fields such as the ones shown in figure 50. These fields, here-to-for refered to as simple cells, exhibit an elongated shape in order to increase orientational sensitivity. They are divided into positive and negative activation areas along their major axes of symmetry(3). Each simple cell corresponds to individual points in a scene. As such, they are used to "store" the preferred orientation of each of these points. The simple cells described above send their rectified ouputs to like-oriented complex cells. Since the complex cells receive inputs from simple cells with opposite direction of contrast sensitivity, they are no longer sensitive to direction of contrast. Outputs from complex cells are gated by habituative chemicals and sent to a corresponding hypercomplex cell network (referred to by Grossberg as the first competitive stage of the CC loop). The habituation process is necessary in order to reset

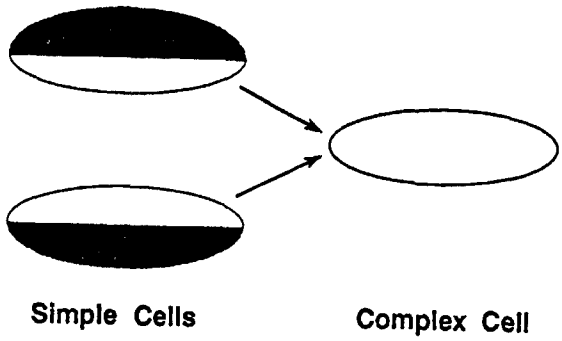


Figure 50. Oriented Receptive Fields - simple/complex cells

boundary signals and thereby prevent a "smearing" effect which would otherwise result during the rapid presentation of input images in time.

A.3 Competition Cooperation Loop

First Competitive Stage: Once again, the interactions between complex and hypercomplex cells can be modeled as an on-center off-surround network whereby the "on-center" cells are activated while the "off-surround" cells are inhibited. As a result of this operation, hypercomplex cells of like position and orientation are turned "on" (excited), while at the same time turning "off" hypercomplex cells of the same orientation but at different locations. One way to visualize this process is to think of the cell structure as a volume in position (x,y) and orientation space in which all complex cells of a like orientation combine to form a layer in the complex cell volume (3). (See figure 51). Imposing a center-surround receptive field on each complex cell at each x,y position in space excites the complex cell at this position while inhibiting like oriented complex cells at surrounding positions within a given orientation layer.(3) The effect is twofold: the edges detected by the complex cells are strengthened, and perpendicular boundaries are formed at the ends of these like-oriented response edges. Grossberg describes this process as end cutting.

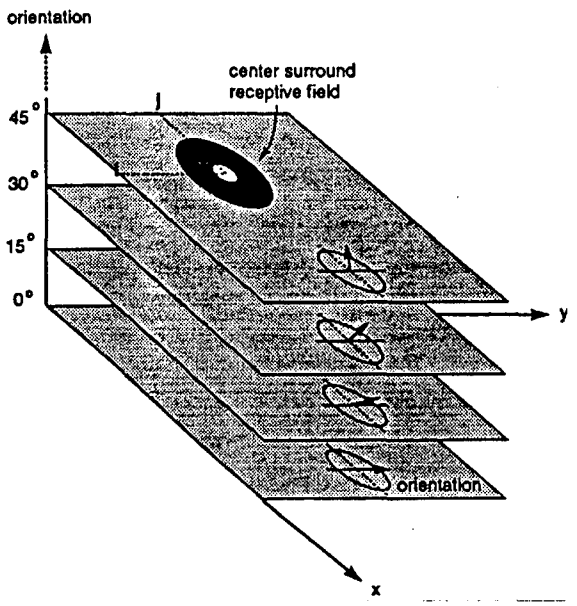


Figure 51. Competitions Among Like Orientations at Different Positions

Second Competitive Stage: The outputs from the first layer of hypercomplex cells serve as inputs for a second round of competions in a higher-order on-center off-surround network. At this level, different orientations compete to be the dominant input at hypercomplex cells corresponding to spatial position. Recall that the first competitive stage excites a range of orientations (layers) at specific locations(figure 50) The second stage uses these orienations as inputs and picks the "best" orientation for that particular spatial location. Once again, the result of such a competition is to sharpen existing boundary signals. **Cooperation Stage:** Outputs from the second competitive stage are input into cooperative bipole cells that initiate long-range boundary grouping and completions.(7) The bipole cells construct boundaries by measuring the location and density of surrounding hypercomplex cell inputs within specific orientation layers. If the receptive fields of a bipole are "sufficiently" activated, then, the bipole cell fires. The activation of individual bipole cells feeds back to hypercomplex cells of the same orientation and position and inhibits nearby

cells. As a result, spatially and orientationally consistent boundaries are enhanced, and inconsistent boundaries are inhibited. An example of this competition-cooperation loop is presented in figure 52(3)(10). The two active-on cells (of the same orientation) in the lower half of the figure activate the receptive fields of the bipole filter(top horizontal line). This process is represented by path 1. Having been sufficiently activated, the bipole cell fires (path 2) thereby exciting hypercomplex cells within that orientation layer. The new "cell excitation" configuration is fed back through the competitive stages. If these hypercomplex excitations survive, a second bipole cell may fire (path 3) Once again, the dipole cell will excite corresponding hypercomplex cells within the same orientation layer (path 4). This process of bottom-up and top-down interaction between hypercomplex cells and bipole cells is repeated until the output converges to a final boundary segmentation.(7).

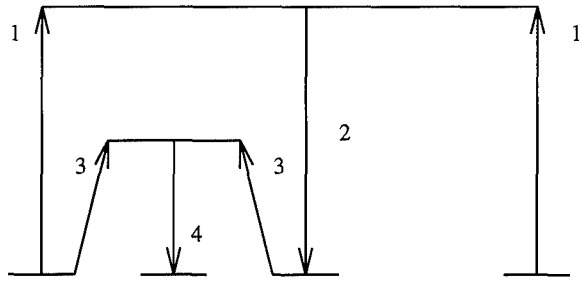


Figure 52. Competition-Cooperation

A.4 BCS and Apparent Motion:

As was stated above, the simple cells of a BCS are sensitive to direction-of-contrast. However, this sensitivity is removed when the outputs from simple cell pairs with opposite direction-of-contrasts are input into complex cells. The resulting network can therefore extract boundary structures independent of illumination fluctuations (ie., shadows) Unfortunately this same network is made insensitive by these properties to direction-of-motion. By itself, then, the existing BCS can

not account for real or apparent motion processing. What is needed, then, is a system which can join the existing network to an additional "stage" of processing which is sensitive to direction-of-motion. Grossberg's Motion BCS performs this task.

Appendix B. Sample MOC Filter Output

For the purposes of the simulations which follow, the front-end of the MOC filter was simplied by assuming that the transient cell responses always took on a value of 1(12). (This same assumption was used by Grossberg so as to simplify demonstration of the filter's output). In the first section which follows, the spatial separation between stimuli was varied from 40 pixels to 550 pixels while the temporal separation was held constant at 10 ms. In the second section, the spatial separation was held constant at 240 pixels while the temporal separation was increased from 40 ms to 440 ms.

B.1 Spatial Separation

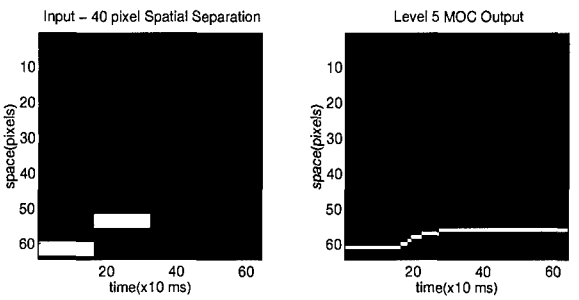


Figure 53. Spatial Separation = 40 pixels

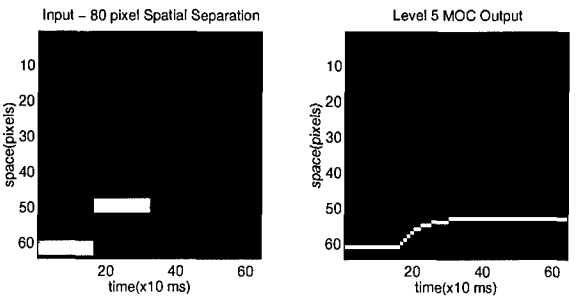


Figure 54. Spatial Separation = 80 pixels

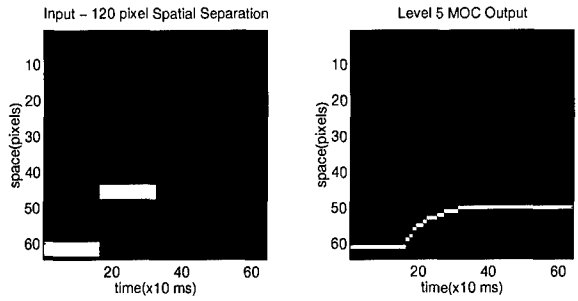


Figure 55. Spatial Separation = 120 pixels

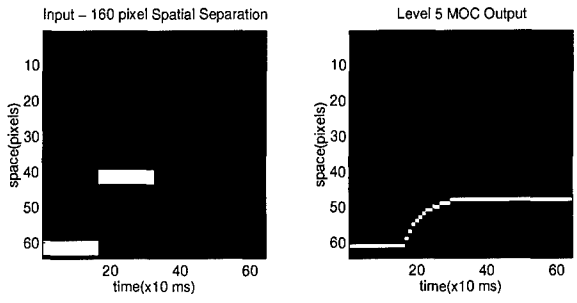


Figure 56. Spatial Separation = 160 pixels

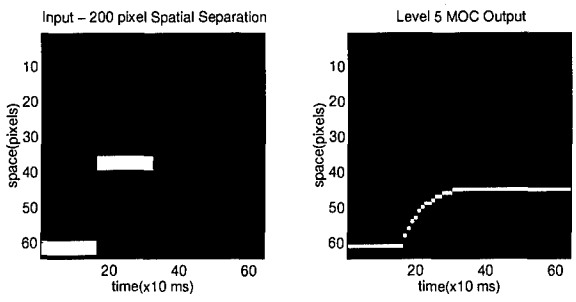


Figure 57. Spatial Separation = 200 pixels

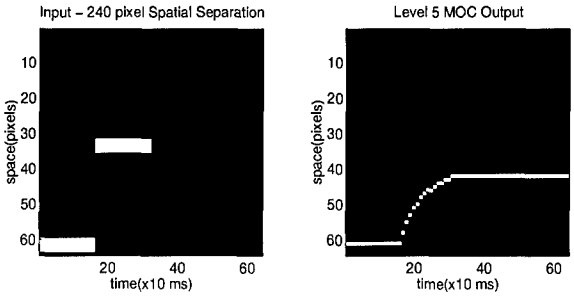


Figure 58. Spatial Separation = 240 pixels

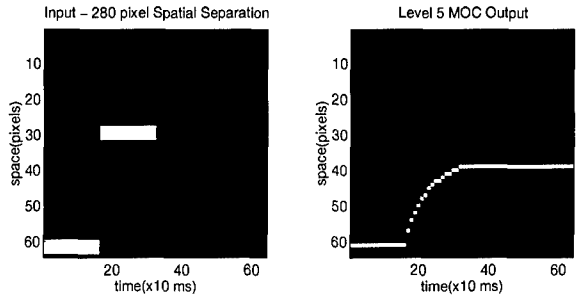


Figure 59. Spatial Separation = 280 pixels

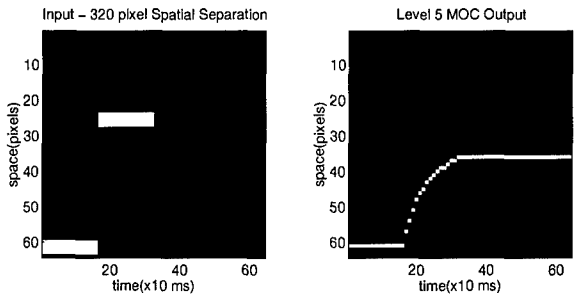


Figure 60. Spatial Separation = 320 pixels

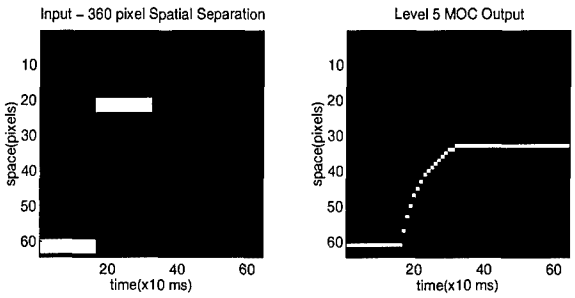


Figure 61. Spatial Separation = 360 pixels

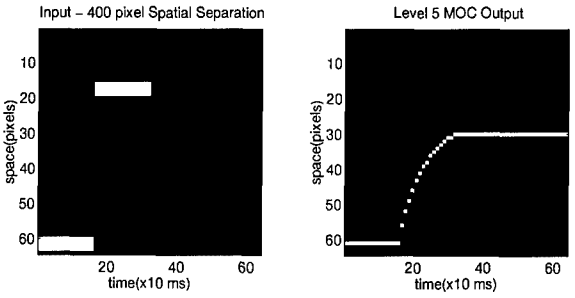


Figure 62. Spatial Separation = 400 pixels

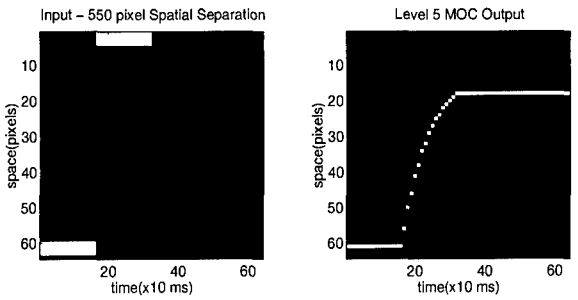


Figure 63. Spatial Separation = 550 pixels

B.2 Temporal Separation

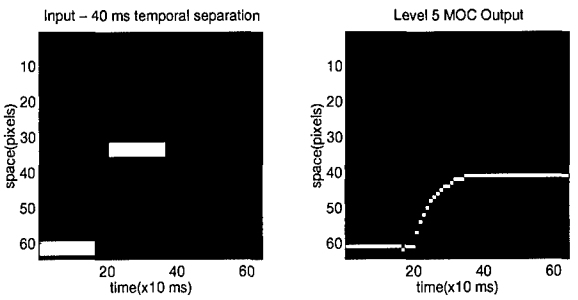


Figure 64. Temporal Separation = 40 ms

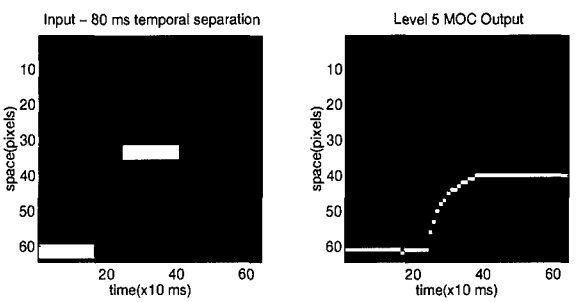


Figure 65. Temporal Separation = 80 ms

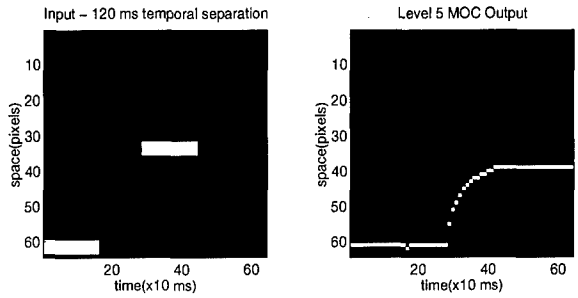


Figure 66. Temporal Separation = 120 ms

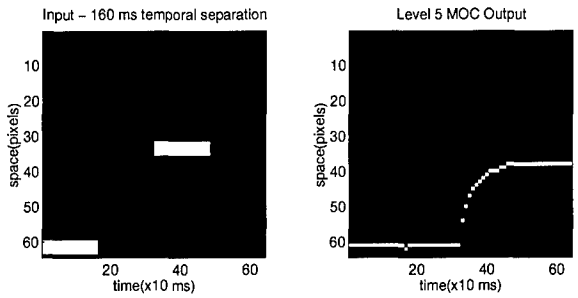


Figure 67. Temporal Separation = 160 ms

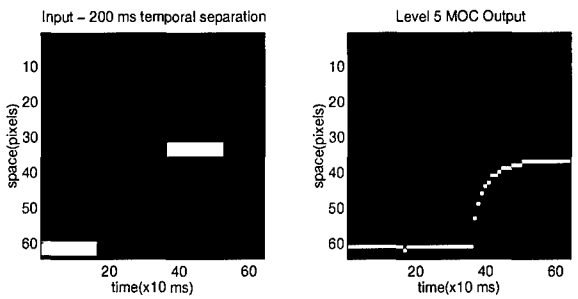


Figure 68. Temporal Separation = 200 ms

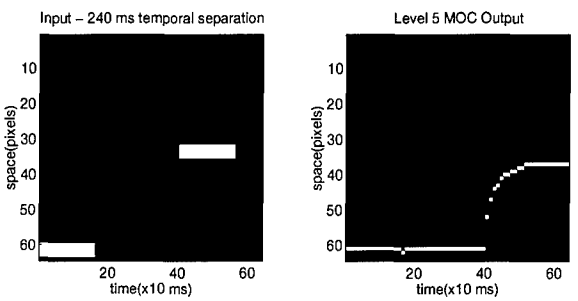


Figure 69. Temporal Separation = 240 ms

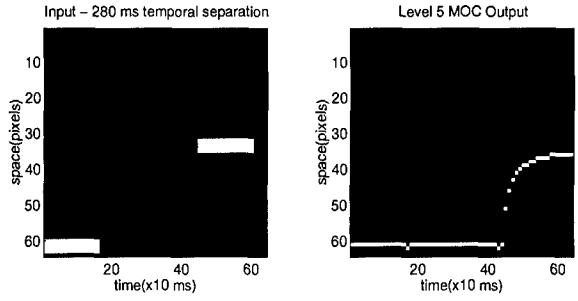


Figure 70. Temporal Separation = 280 ms

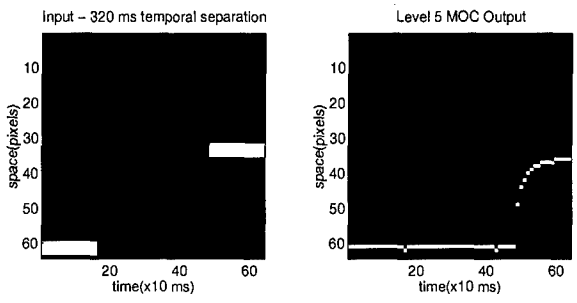


Figure 71. Temporal Separation = 320 ms

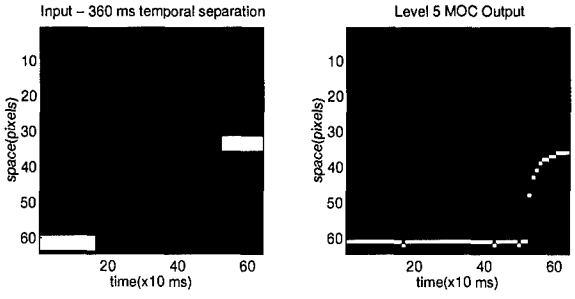


Figure 72. Temporal Separation = 360 ms

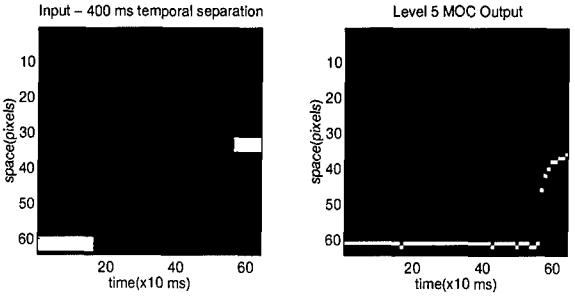


Figure 73. Temporal Separation = 400 ms

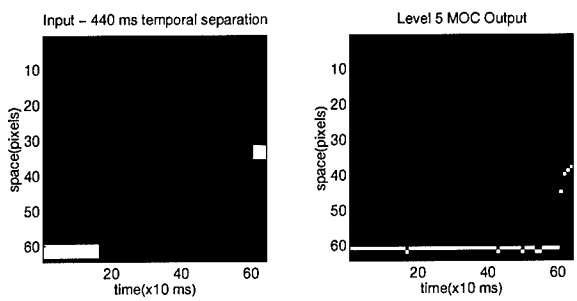


Figure 74. Temporal Separation = 440 ms

Appendix C. Motion Oriented Contrast(MOC) Filter Computer Code

```
/***********************************************
* *
*  Grossberg's Motion Oriented Contrast Filter  *
* (MOC) *
* *
* This routing implements Grossberg's     *
* Motion Oriented Contrast Filter (MOC)   *
* Key parameters are identified in  *
* appropriate subroutines *
* Designer:  Don Hill, GE 95D *
*    Thesis Research *
* *
***********************************************/


#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "macros_bcs.h"


#define SQR(a)       (a)*(a)


float Shunt();
```

```
float Orient();

float Transient();

float Transmit();

void free_matrix();

void output();


float    **matrix();

main(argc,argv)


char *argv[];

int argc;

{


int  Row, Col, i, j, k, l, loc1, loc2;

float   H, J, F, G, N, P, Q;

float   x1_in_DB,x1_in_BD;

float   result,result1,total;

float   **image_array, **bcs_array, **x1_array;

float   **x2BD_array, **x2DB_array;

float   **y_array,**dy_array;

float   **w_on_array, **w_off_array;

float   **z_on_array, **z_off_array;

float   **x3_on_array, **x3_off_array;
```

```
float   **BD_on_array, **BD_off_array;

float   **DB_on_array, **DB_off_array;

float   **local_r_array, **local_l_array;

float   **kernel, **R5_array, **L5_array;

float   **R_5_out, **L_5_out;

float   ***out_array, *in_array;

FILE *image_ptr, *file_ptr, *FID;

char in_string[80];


/* read in input file */


if(argc != 4)

{

  printf("!!! The command line should be: !!!\n\n   test [in_file][row_length][col_length] (

 exit(0);

}


if((image_ptr = fopen(argv[1], "r")) == NULL)

  {

    printf("I can't find the file. \n");

    exit(1);

  }
```

```
Row = atoi(argv[2]);

Col = atoi(argv[3]);




/********* Allocate memory for arrays *********/


image_array = matrix(0, Row-1, 0, Col-1);

bcs_array = matrix(0, Row-1, 0, Col-1);

x1_array = matrix(0, Row-1, 0, Col-1);

x2BD_array = matrix(0, Row-1, 0, Col-1);

x2DB_array = matrix(0, Row-1, 0, Col-1);

y_array = matrix(0, Row-1, 0, Col-1);

dy_array = matrix(0, Row-1, 0, Col-1);

w_on_array = matrix(0, Row-1, 0, Col-1);

w_off_array = matrix(0, Row-1, 0, Col-1);

z_on_array = matrix(0, Row-1, 0, Col-1);

z_off_array = matrix(0, Row-1, 0, Col-1);

x3_on_array = matrix(0, Row-1, 0, Col-1);

x3_off_array = matrix(0, Row-1, 0, Col-1);

BD_on_array = matrix(0, Row-1, 0, Col-1);

BD_off_array = matrix(0, Row-1, 0, Col-1);

DB_on_array = matrix(0, Row-1, 0, Col-1);
```

```
DB_off_array = matrix(0, Row-1, 0, Col-1);

local_r_array = matrix(0, Row-1, 0, Col-1);

local_l_array = matrix(0, Row-1, 0, Col-1);

kernel = matrix(0, Row-1, 0, Col-1);

R5_array = matrix(0, Row-1, 0, Col-1);

L5_array = matrix(0, Row-1, 0, Col-1);

R_5_out = matrix(0, Row-1, 0, Col-1);

L_5_out = matrix(0, Row-1, 0, Col-1);

/********* Read in image array ***************/


loopij(Row,Col)

  fscanf(image_ptr, "%f ", &image_array[i][j]);


/********* Read in Static BCS inputs **********/


FID=fopen("/tmp_mnt/home/hawkeye7/95d/dehill/BCS/MOC_code/simbcs.dat","rt");

loopij(Row,Col)

  fscanf(FID, "%f ", &bcs_array[i][j]);


/********* Preprocess input (level 1) *********/


printf("Preprocessing data (shunting response) \n");
```

```
for (i=0;i<Row;i++)
  {

  result = 0.0;
  for (j=0;j<Col;j++)
    {
    x1_array[i][j] = result;
    result=Shunt(x1_array[i][j],image_array[i][j]);
    x1_array[i][j]=result;
    }
  }

printf("\n \n ******Level 1 Finished******* \n \n");

/********* Create Oriented Sustained Cells ******/
/*********          (Level 2)    ******/

/*note:  two types of oriented sustained cells must
be computed:  bright-dark and dark-bright cells*/
```

```
printf("creating Oriented sustained cells (level 2) \n");


for (i=0;i<Row;i++)
  {
  result = 0.0;
  result1 = 0.0;


  for (j=0;j<Col;j++)
   {
    if (i==0) /* special case -- edge of input */
      x1_in_DB = x1_array[i][j];
    else
    if(x1_array[i][j] - x1_array[(i-1)][j] > 0)
      x1_in_DB = x1_array[i][j] - x1_array[(i-1)][j];
    else
      x1_in_DB = 0.0;


    if (i==(Row-1)) /* special case -- edge of input */
      x1_in_BD = x1_array[i][j];
    else
    if(x1_array[i][j] - x1_array[(i+1)][j] > 0)
      x1_in_BD = x1_array[i][j] - x1_array[(i+1)][j];
    else
```

```
      x1_in_BD = 0.0;

    x2BD_array[i][j] = result;
    x2DB_array[i][j] = result1;
    result= Orient(x2BD_array[i][j],x1_in_BD,0);
    result1 = Orient(x2DB_array[i][j],x1_in_DB,0);
    x2BD_array[i][j]=result;
    x2DB_array[i][j]=result1;
    }
   }

printf("\n \n ******Level 2 Finished******* \n \n");

/********* Create Level 3 Transient Cells ******/

printf("\n \n calculating y array \n \n");

for (i=0;i<Row;i++)
  {
  result=0;
  for (j=0;j<Col;j++)
    {
    y_array[i][j] = result;
```

```
    result = Transient(y_array[i][j],x1_array[i][j],0);
    y_array[i][j] = result;

/*printf("x1: %f   y: %f \n",x1_array[i][j],y_array[i][j]);*/
    }
  }

printf("\n \n ****** y_array calculated ******** \n \n");

printf("\n \n ****** calculating dy/dt ******* \n \n");

F = 0.4;
G = 2.0;

for (i=0;i<Row;i++)
  {
  for (j=0;j<Col;j++)
    {
    /*dy_array[i][j] = -F*y_array[i][j] + (G-y_array[i][j]) * (0 + x1_array[i][j]);*/

    if (j==0)
      {
      dy_array[i][j] = ((y_array[i][j+1]-y_array[i][j])/.1)/2;
```

```
      }

    else

    if (j==(Col-1))
      {
      dy_array[i][j] = ((y_array[i][j]-y_array[i][j-1])/.1);
      }

    else
      {
      dy_array[i][j] = (((y_array[i][j+1]-y_array[i][j])/.1)+((y_array[i][j]-y_array[i][j-1]
      }
    }
  }

  printf("\n \n ****** generating transient half wave rectifications ****** \n \n");

H = 0.5;
J = -0.001;

for (i=0;i<Row;i++)
  {
```

```
  for (j=0;j<Col;j++)
    {
    w_on_array[i][j] = dy_array[i][j] - H;
    if (w_on_array[i][j] < 0)
w_on_array[i][j] = 0;

    w_off_array[i][j] = J - dy_array[i][j];
    if (w_off_array[i][j] < 0)
w_off_array[i][j] = 0;
    }
  }

printf("\n \n ****** w arrays calculated ****** \n \n");

printf("\n \n ****** calculating transmitter gate array, z ****** \n \n");

for (i=0;i<Row;i++)
  {

  result = 0.0;
  result1 = 0.0;

  for (j=0;j<Col;j++)
```

```
    {
    z_on_array[i][j] = result;
    z_off_array[i][j] = result1;
    result=Transmit(z_on_array[i][j],w_on_array[i][j]);
    result1=Transmit(z_off_array[i][j],w_off_array[i][j]);
    z_on_array[i][j] = result;
    z_off_array[i][j] = result1;
    }
  }

printf("\n \n ****** z_arrays calculated ******** \n \n");

printf("\n \n ****** calculating level 3 transient response ****** \n \n");

for (i=0;i<Row;i++)
  {
  for (j=0;j<Col;j++)
    {
    x3_on_array[i][j] = w_on_array[i][j]*z_on_array[i][j];
    x3_off_array[i][j] = w_off_array[i][j]*z_off_array[i][j];
    }
  }
```

```
printf("\n \n ****** level 3 transient response cells calculated ****** \n \n");

printf("\n \n ******Level 3 Finished******* \n \n");

/****** Level 4 Processing ******/

printf("\n \n ****** calculating sustained transient cells ****** \n \n");

for (i=0;i<Row;i++)
  {
  for (j=0;j<Col;j++)
    {
    BD_on_array[i][j] = x2BD_array[i][j] * x3_on_array[i][j];
    BD_off_array[i][j] = x2BD_array[i][j] * x3_off_array[i][j];
    DB_on_array[i][j] = x2DB_array[i][j] * x3_on_array[i][j];
    DB_off_array[i][j] = x2DB_array[i][j] * x3_off_array[i][j];
    }
  }
printf("\n \n ******Level 4 Finished******* \n \n");

/****** Level 5 Processing ******/
```

```
printf("\n \n ***** calculating local and global motion cells ***** \n \n");


N = 5.0;

P = 1.0;


 for (i=0;i<Row;i++)

  {

  for (j=0;j<Col;j++)

    {

    /*local_r_array[i][j] = N * BD_on_array[i][j] + P * DB_off_array[i][j];*/

    local_l_array[i][j] = P * BD_off_array[i][j] + N * DB_on_array[i][j];


/*  level 5 input set equal to level 1 sustained cells if transient response

    assumed to be 1: */


    local_r_array[i][j] = x1_array[i][j];

    }

  }


printf("\n \n caculating Gaussian kernal \n \n ");
```

```
 Q = 60.0; /* sets breadth of kernel */


for (i=0;i<Row;i++)
  {
  for (k=0;k<Row;k++)
    {
    kernel[i][k]=exp(-SQR(k-i)/(2 * SQR(Q)));
    }
  }
for (i=0; i<Row;i++)
  {
  for (j=0;j<Col;j++)
    {
    R5_array[i][j] = 0;
    L5_array[i][j] = 0;
    for (k=0;k<Row;k++)
      {
        R5_array[i][j] = R5_array[i][j] + kernel[i][k]*local_r_array[k][j];
        L5_array[i][j] = L5_array[i][j] + kernel[i][k]*local_l_array[k][j];
      }
    }
  }
```

```
for (i=0;i<Row;i++)
  for (j=0;j<Col;j++)
    {
    R_5_out[i][j]=0;
    L_5_out[i][j]=0;
    }

for (j=0;j<Col;j++)
  {
  result = R5_array[0][j];
  loc1 = 0;
  loc2 = j;
  for (i=1;i<Row;i++)
    {
    if (R5_array[i][j] > result)
      {
      result = R5_array[i][j];
      loc1 = i;
      loc2 = j;
      }
    }
  R_5_out[loc1][loc2] = R5_array[loc1][loc2];
  }
```

```
for (j=0;j<Col;j++)
  {
  result = L5_array[0][j];
  loc1 = 0;
  loc2 = j;
  for (i=1;i<Row;i++)
    {
    if (L5_array[i][j] > result)
      {
      result = L5_array[i][j];
      loc1 = i;
      loc2 = j;
      }
    }
  L_5_out[loc1][loc2] = L5_array[loc1][loc2];
  }


output(R5_array,Row,Col);


/******  Free up allocated memory  *******/
```

```
free_matrix(image_array,0,Row-1,0,Col-1);

free_matrix(bcs_array,0,Row-1,0,Col-1);

free_matrix(x1_array,0,Row-1,0,Col-1);

free_matrix(x2BD_array,0,Row-1,0,Col-1);

free_matrix(x2DB_array,0,Row-1,0,Col-1);

free_matrix(y_array,0,Row-1,0,Col-1);

free_matrix(dy_array,0,Row-1,0,Col-1);

free_matrix(w_on_array,0,Row-1,0,Col-1);

free_matrix(w_off_array,0,Row-1,0,Col-1);

free_matrix(z_on_array,0,Row-1,0,Col-1);

free_matrix(z_off_array,0,Row-1,0,Col-1);

free_matrix(x3_on_array,0,Row-1,0,Col-1);

free_matrix(x3_off_array,0,Row-1,0,Col-1);

free_matrix(BD_on_array,0,Row-1,0,Col-1);

free_matrix(BD_off_array,0,Row-1,0,Col-1);

free_matrix(DB_on_array,0,Row-1,0,Col-1);

free_matrix(DB_off_array,0,Row-1,0,Col-1);

free_matrix(local_r_array,0,Row-1,0,Col-1);

free_matrix(local_l_array,0,Row-1,0,Col-1);

free_matrix(kernel,0,Row-1,0,Col-1);

free_matrix(R5_array,0,Row-1,0,Col-1);

free_matrix(L5_array,0,Row-1,0,Col-1);

free_matrix(R_5_out,0,Row-1,0,Col-1);
```

```
free_matrix(L_5_out,0,Row-1,0,Col-1);

}

/********* End of Main Program **********/



/********* Begin Subroutines *************/



/********* Level 1 Shunting Response *****


t =  initial time, t0

h =  range of differentiation

n =  number of iterations

x =  integration variable

input = image input

A = rate of passive decay

B = maximum activity of cell


This subroutine performs an integration to

construct the shunting response of input

image
```

```
*****************************************/

float Shunt(x,input)

float x;
float  input;

{
float t=0;
float h=.1;
float n=100;
float FNF;
float A=.5;
float B=10;

h=(h-t)/n;

while(n>0)
{
FNF = -.A*x + (B-x)*input;
x=x+h*FNF;
t=t+h;
n=n-1;
```

```
}

return(x);
}

/******** Level 2 Oriented Sustained Cells ****

two matrices -- x2BD, x2DB use Orient to produce
oriented sustained cells

t =  initial time, t0
h =  range of differentiation
n =  number of iterations
x =  integration variable
input = image input
bcs = bcs input
C = rate of passive decay
D = maximum activity of cell
E = oriented input scale value

This subroutine performs an integration to
```

```
construct the oriented sustained response for

bright-dark and dark-bright cells

**********************************************/

float Orient(x,input,bcs)

float x;

float input;

float bcs;

{

float C=0.1;

float D=10.0;

float E=10.0;

float t=0;

float h=.1;

float n=100;

float FNF;

h=(h-t)/n;
```

```
while(n>0)
{
FNF = -C*x + (D-x) * E * (bcs + input);
x=x+h*FNF;
t=t+h;
n=n-1;
}

return(x);
}

/********* Level 3 Transient Cells *****

t =  initial time, t0
h =  range of differentiation
n =   number of iterations
y = integration variable
input = level 1 sustained response input
bcs = bcs input
F = rate of passive decay
G = maximum activity of cell

This subroutine performs an integration to
```

```
calculate transient-on and transient-off

response activity


*****************************************/


float Transient(y,input,bcs)


float y;

float input;

float bcs;


{


float F=0.4;

float G=2.0;


float t=0;

float h=.1;

float n=100;

float FNF;


h=(h-t)/n;
```

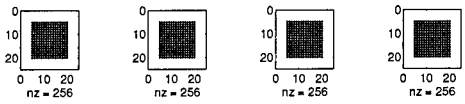
```
while(n>0)
{
FNF = -F*y + (G-y)*(bcs + input);
y=y+h*FNF;
t=t+h;
n=n-1;
}

return(y);
}

/********* Level 3 Transmitter Gates *****

t =  initial time, t0
h =  range of differentiation
n =   number of iterations
z = integration variable
input = rectified transient response input
K = accumulation rate
L = maximum activity of cell
M = habituation scale parameter

This subroutine performs a habituation of
```

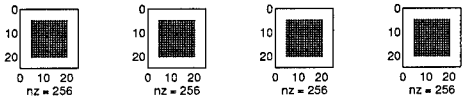
```
the rectified transient response inputs


*****************************************/

float Transmit(z,input)


float z;

float input;


{


float K=0.06;

float L=3.0;

float M=5.0;


float t=0;

float h=.1;

float n=100;

float FNF;


h=(h-t)/n;


while(n>0)

{
```

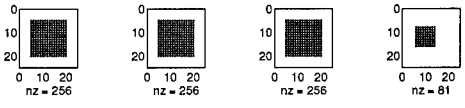
```
FNF = K*(L-z) - M*input*z;

z=z+h*FNF;

t=t+h;

n=n-1;

}


return(z);

}




/*******  Function output  *******/


void output(in_array,Row,Col)

float **in_array;

int Row;

int Col;

{

  int i,j,k;

  FILE *file_ptr;


  printf("\n\n");

  printf("*** Making the output file ***");
```

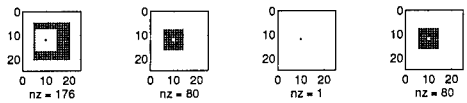
```
  printf("\n\n");



  file_ptr = fopen("R5arb.dat", "w");


  for (i=0;i<Row;i++)
    {
    for (j=0;j<Col;j++)
      {
      fprintf(file_ptr, "%f ", in_array[i][j]);
      }
    fprintf(file_ptr, "\n");
    }


  printf("      *******  DONE WITH BCS *******");
  printf("\n\n");
}
```

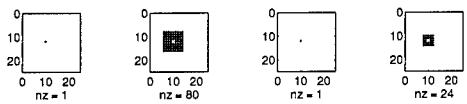
Appendix D. PCNN Level 2 Apparent Motion Pulse Output

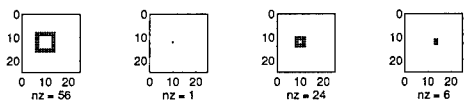


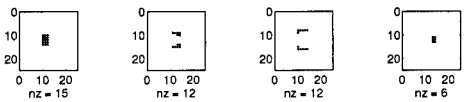


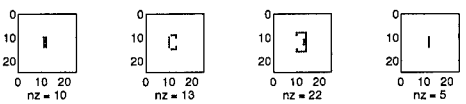


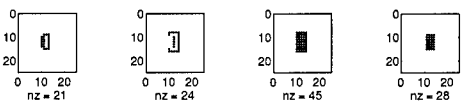

0
10
20
0 10 20
nz = 176
0
10
20
0 10 20
nz = 80
0
10
20
0 10 20
nz = 1
0
10
20
0 10 20
nz = 80

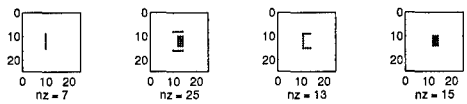

0
10
20
0 10 20
nz = 1
0
10
20
0 10 20
nz = 80
0
10
20
0 10 20
nz = 1
0
10
20
0 10 20
nz = 24

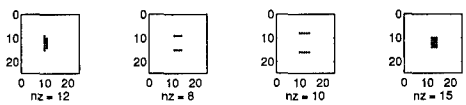

0
10
20
0 10 20
nz = 56
0
10
20
0 10 20
nz = 1
0
10
20
0 10 20
nz = 24
0
10
20
0 10 20
nz = 6

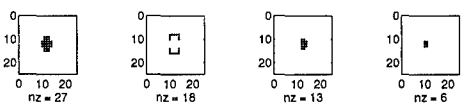

0
10
20
0 10 20
nz = 15
nz = 12
nz = 12
nz = 6

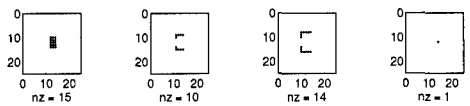

0
10
20
0 10 20
nz = 10
nz = 13
nz = 22
nz = 5

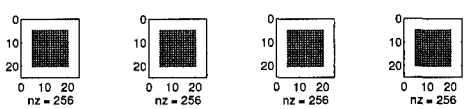

0
10
20
0 10 20
nz = 21
nz = 24
nz = 45
nz = 28

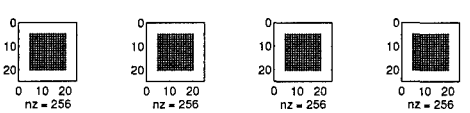

0
10
20
0 10 20
nz = 7
nz = 25
nz = 13
nz = 15

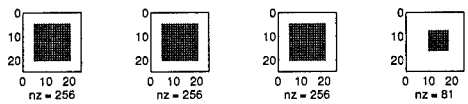

0
10
20
0 10 20
nz = 12
nz = 8
nz = 10
nz = 15

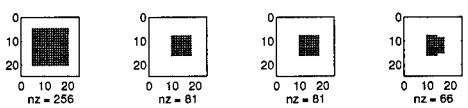

0
10
20
0 10 20
nz = 27
nz = 18
nz = 13
nz = 6

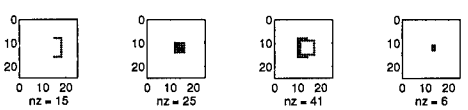

0
10
20
0 10 20
nz = 15
nz = 10
nz = 14
nz = 1


0
10
20
0 10 20
nz = 256


0
10
20
0 10 20
nz = 256


0
10
20
0 10 20
nz = 256
0
10
20
0 10 20
nz = 256
0
10
20
0 10 20
nz = 256
0
10
20
0 10 20
nz = 81


0
10
20
0 10 20
nz = 256
0
10
20
0 10 20
nz = 81
0
10
20
0 10 20
nz = 81
0
10
20
0 10 20
nz = 66


0
10
20
0 10 20
nz = 15
0
10
20
0 10 20
nz = 25
0
10
20
0 10 20
nz = 41
0
10
20
0 10 20
nz = 6

Appendix E. Pulse Coupled Neural Network Code (Matlab Implementation)

```
% Pulse coded neural network

% Written by Capt Randy Broussard.  Modified by Capt Don Hill


clear all


LRadius1=0;  % radius of linking

Beta=0;  % linking strength (was -.125)

Tl=5; %(10) Time constant of U pulse (Input + Linking signals)

Vl=5;  % Amplification of U signal (U pulse height)

Vf=5;

w=1; % all weights set to 1 for this case

Theta1_0=.1; %(.1) threshold, sets lowest input that can create a pulse

Theta2_0=10; % level 2 threshold

Ts=.5; % Time constant of inhibiting pulse (Theta)

Vs=10; % Amplif. of pulse inhibiting signal (Theta pulse height)


FRadius2=0; % (was 5) Feeding radius of linking in layer 2

LRadius2=0;  % (was 5) Linking radius of linking in layer 2

MaxRadius=max(FRadius2,LRadius2);

QF=4; % Spatial Kernel weight (larger Q, broader kernel)

QL=2;

Beta2= -.015;   % -.125; linking strength in layer 2 (was -.125)
```

```
%Tfd=1; % Decay time constant of F pulse (Input signal)

%Tfr=1; % Rise time constant of F pulse (Input signal)

%Vf=25; % Amplif. of input pulse signal


wF2=zeros((2*FRadius2+1),(2*FRadius2+1));

wL2=zeros((2*LRadius2+1),(2*LRadius2+1));


for i= 1:(2*FRadius2+1)

  for j= 1:(2*FRadius2+1)

    if (i==(FRadius2+1))

      wF2(i,j)=exp(-((abs(i-j))^2)/(2*QF^2));

    else

      wF2(i,j)=0;

    end;

  end;

end;

wF2


for i= 1:(2*LRadius2+1)

  for j= 1:(2*LRadius2+1)

    if (i==(LRadius2+1))

      wL2(i,j)=exp(-((abs(i-j))^2)/(2*QL^2));

    else
```

```
      wL2(i,j)=0;
    end;
  end;
end;
wL2
T1=exp(-1/Ts);
T2=exp(-1/Tl);
load box0.dat;
load box1.dat;
%load box2.dat;
load box3.dat;
% Input=box0;     %/DataMax;
load box5.dat;
Input=box5;
%DataMax=max(max(box1));
%Input=box1/DataMax;
%load pcnndata.dat;
%DataMax=max(max(pcnndata));
%Input=pcnndata/DataMax; %image must be padded for correctness
[Ysize,Xsize]=size(Input);
%Theta=Input>0;
%X=Theta;
Theta=zeros(Xsize,Ysize);
```

```
X=zeros(Xsize,Ysize);

L=zeros(Xsize,Ysize);

F=zeros(Xsize,Ysize);

F1=zeros(Xsize,Ysize);


Y=zeros(Xsize,Ysize);

L2=zeros(Xsize,Ysize);

F2=zeros(Xsize,Ysize);

Theta2=zeros(Xsize,Ysize);

figure(1);

clg;

%subplot(1,2,1);

%spy(Input);

figure(2);

clg;

%figure(3);

%clg;

%figure(4);

%clg;

pause


for n=1:25;
```

```
  if (n==1) %present first box at n=1
    %DataMax=max(max(box1));
    Input=box5;
    %Theta=Input>0;
    %X=Theta;
    %X=Input>0;
  end;


%  if (n==25) %present second box at n=50
    %DataMax=max(max(box2));
    %Input=box2/DataMax;
%    Input=box1;
    %Theta=Input>0;
    %X=Theta;
%  end;


%  if (n==50)
%    Input=box0;
%  end;
  ThetaOld=Theta;
% fprintf(1,'max F2=%f at begin of n=%d\n',max(max(F2)),n);
  ThetaOld2=Theta2;
  change=1; % Used to tell when linking oscilations have ended
```

```
  if(n==1)
    Redraw=1;  % Redraw screen to clear drawing of time 0 pulse
  else
    Redraw=0;  % Used to tell if screen needs redrawing
  end
  while(change>0);
    change=0;
    for row=1+LRadius1:Ysize-LRadius1  %loops that calculate the k-th neuron
for col=1+LRadius1:Xsize-LRadius1  %k-th neuron is at location row,col in matrix
Lk=0;
for or=(-LRadius1):LRadius1  %loops that sum the linking inputs
for oc=(-LRadius1):LRadius1
if(or~=0) & (oc~=0)
Lk=Lk+L(row+or,col+oc);
end
end
end
Fk=Input(row,col);  %normally would be sum of Fik's
%Fk=F1(row,col);
Uk=Fk*(1+Beta*Lk);
Theta(row,col)=Theta1_0+ThetaOld(row,col)*T1+Vs*X(row,col);
if(Uk>=Theta(row,col))
if(X(row,col)<1)
```

```
X(row,col)=1;

change=change+1;

Redraw=1;

end

end

%********************** debug probes ******************************

if (row==12)

if(col==12)

ThA1(n)=Theta(row,col);

UA1(n)=Uk;

end

% if(col==13)

% ThB1(n)=Theta(row,col);

% UB1(n)=Uk;

% end

% if(col==10)

% ThC1(n)=Theta(row,col);

% UC1(n)=Uk;

% end

end

% fprintf(1,'t=%d  change=%d   Y(%d,%d)=%d\n',t,change,row,col,Y(row,col));

%********************** end debug probes ***************************

end  %col
```

```
    end  %row
  end  %while changes
L=L*T2+Vl*X;  %weight is 1
%F1=(F1*T2)+Vf*Input;  %weight is 1


  change=1;
  while(change>0);
    change=0;
    for row=(1+MaxRadius):(Ysize-MaxRadius)  %loops that calculate the k-th neuron
for col=(1+MaxRadius):(Xsize-MaxRadius)  %k-th neuron is at location row,col in matrix
Lk=0;
for or=(-LRadius2):LRadius2  %loops that sum the linking inputs
for oc=(-LRadius2):LRadius2
if(or~=0) | (oc~=0)
Lk=Lk+L2(row+or,col+oc)*wL2(or+LRadius2+1,oc+LRadius2+1);
end
end
end
Fk=0;
for or=(-FRadius2):FRadius2  %loops that sum the linking inputs
for oc=(-FRadius2):FRadius2
Fk=Fk+F2(row+or,col+oc)*wF2(or+FRadius2+1,oc+FRadius2+1);
end  %oc
```

```
end         %or
  Uk=Fk*(1+(Beta2*Lk));
%Uk=Fk;
Theta2(row,col)=Theta2_0+ThetaOld2(row,col)*T1+Vs*Y(row,col);
%Theta2(row,col)=Theta2_0+(ThetaOld2(row,col)+(.05*Lk))*T1+Vs*Y(row,col);
if(Uk>=Theta2(row,col));
if(Y(row,col)<1)
Y(row,col)=1;
change=change+1;
Redraw=1;
end
end
%********************** debug probes ******************************
if (row==12)
if(col==12)
ThA2(n)=Theta2(row,col);
UA2(n)=Uk;
end
% if(col==13)
% ThB2(n)=Theta2(row,col);
% UB2(n)=Uk;
% end
% if(col==10)
```

```
% ThC2(n)=Theta2(row,col);
% UC2(n)=Uk;
% end
end
% fprintf(1,'t=%d  change=%d   Y(%d,%d)=%d\n',t,change,row,col,Y(row,col));
%************************ end debug probes ***************************
end  %col
    end  %row2
  end  %while changes
L2=(L2*T2)+Vl*Y;  %weight is 1
F2=(F2*T2)+Vf*X;  %weight is 1
%************************ Update graphs as necessary *****************
  fprintf(1,'n=%d\n',n);
  if(rem(n,1)==0) | (Redraw>0)
    figure(2);
    subplot(2,1,1);
    plot(1:n,ThA1,'r-',1:n,UA1,'g-');
    subplot(2,1,2);
    plot(1:n,ThA2,'r-',1:n,UA2,'g-');

    %figure(2);
    %subplot(3,2,3);
    %plot(1:n,ThB1,'r-',1:n,UB1,'g-');
```

```
  %subplot(3,2,4);
  %plot(1:n,ThB2,'r-',1:n,UB2,'g-');

  %figure(3);
  %subplot(3,2,5);
  %plot(1:n,ThC1,'r-',1:n,UC1,'g-');
  %subplot(3,2,6);
  %plot(1:n,ThC2,'r-',1:n,UC2,'g-');

end
if(Redraw>0)
%if(Redraw>2)
  figure(1);
  subplot(2,1,1);
  spy(X);
  title('Pulse Output');
  %figure(1)
  subplot(2,1,2);
  spy(Y);
  %title('Pulse Output - Layer 2');

  fprintf(1,'Pattern changed at time %d\n',n);
  TitleString=sprintf('Pulse%d.dat',n);
```

```
    title(TitleString);
    FID=fopen(TitleString,'w');
    for row=1+MaxRadius:Ysize-MaxRadius
for col=1+MaxRadius:Xsize-MaxRadius
  fprintf(FID,'%f ',Y(row,col));
end
fprintf(FID,'\n');
    end
    fclose(FID);
    pause(1);

    X=zeros(Ysize,Xsize);
    Y=zeros(Ysize,Xsize);
    figure(1);
    subplot(2,1,1);
    spy(X);
    subplot(2,1,2);
    spy(Y);
  end %if Redraw
end  %n
```

Appendix F. Multi-Modal Test Set-up Verification

Measurements for test set-up verification were carried out using a dual channel Hewlett Packard 7000 series storage oscilloscope. The output from the soundcard's speaker/headphone jack provided the input signal for channel 1 while the voltage generated from a photocell directed at the video screen provided input to channel 2. A graphical representation of this test setup is represented in figure 75. Prior to test verification, the size of the visual stimuli's black boxes was

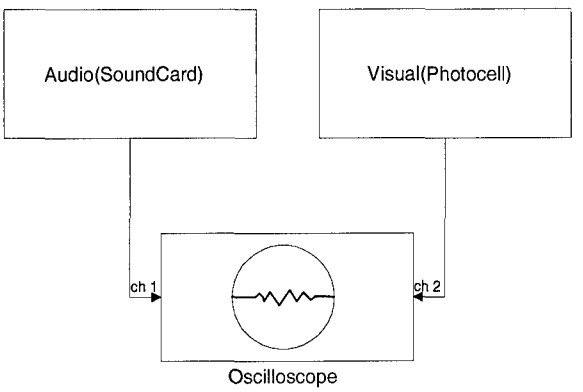


Figure 75. Time Delay Verification Setup

increased to 20x20 pixels to facilitate creation of a clean, easily recognized voltage level on the oscilloscope. Under normal test conditions, a selection of "go" would result in a call to erase the screen (except for the centered black cross-hair), followed shortly thereafter by the appearence of the first of the two boxes. Using the presence of one of the black boxes as a resting condition, the call to erase it from the screen resulted in a momentary rise in photocell voltage – until the box reappeared. Ideally, the oscilloscope would have been set to trigger on a negative slope at a level corresponding to the decrease in voltage caused by the appearence of the black box. However, illuminance of the video screen was not constant – the screen was refreshed every 20ms (50Hz refresh rate). As a result, the voltage seen by the photocell increased and decreased periodically.

To "capture" the temporal delay between onset of visual and audio stimuli, the oscilloscope was set to trigger off a rising slope of channel 2 (photocell voltage) at a level corresponding to the increased illumination seen by the photocell when the black box was erased. By maximizing the persistence of the oscilloscope for both channels 1 and 2, the temporal delay between visual and auditory stimuli could be measured and recorded. This procedure was repeated at 10 ms intervals for a range of delays between 0 and 40 ms. Since the timing delays generated by the hardware could not be totally insulated from the effects of the hardware's clock pulses, several measurements (30) were taken at each interval. The recorded measurements, as well as their corresponding mean and variance, are shown in table 28. The results of a pair-wise test of significance performed on the measured data is presented in table 29.

Table 28. Test Setup Timing Verification Results

	Calculated Delay							
Measurement	50	40	30	20	10	0	-10	-20
1	55	50	25	35	20	5	-10	-40
2	60	50	25	35	20	5	-20	-40
3	55	50	25	30	15	0	-30	-25
4	60	40	30	30	18	-5	-20	-30
5	60	30	30	15	8	-5	-5	-25
6	60	50	20	45	20	0	-18	-85
7	55	50	40	25	10	0	-10	-55
8	75	50	30	40	20	-5	-20	-55
9	50	40	45	35	20	-10	-30	-70
10	45	50	40	20	10	-10	-20	-70
11	48	30	20	20	10	-5	-5	-40
12	60	50	25	25	15	-10	-18	-40
13	70	50	40	35	5	-10	-10	-25
14	65	50	40	20	20	-5	-20	-30
15	68	50	35	20	10	0	-30	-25
16	50	40	35	35	10	0	-20	-55
17	60	30	28	30	5	-5	-5	-55
18	65	50	25	40	20	10	-18	-70
19	55	50	35	20	10	-5	-10	-40
20	55	50	20	30	5	5	-20	-25
21	60	40	20	40	20	0	-30	-30
22	65	30	20	35	0	5	-20	-25
23	60	50	45	30	10	-5	-5	-55
24	60	50	20	25	20	-10	-18	-55
25	55	50	30	45	10	0	-10	-70
26	60	40	20	45	15	-10	-20	-40
27	70	30	18	40	20	0	-30	-40
28	50	50	40	40	10	-10	-20	-25
29	70	50	35	30	5	-10	-5	-30
30	55	40	40	35	18	0	-18	-25
mean	39.2	44.7	30.0	31.7	13.3	-3	-17.2	-43.2
standard deviation	7.2	7.8	8.6	8.4	6.1	5.7	8.1	17.5

Table 29. Pairwise Test of Significance

Interval 1	*Interval 2*	*T*	*Accept NULL Hypothesis?*
50	40	7.6850	NO
50	30	13.2272	NO
50	20	14.9638	NO
50	10	27.6892	NO
50	0	38.7396	NO
50	-10	37.6622	NO
50	-20	32.2326	NO
40	30	8.3081	NO
40	20	6.3209	NO
40	10	19.2456	NO
40	0	26.8759	NO
40	-10	29.6481	NO
40	-20	23.6163	NO
30	20	-0.6628	YES
30	10	8.0924	NO
30	0	15.1669	NO
30	-10	24.1155	NO
30	-20	20.1945	NO
20	10	11.3282	NO
20	0	21.3365	NO
20	-10	20.3209	NO
20	-20	19.1221	NO
10	0	11.4075	NO
10	-10	13.6511	NO
10	-20	15.4061	NO
0	-10	7.3138	NO
0	-20	11.9227	NO
-10	-20	7.1544	NO

Appendix G. MultiModal Noncausality Computer Code

G.1 Source (.CPP) Code*

```
#define WIN31 1

#include <windows.h>

#include <mmsystem.h>

#include    <stdio.h>

#include    <stdlib.h>

#include    <math.h>


HWND HWindow;   // handle of main window

HANDLE ghInstance;  // instance number of main window

HWND Button1, Button2, Button3, Button4, Button5; // Handle to buttons in main window

int Rx1=260,Ry1=200,Rx2,Ry2; // rectangle corner points

int Rwidth=5, Rheight=5; // width and height of rectangle

int dx=500;  // stepsize for apparent motion dot

int BoxOnTime=50;   // in milliseconds

int BetweenBoxTime=50;  // in milliseconds

int SoundDelayTime=100;  // in milliseconds

int MovedRight=0; // # times visual stimuli appeared right first

int MovedLeft=0; // # times visual stimuli appeared left first

int GuessedRight=0; // # times test subject guessed right

int GuessedLeft=0; // # times test subject guessed left
```

```
int TestCount=0; // total # of test trials completed

int SilentCount=0; // # of silent test trials completed

int SyncCount=0; // # of in-order test trials completed

int OutofSyncCount=0; // # of out-of-order test trials completed

int Correct=0; // total # of correct choices made

int Incorrect=0; // total # of incorrect choices made

int SilentCorrect=0; // # of correct silent test choices

int SilentIncorrect=0; // # of incorrect silent test choices

int SyncCorrect=0; // # of correct in-order test choices

int SyncIncorrect=0; // # of incorrect in-order test choices

int OutofSyncCorrect=0; // # of correct out-of-order test choices

int OutofSyncIncorrect=0; // # of incorrect out-of-order test choice

int LRdir=0; // if 1, initial visual onset occurs left

int RLdir=0; // if 1, initial visual onset occurs right

int RLsound=0; // if 1, initial audio onset occurs left

int LRsound=0; // if 1, initial audio onset occurs right

int Nosound=0; // if 1, indicates wavefile contained no sound

int rnum; // dummy variable (used in case statements)

int flag=0; // dummy variable (used in randomization call)

long LoopsPerMilliSec; // indicates # iterative loops executed/ms

long Loops1; // dummy variable (used in time delay loop)

long delta=0; // temporary

long delta1=0; // temporary
```

```
WORD deviceID; // windows device id
WORD waveIDH, waveIDL, waveIDN; // windows wave file id

// sound functions (these functions were down-loaded from a BBS on
// the Net)

/*********************************************************************
 FUNCTION   :  ErrorProc(WORD)
 PURPOSE    :  ErrorProc calls mciGetErrorString to display an error
      message returned by MCI.
 COMMENTS   :  N/A
 HISTORY    :  N/A
*********************************************************************/
void FAR PASCAL ErrorProc(dwResult)
DWORD dwResult;
{
HANDLE hMem;
LPSTR lpStringBuff;
      hMem=GlobalAlloc(GHND,80);
if (hMem)
{
lpStringBuff=GlobalLock(hMem);
if (lpStringBuff)
```

```
{
if(mciGetErrorString(dwResult,lpStringBuff,80))   MessageBox(NULL,lpStringBuff,"ERROR",MB_OK
else
MessageBox(NULL,"Generic Error","ERROR",MB_OK);
GlobalUnlock(hMem);
}
else
MessageBox(NULL,"Lock Failed","ERROR",MB_OK);
GlobalFree(hMem);
}
else
MessageBox(NULL,"Alloc Failed","ERROR",MB_OK);
return;
}
/*********************************************************************

 FUNCTION   :  CloseAllDevices()
 PURPOSE    :  This function closes all currently open devices for the application.
 COMMENTS   :  N/A
 HISTORY    :  N/A

*********************************************************************/
BOOL FAR PASCAL CloseAllDevices()
```

```
{
DWORD dwRes;
// Close the global device
dwRes=mciSendCommand(MCI_ALL_DEVICE_ID,MCI_CLOSE,0,NULL);
if (dwRes)
{
ErrorProc(dwRes);
return(FALSE);
}
return(TRUE);
}

/*************************************************************************

 FUNCTION   :  SndPlaySnd(HWND,LPSTR)
 PURPOSE    :  This function demonstrates how to use the function
      sndPlaySound.  This is the easiest way to play a wave file.
 COMMENTS   :  The one setback to using sndPlaySound is that the entire
      sound is loaded into memory at once.  If you have a very
      large sound sndPlaySound will fail if the sound will not
      load into physical memory.
 HISTORY    :  N/A
```

```
*************************************************************************/

BOOL FAR PASCAL SndPlaySnd(lpWavefile)
LPSTR lpWavefile;
{
  WORD wFlags;
  MessageBox(NULL,lpWavefile,(LPSTR)"Playing Sound File",MB_OK);
  wFlags=SND_ASYNC;
  if (sndPlaySound(lpWavefile,wFlags))
return(TRUE);
  else
return(FALSE);
}

/*************************************************************************

 FUNCTION   :  OpenWaveDevice(HWND)
 PURPOSE    :  This function opens the device waveaudio and leaves it open.
      The device could also be opened by assigning the string "waveaudio"
      as follows:
       mciopen.lpstrDeviceType="waveaudio"
      Using this format you would only have to specify MCI_OPEN_TYPE for dwflags.
```

```
 COMMENTS   :  Opening the driver once and leaving it open saves time.
      You can then use this driver ID in the following
      calls to open elements (wavefiles).  This saves time since
      if you open an element without specifying an open device
      ID mci must open the driver and close the driver each
      time. When playing multiple elements (wavefiles) this
      excess time can be costly.  Therefore to conserve time you should
      open the waveaudio device, play the waves, close the
      waveaudio device.
 HISTORY    :  N/A

*****************************************************************************/

WORD FAR PASCAL OpenWaveDevice(void)
{
  MCI_OPEN_PARMS mciopen;
  DWORD dwRes;
  WORD wGlobalDeviceID;
  DWORD dwFlags;
  mciopen.wDeviceID=0;
  mciopen.lpstrDeviceType=(LPSTR)MCI_DEVTYPE_WAVEFORM_AUDIO;

  // when specifying the device ID you must also include the
```

```
        //MCI_OPEN_TYPE flag.

  dwFlags= MCI_OPEN_TYPE_ID | MCI_OPEN_TYPE;
  dwRes=mciSendCommand(0,MCI_OPEN,dwFlags,(DWORD)(LPSTR)&mciopen);
  if (dwRes)
{
ErrorProc(dwRes);
return(FALSE);
}
  wGlobalDeviceID=mciopen.wDeviceID;
  return(wGlobalDeviceID);
  }

/*********************************************************************

 FUNCTION   :  OpenWaveFile(HWND,WORD,LPSTR)
 PURPOSE    :  Open wave file for use by windows program
 COMMENTS   :  N/A
 HISTORY    :  N/A

*********************************************************************/

WORD FAR PASCAL OpenWaveFile(wGlobalDeviceID,lpWaveName,lpAlias)
```

```
WORD wGlobalDeviceID;
LPSTR lpWaveName;
LPSTR lpAlias;
{
  MCI_OPEN_PARMS mciopen;
  //MCI_PLAY_PARMS mciplay;
  DWORD dwRes;
  WORD wDeviceID;
  DWORD dwFlags;
  //char szWaveMessage[80];
  if (!wGlobalDeviceID)
 return FALSE;

  // Open the wave file independent of the device. This will speed up
  // the playing of the wavefile.

  dwFlags= MCI_OPEN_ELEMENT | MCI_OPEN_ALIAS;
  mciopen.lpstrElementName=(LPSTR)lpWaveName;

  // All strings must be initialized otherwise this will crash under
  // the debug version of mmsystem.

  mciopen.lpstrDeviceType="\0";
```

```
  mciopen.lpstrAlias=(LPSTR)lpAlias;

dwRes=mciSendCommand(wGlobalDeviceID,MCI_OPEN,dwFlags,(DWORD)(LPSTR)&mciopen);

  if (dwRes)
{
ErrorProc(dwRes);
return FALSE;
}

  // play the wave file.

  wDeviceID=mciopen.wDeviceID;

  return wDeviceID;
  }

/*****************************************************************

 FUNCTION   :   PlayWaveFile(wDeviceID)
 PURPOSE    :   This function takes a currently open device ID and a
       string representing a wave file and plays that wavefile.
  When the wavefile is finished playing the function closes
```

```
   the wavefile.
  COMMENTS  :   dwFlags is set to MCI_WAIT so that the function will not
   return until the wavefile is done playing.  If you want
   the function to return before the wavefile is finished
   playing omit the MCI_WAIT flag.
  HISTORY   : N/A

*************************************************************************/

BOOL FAR PASCAL PlayWaveFile(wDeviceID)
WORD wDeviceID;
{
   //MCI_OPEN_PARMS mciopen;

   MCI_PLAY_PARMS mciplay;
   DWORD dwRes;
   DWORD dwFlags;
   MSG msg;

         //char szWaveMessage[80];

   if (!wDeviceID)
  return FALSE;
```

```
  // Open the wave file independent of the device.  This will speed up
  // the playing of the wavefile.

  mciplay.dwFrom=0;

  dwFlags=MCI_FROM;

dwRes=mciSendCommand(wDeviceID,MCI_PLAY,dwFlags,(DWORD)(LPSTR)&mciplay);


  if (dwRes)
{
 ErrorProc(dwRes);
 return FALSE;
}

  while (PeekMessage(&msg,HWindow, NULL, NULL, PM_REMOVE))
   {
           TranslateMessage(&msg);
        DispatchMessage(&msg);
      }
  return TRUE;
```

```
}

/*********************************************************************

 FUNCTION   :   StopWaveFile(wDeviceID)

 PURPOSE    :   This function takes a currently open device ID and a

  string representing a wave file and stops play of that wavefile.

 COMMENTS   : N/A

 HISTORY    : N/A


*********************************************************************/

BOOL FAR PASCAL StopWaveFile(wDeviceID)

WORD wDeviceID;

{

  DWORD dwRes;

  DWORD dwFlags;

  if (!wDeviceID)

 return FALSE;


  // Open the wave file independent of the device. This will speed up

  // the playing of the wavefile.


  dwFlags=MCI_WAIT;
```

```
dwRes=mciSendCommand(wDeviceID,MCI_STOP,dwFlags,(DWORD)(LPVOID)NULL);
  if (dwRes)
{
 ErrorProc(dwRes);
 return FALSE;
}
  return TRUE;
  }

/*********************************************************************
 FUNCTION   :   CloseWaveFile(HWND,WORD,LPSTR)
 PURPOSE    :   This function takes a currently open device ID and a
  string representing a wave file and closes that wavefile.
 COMMENTS   :   N/A
 HISTORY    : N/A

*********************************************************************/
BOOL FAR PASCAL CloseWaveFile(wDeviceID)
WORD wDeviceID;
{
  DWORD dwRes;
  if (!wDeviceID)
```

```
 return FALSE;

  // Open the wave file independent of the device. This will speed up
  // the playing of the wavefile.

  dwRes=mciSendCommand(wDeviceID,MCI_CLOSE,0,NULL);
  if (dwRes)
 {
 ErrorProc(dwRes);
 return FALSE;
 }
  return TRUE;
}

/*********************************************************************

 FUNCTION  :   CloseWaveDevice(HWND,WORD)
 PURPOSE   :   This function closes a currently open waveaudio device.
 COMMENTS  : To conserve system resources you should always close a
  device when you are not using it.
 HISTORY   : N/A

*********************************************************************/
```

```
BOOL FAR PASCAL CloseWaveDevice(wGlobalDeviceID)

WORD wGlobalDeviceID;

{

  DWORD dwRes;

  if (!wGlobalDeviceID)

 return FALSE;


  // Close the global device


  dwRes=mciSendCommand(wGlobalDeviceID,MCI_CLOSE,0,NULL);

       .

  if (dwRes)

{

ErrorProc(dwRes);

return(FALSE);

}

  return(TRUE);

}


/******************* Drawing Functions **********************/



/*********************************************************************
```

```
 FUNCTION   :  void Drawline(int x1,int y1, int x2, int y2)

 PURPOSE    :  This function draws a line from coord (x1,y1) to (x2,y2):

 COMMENTS   :  N/A

 HISTORY    :  Written by Capt Randy Broussard


*********************************************************************/

void DrawLine(int x1,int y1,int x2,int y2)


{

HDC hdc;

hdc=GetDC(HWindow);

SelectObject(hdc, GetStockObject(BLACK_PEN));

MoveTo(hdc,x1,(int)y1);

LineTo(hdc,x2,y2);

ReleaseDC(HWindow,hdc);

}



/*********************************************************************

 FUNCTION   :  void PrintText(int x,int y,char * String)

 PURPOSE    :  This function prints  a text string beginning at coord (x,y):
```

```
 COMMENTS   :  N/A

 HISTORY    :  Written by Capt Randy Broussard



*********************************************************************/

void PrintText(int x,int y,char * String)


{

HDC hdc;


hdc=GetDC(HWindow);

SetTextColor(hdc,GetSysColor(COLOR_WINDOWTEXT));

SetTextAlign(hdc,TA_LEFT | TA_TOP | TA_NOUPDATECP);

TextOut(hdc,x,y,String,lstrlen(String));

ReleaseDC(HWindow,hdc);

}



/*********************************************************************


 FUNCTION   :  void EraseScreen()

 PURPOSE    :  This function erases a screen by painting a 640x400 rectangle

      with white
```

```
 COMMENTS   :  N/A

 HISTORY    :  Written by Capt Randy Broussard


*********************************************************************/

void EraseScreen()


{

HDC hdc;


hdc=GetDC(HWindow);

SelectObject(hdc,GetStockObject(WHITE_BRUSH));

Rectangle(hdc,0,0,640,400);

ReleaseDC(HWindow,hdc);

}



/*********************************************************************


 FUNCTION   :  void DrawBlackRectangle(int x1,int y1,int x2,int y2)

 PURPOSE    :  This function draws a rectangle from upper left

      corner at (x1,y1) to lower right corner at (x2,y2)

      and fills it in with black

 COMMENTS   :  N/A
```

```
 HISTORY    :  Written by Capt Randy Broussard

*************************************************************************/
void DrawBlackRectangle(int x1,int y1,int x2,int y2)

{
HDC hdc;

hdc=GetDC(HWindow);
SelectObject(hdc, GetStockObject(BLACK_PEN));
SelectObject(hdc, GetStockObject(BLACK_BRUSH));
Rectangle(hdc,x1,y1,x2,y2);
ReleaseDC(HWindow,hdc);
}



/*************************************************************************

 FUNCTION   :  void DrawWhiteRectangle(int x1,int y1,int x2,int y2)
 PURPOSE    :  This function draws a rectangle from upper left
      corner at (x1,y1) to lower right corner at (x2,y2)
      and fills it in with black
 COMMENTS   :  N/A
```

```
 HISTORY    :  Written by Capt Randy Broussard

*********************************************************************/
void DrawWhiteRectangle(int x1,int y1,int x2,int y2)
{
HDC hdc;

hdc=GetDC(HWindow);
SelectObject(hdc, GetStockObject(WHITE_PEN));
SelectObject(hdc, GetStockObject(WHITE_BRUSH));
Rectangle(hdc,x1,y1,x2,y2);
ReleaseDC(HWindow,hdc);
}


/*********************************************************************

 FUNCTION   :   int SilentDots()
 PURPOSE    :   This function, when called from main, erases the screen and places
a black cross-hair in the center.  At the same time, a call to the
sound card is made. After the given SoundDelayLoops have been
executed, a black dot is placed on either the left or right side of
the screen - depending on the random number generated.  After
```

```
BetweenBoxLoops, a second black dot is placed on the opposite side
of the screen.
 COMMENTS   :   The wavefile, 'NoSound.wav', contains only static - no audio tone
 HISTORY    :   N/A

**********************************************************************/
int SilentDots()
{
char TempString[80];
long Loops,BetweenBoxLoops,SoundDelayLoops;
DWORD Time1,Time2,Time3;

EraseScreen();
DrawBlackRectangle(318,195,322,215);
DrawBlackRectangle(310,203,330,207);

Rx2=Rx1+Rwidth;
Ry2=Ry1+Rheight;
BetweenBoxLoops=(long)BetweenBoxTime*LoopsPerMilliSec/2;
SoundDelayLoops=(long)SoundDelayTime*LoopsPerMilliSec/2;
for(Loops=0;Loops<=5000000L;Loops++);
if(random(2)==1)
{
```

```
LRdir=1;
Nosound=1;
PlayWaveFile(waveIDN);
for(Loops=0;Loops<=SoundDelayLoops;Loops++); //delay between visual-audio onset
DrawBlackRectangle(Rx1,Ry1,Rx2,Ry2);
for(Loops=0;Loops<=BetweenBoxLoops;Loops++); //delay between visual dot display
DrawBlackRectangle(Rx1+dx+Rwidth,Ry1,Rx2+dx+Rwidth,Ry2);
}
else
{
RLdir=1;
Nosound=1;
PlayWaveFile(waveIDN);
for(Loops=0;Loops<=SoundDelayLoops;Loops++); //delay between visual-audio onset
DrawBlackRectangle(Rx1+dx+Rwidth,Ry1,Rx2+dx+Rwidth,Ry2);
for(Loops=0;Loops<=BetweenBoxLoops;Loops++); //delay between visual dot display
DrawBlackRectangle(Rx1,Ry1,Rx2,Ry2);
}
return 0;
}

/*********************************************************************
```

```
 FUNCTION  :   int SyncDots()
 PURPOSE   :   This function, when called from main, erases the screen and places
a black cross-hair in the center.  At the same time, a call to the
sound card is made. After the given SoundDelayLoops have been
executed, a black dot is placed on either the left or right side of
the screen - depending on the random number generated.  After
BetweenBoxLoops, a second black dot is placed on the opposite side
of the screen.
 COMMENTS  :   The visual and audio stimuli are presented in order
 HISTORY   :   N/A

*************************************************************************/
int SyncDots()
{
char TempString[80];
long Loops,BetweenBoxLoops,SoundDelayLoops;
//DWORD Time1,Time2;

EraseScreen();

DrawBlackRectangle(318,195,322,215);
DrawBlackRectangle(310,203,330,207);
```

```
Rx2=Rx1+Rwidth;
Ry2=Ry1+Rheight;
BetweenBoxLoops=(long)BetweenBoxTime*LoopsPerMilliSec/2;
SoundDelayLoops=(long)SoundDelayTime*LoopsPerMilliSec/2;
for(Loops=0;Loops<=5000000L;Loops++);
if(random(2)==1)
{
LRdir=1;
LRsound=1;
PlayWaveFile(waveIDL);
for(Loops=0;Loops<=SoundDelayLoops;Loops++); //delay between visual-audio onset
DrawBlackRectangle(Rx1,Ry1,Rx2,Ry2);
for(Loops=0;Loops<=BetweenBoxLoops;Loops++); //delay between visual dot display
DrawBlackRectangle(Rx1+dx+Rwidth,Ry1,Rx2+dx+Rwidth,Ry2);
}
else
{
RLdir=1;
RLsound=1;
PlayWaveFile(waveIDH);
for(Loops=0;Loops<=SoundDelayLoops;Loops++); //delay between visual-audio onset
DrawBlackRectangle(Rx1+dx+Rwidth,Ry1,Rx2+dx+Rwidth,Ry2);
```

```
for(Loops=0;Loops<=BetweenBoxLoops;Loops++); //delay between visual dot display
DrawBlackRectangle(Rx1,Ry1,Rx2,Ry2);
}
return 0;
}


/*********************************************************************

 FUNCTION  :    int OutofSyncDots()
 PURPOSE   :   This function, when called from main, erases the screen and places
a black cross-hair in the center.  At the same time, a call to the
sound card is made. After the given SoundDelayLoops have been
executed, a black dot is placed on either the left or right side of
the screen - depending on the random number generated.  After
BetweenBoxLoops, a second black dot is placed on the opposite side
of the screen.
 COMMENTS  :   The visual and audio stimuli are presented out-of-order order
 HISTORY   :   N/A

*********************************************************************/
int OutofSyncDots()
{
```

```
char TempString[80];
long Loops,BoxOnLoops,BetweenBoxLoops,SoundDelayLoops;
EraseScreen();

DrawBlackRectangle(318,195,322,215);
DrawBlackRectangle(310,203,330,207);

Rx2=Rx1+Rwidth;
Ry2=Ry1+Rheight;
BetweenBoxLoops=(long)BetweenBoxTime*LoopsPerMilliSec/2;
SoundDelayLoops=(long)SoundDelayTime*LoopsPerMilliSec/2;
for(Loops=0;Loops<=5000000L;Loops++);
if(random(2)==1)
{
LRdir=1;
RLsound=1;
PlayWaveFile(waveIDH);
for(Loops=0;Loops<=SoundDelayLoops;Loops++); //delay between visual-audio onset
DrawBlackRectangle(Rx1,Ry1,Rx2,Ry2);
for(Loops=0;Loops<=BetweenBoxLoops;Loops++); //delay between visual dot display
DrawBlackRectangle(Rx1+dx+Rwidth,Ry1,Rx2+dx+Rwidth,Ry2);
}
else
```

```
{
RLdir=1;
LRsound=1;
PlayWaveFile(waveIDL);
for(Loops=0;Loops<=SoundDelayLoops;Loops++); //delay between visual-audio onset
DrawBlackRectangle(Rx1+dx+Rwidth,Ry1,Rx2+dx+Rwidth,Ry2);
for(Loops=0;Loops<=BetweenBoxLoops;Loops++); //delay between visual dot display
DrawBlackRectangle(Rx1,Ry1,Rx2,Ry2);
}
return 0;
}
```

```
/*************************************************************************

 FUNCTION  :    int CalibrateTimer()
 PURPOSE   :   This function is executed at the beginning of the program.  When
called, the screen is erased and the text 'calibrating timer -
please wait' placed in the middle.  The routing then makes a call
to the windows routine, GetCurrentTime, to get the current windows
time in milliseconds.  An arbitrarily large number of iterative
loops are then executed, after which the current windows time, in
milliseconds, is again retrieved. Since the total number of loops
```

```
executed is known, as well as the starting and stopping times for
execution of the loops, the number of loops/millisecond can be
calculated.  This number is then used throughout the program to
determine time intervals.
 COMMENTS   :   N/A
 HISTORY    :   Written by Capt Randy Broussard, modified by Capt Don Hill

**********************************************************************/
int CalibrateTimer()
{
char TempString[80];
long Loops;
DWORD Time1,Time2;

EraseScreen();

PrintText(200,140,"Calibrating Timer - Please Wait");
Time1=GetCurrentTime();
for(Loops=0;Loops<=50000000L;Loops++);
Time2=GetCurrentTime();
LoopsPerMilliSec=50000000L/(Time2-Time1);
wsprintf(TempString,"%ld loops performed per MilliSec",LoopsPerMilliSec);
PrintText(200,160,TempString);
```

```
PrintText(200,180,"Timer Calibration Completed");
randomize();
DrawLine(320,200,320,210); //draws black cross-hair
DrawLine(315,205,325,205);

return 0;
}

/******************** Change Parameters Dialog Box ***********************/
/*****************************************************************

 FUNCTION  :    far PASCAL ChangeParametersMsgLoop(HWND hDlg, WORD iMessage,
WORD wParam, LONG lParam)
 PURPOSE   :   This Dialog box, created using Turbo C++ Workshop, allows ascii
values to be entered into the Change Parameters box and changed
into integers for use throughout the program.  In addition, the
routine provides the user with an up-to-date look at various
program parameters
 COMMENTS  :   N/A
 HISTORY   :   Written by Capt Randy Broussard, modified by Capt Don Hill

*****************************************************************/
```

```
#pragma argsused
BOOL far PASCAL ChangeParametersMsgLoop(HWND hDlg, WORD iMessage, WORD wParam, LONG lParam)
{
char TempString[80];

switch(iMessage)
{
case WM_INITDIALOG:
itoa(dx,TempString,10);
SetDlgItemText(hDlg,101,(LPSTR)TempString);
itoa(Rwidth,TempString,10);
SetDlgItemText(hDlg,102,(LPSTR)TempString);
itoa(Rheight,TempString,10);
SetDlgItemText(hDlg,103,(LPSTR)TempString);
itoa(BetweenBoxTime,TempString,10);
SetDlgItemText(hDlg,104,(LPSTR)TempString);
itoa(SoundDelayTime,TempString,10);
SetDlgItemText(hDlg,106,(LPSTR)TempString);

return TRUE;
case WM_COMMAND:
switch(wParam)
{
```

```
case IDOK:
GetDlgItemText(hDlg,101,(LPSTR)TempString,sizeof(TempString));
dx=atoi(TempString);
GetDlgItemText(hDlg,102,(LPSTR)TempString,sizeof(TempString));
Rwidth=atoi(TempString);
GetDlgItemText(hDlg,103,(LPSTR)TempString,sizeof(TempString));
Rheight=atoi(TempString);
GetDlgItemText(hDlg,104,(LPSTR)TempString,sizeof(TempString));
BetweenBoxTime=atoi(TempString);
GetDlgItemText(hDlg,106,(LPSTR)TempString,sizeof(TempString));
SoundDelayTime=atoi(TempString);
EndDialog(hDlg,1);
return TRUE;
case WM_DESTROY:
EndDialog(hDlg,0);
return TRUE;
}
break;
}
return FALSE;
}

int ChangeParameters()
```

```
{
static FARPROC lpfnDialogProc;
int OkWasPressed=0;

lpfnDialogProc=MakeProcInstance((FARPROC)ChangeParametersMsgLoop,ghInstance);
OkWasPressed=DialogBox(ghInstance,"PARAMETERS",HWindow,lpfnDialogProc);
FreeProcInstance(lpfnDialogProc);
return(OkWasPressed);
}

/******************** TestStatus Dialog Box ******************/
/*****************************************************************

 FUNCTION  :    far PASCAL TestStatusMsgLoop(HWND hDlg, WORD iMessage, WORD
wParam, LONG lParam)
 PURPOSE   :   This Dialog box, created using Turbo C++ Workshop, allows ascii
values to be entered into the TestStatus box and changed
into integers for use throughout the program.  In addition,
this program provides the user with a current listing of all
test parameters
 COMMENTS  :   N/A
 HISTORY   :   Written by Capt Don Hill
```

```
**************************************************************************/

#pragma argsused

BOOL far PASCAL TestStatusMsgLoop(HWND hDlg, WORD iMessage, WORD wParam, LONG lParam)
{
char TempString[80];

switch(iMessage)
{
case WM_INITDIALOG:
itoa(MovedRight,TempString,10);
SetDlgItemText(hDlg,301,(LPSTR)TempString);
itoa(MovedLeft,TempString,10);
SetDlgItemText(hDlg,302,(LPSTR)TempString);
itoa(GuessedRight,TempString,10);
SetDlgItemText(hDlg,303,(LPSTR)TempString);
itoa(GuessedLeft,TempString,10);
SetDlgItemText(hDlg,304,(LPSTR)TempString);
itoa(Correct,TempString,10);
SetDlgItemText(hDlg,305,(LPSTR)TempString);
itoa(Incorrect,TempString,10);
SetDlgItemText(hDlg,306,(LPSTR)TempString);
```

```
    itoa(TestCount,TempString,10);
SetDlgItemText(hDlg,307,(LPSTR)TempString);
itoa(SilentCorrect,TempString,10);
SetDlgItemText(hDlg,308,(LPSTR)TempString);
itoa(SilentIncorrect,TempString,10);
SetDlgItemText(hDlg,309,(LPSTR)TempString);
itoa(SyncCorrect,TempString,10);
SetDlgItemText(hDlg,310,(LPSTR)TempString);
itoa(SyncIncorrect,TempString,10);
SetDlgItemText(hDlg,311,(LPSTR)TempString);
itoa(OutofSyncCorrect,TempString,10);
SetDlgItemText(hDlg,312,(LPSTR)TempString);
itoa(OutofSyncIncorrect,TempString,10);
SetDlgItemText(hDlg,313,(LPSTR)TempString);

return TRUE;
case WM_COMMAND:
switch(wParam)
{
case IDOK:
GetDlgItemText(hDlg,301,(LPSTR)TempString,sizeof(TempString));
MovedRight=atoi(TempString);
GetDlgItemText(hDlg,302,(LPSTR)TempString,sizeof(TempString));
```

```
MovedLeft=atoi(TempString);
GetDlgItemText(hDlg,303,(LPSTR)TempString,sizeof(TempString));
GuessedRight=atoi(TempString);
GetDlgItemText(hDlg,304,(LPSTR)TempString,sizeof(TempString));
GuessedLeft=atoi(TempString);
GetDlgItemText(hDlg,305,(LPSTR)TempString,sizeof(TempString));
Correct=atoi(TempString);
GetDlgItemText(hDlg,306,(LPSTR)TempString,sizeof(TempString));
Incorrect=atoi(TempString);
GetDlgItemText(hDlg,307,(LPSTR)TempString,sizeof(TempString));
TestCount=atoi(TempString);
GetDlgItemText(hDlg,308,(LPSTR)TempString,sizeof(TempString));
SilentCorrect=atoi(TempString);
GetDlgItemText(hDlg,309,(LPSTR)TempString,sizeof(TempString));
SilentIncorrect=atoi(TempString);
GetDlgItemText(hDlg,310,(LPSTR)TempString,sizeof(TempString));
SyncCorrect=atoi(TempString);
GetDlgItemText(hDlg,311,(LPSTR)TempString,sizeof(TempString));
SyncIncorrect=atoi(TempString);
GetDlgItemText(hDlg,312,(LPSTR)TempString,sizeof(TempString));
OutofSyncCorrect=atoi(TempString);
GetDlgItemText(hDlg,313,(LPSTR)TempString,sizeof(TempString));
OutofSyncIncorrect=atoi(TempString);
```

```
EndDialog(hDlg,1);
return TRUE;
case WM_DESTROY:
EndDialog(hDlg,0);
return TRUE;
}
break;
}
return FALSE;
}

int TestStatus()
{
static FARPROC lpfnDialogProc;
int OkWasPressed=0;

lpfnDialogProc=MakeProcInstance((FARPROC)TestStatusMsgLoop,ghInstance);
OkWasPressed=DialogBox(ghInstance,"STATUS",HWindow,lpfnDialogProc);
FreeProcInstance(lpfnDialogProc);
return(OkWasPressed);
}

/******************** Main Window Message proceedure ***********************/
```

```
long FAR PASCAL WndProc(HWND hWnd, unsigned iMessage, unsigned wParam, LONG lParam)
{
char TempString[100];
Rx1= 320-(dx/2)-Rwidth;

switch(iMessage)
{
case WM_COMMAND:
switch(HIWORD(lParam))
{
case WM_MENUSELECT:
default:
switch(wParam)
{
case 101:
ChangeParameters();
break;
case 102:
TestStatus();
break;
case 103:
```

```
PostQuitMessage(0);
break;
case 203:
GuessedLeft++;
TestCount++;
if (LRdir==1)
{
MovedRight++;
Correct++;
  switch(rnum)
{
case 0:
SilentCorrect++;
break;
case 1:
SyncCorrect++;
break;
case 2:
OutofSyncCorrect++;
break;
}
}
else
```

```
{
MovedLeft++;
Incorrect++;
switch(rnum)
{
case 0:
SilentIncorrect++;
break;
case 1:
SyncIncorrect++;
break;
case 2:
OutofSyncIncorrect++;
break;
}
}
RLdir=0;
LRdir=0;
RLsound=0;
LRsound=0;
if (TestCount==60)
{
wsprintf(TempString,"THIS CONCLUDES THE TEST");
```

```
PrintText(50,200,TempString);
break;
}
for(Loops1=0;Loops1<=2500000L;Loops1++);
wsprintf(TempString,"Ready");
                    PrintText(50,380,TempString);
break;

case 204:
GuessedRight++;
                    TestCount++;
if (RLdir==1)
{
MovedLeft++;
Correct++;
switch(rnum)
{
case 0:
SilentCorrect++;
break;
case 1:
SyncCorrect++;
break;
```

```
case 2:
OutofSyncCorrect++;
break;
}
}
else
{
MovedRight++;
Incorrect++;
switch(rnum)
{
case 0:
SilentIncorrect++;
break;
case 1:
SyncIncorrect++;
break;
case 2:
OutofSyncIncorrect++;
break;
}
}
}
```

```
RLdir=0;

LRdir=0;

RLsound=0;

LRsound=0;

if (TestCount==60)

{

wsprintf(TempString,"THIS CONCLUDES THE TEST");

PrintText(50,200,TempString);

break;

}

for(Loops1=0;Loops1<=2500000L;Loops1++);

wsprintf(TempString,"Ready");

PrintText(50,380,TempString);

break;


case 205:

//TestCount++;

//randomize();

flag = 0;

while(flag == 0)

{

rnum = random(3);

if((rnum == 0) && (SilentCount < 20))
```

```
{
SilentCount++;
SilentDots();
flag=1;
break;
}
else if ((rnum == 1) && (SyncCount < 20))
{
SyncCount++;
SyncDots();
flag=1;
break;
}
else if ((rnum == 2) && (OutofSyncCount < 20))
{
OutofSyncCount++;
OutofSyncDots();
                              flag=1;
break;
}
}
                        break;
                    case 105:
```

```
MovedLeft=0;

MovedRight=0;

GuessedLeft=0;

GuessedRight=0;

Correct=0;

Incorrect=0;

RLdir=0;

LRdir=0;

RLsound=0;

LRsound=0;

Nosound=0;

SilentCorrect=0;

SilentIncorrect=0;

SyncCorrect=0;

SyncIncorrect=0;

OutofSyncCorrect=0;

OutofSyncIncorrect=0;

TestCount=0;

SilentCount=0;

SyncCount=0;

OutofSyncCount=0;

EraseScreen();

wsprintf(TempString,"Test Count: %d ",TestCount);
```

```
PrintText(50,380,TempString);
break;

}
break;
}
break;
case WM_SIZE:
if(wParam==SIZEFULLSCREEN)
{
ShowWindow(HWindow, SW_SHOWNORMAL);
}
break;
default:
return DefWindowProc(hWnd, iMessage, wParam, lParam);
}
return(0L);
}

/******************** Create Main Window and Buttons *************/

#pragma argsused
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
```

```
LPSTR lpszCmdLine, int nCmdShow)

{

HWND hWnd;

MSG  Msg;

WNDCLASS WndClass;



ghInstance=hInstance;


if(!hPrevInstance)

{

WndClass.style=CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS;

WndClass.lpfnWndProc=WndProc;

WndClass.hIcon = LoadIcon(ghInstance,"apmotion");

WndClass.hbrBackground=GetStockObject(WHITE_BRUSH);

WndClass.hCursor=NULL;

WndClass.hInstance=hInstance;

WndClass.lpszMenuName="MAINMENU";

WndClass.lpszClassName="APMOTION";

WndClass.cbClsExtra=0;

WndClass.cbWndExtra=0;

RegisterClass(&WndClass);

}
```

```
hWnd=CreateWindow(
"APMOTION",
"Apparent Motion - Noncausality Test",
WS_OVERLAPPED|WS_CAPTION|WS_SYSMENU|WS_MINIMIZEBOX|WS_MAXIMIZE,
0,
0,
640,
480,
NULL,
NULL,
hInstance,
NULL);
HWindow=hWnd;
ShowWindow(hWnd, nCmdShow);

Button3=CreateWindow("BUTTON","GO",WS_CHILD|WS_VISIBLE|BS_PUSHBUTTON,
315,410,75,25,HWindow,205,ghInstance,NULL);
ShowWindow(Button3, SW_SHOW);


Button4=CreateWindow("BUTTON","Left",WS_CHILD|WS_VISIBLE|BS_PUSHBUTTON,
480,410,75,25,HWindow,203,ghInstance,NULL);
```

```
ShowWindow(Button4, SW_SHOW);
Button5=CreateWindow("BUTTON","Right",WS_CHILD|WS_VISIBLE|BS_PUSHBUTTON,
560,410,75,25,HWindow,204,ghInstance,NULL);
ShowWindow(Button5, SW_SHOW);

CalibrateTimer();

//open sound driver and .wav files

deviceID=OpenWaveDevice();
waveIDH=OpenWaveFile(deviceID,(LPSTR)"rlwd.wav",(LPSTR)"htone");
waveIDL=OpenWaveFile(deviceID,(LPSTR)"lrwd.wav",(LPSTR)"ltone");
waveIDN=OpenWaveFile(deviceID,(LPSTR)"nosnd.wav",(LPSTR)"notone");
while(GetMessage(&Msg,NULL,0,0))
{
TranslateMessage(&Msg);
DispatchMessage(&Msg);
}
//close .wav files and sound driver

CloseWaveFile(waveIDL);
CloseWaveFile(waveIDH);
CloseWaveFile(waveIDN);
```

```
CloseWaveDevice(deviceID);

return Msg.wParam;

}
```

*G.2 *.RC Code*

```
// ObjectWindows - (C) Copyright 1991 by Borland International


#include "windows.h"

#include "owlrc.h"


#include "inputdia.dlg"

#include "filedial.dlg"

#include "stdwnds.dlg"


#include "editacc.rc"

#include "fileacc.rc"


#include "editmenu.rc"

#include "filemenu.rc"


PARAMETERS DIALOG 22, 15, 141, 152

STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU

CAPTION "Parameters"

{

 EDITTEXT 101, 90, 12, 35, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP

 EDITTEXT 104, 90, 32, 35, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP

 RTEXT "Motion Step Size", -1, 21, 13, 65, 8
```

```
 RTEXT "Time Between Steps", -1, 15, 32, 70, 8
 EDITTEXT 102, 90, 78, 35, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
 EDITTEXT 103, 90, 99, 35, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
 EDITTEXT 106, 90, 56, 35, 12, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
 PUSHBUTTON "&Ok", IDOK, 19, 121, 40, 20, WS_CHILD | WS_VISIBLE | WS_TABSTOP
 PUSHBUTTON "&Cancel", IDCANCEL, 87, 120, 40, 20, WS_CHILD | WS_VISIBLE | WS_TABSTOP
 RTEXT "Rectangle Width", -1, 19, 81, 65, 8
 RTEXT "Rectangle Height", -1, 19, 102, 65, 8
 RTEXT "Visual Delay", -1, 20, 58, 63, 8
}


STATUS DIALOG 8, -16, 198, 189
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Status"
{
 RTEXT "TEST COUNT:", -1, 2, 4, 65, 8, SS_RIGHT | WS_CHILD | WS_VISIBLE | WS_GROUP
 EDITTEXT 307, 78, 2, 18, 12
 PUSHBUTTON "&OK", IDOK, 36, 166, 50, 14
 PUSHBUTTON "&Cancel", IDCANCEL, 115, 166, 50, 14
 RTEXT "INCORRECT:", -1, 105, 71, 60, 8, SS_RIGHT | WS_CHILD | WS_VISIBLE | WS_GROUP
 EDITTEXT 306, 175, 69, 18, 12
 EDITTEXT 302, 78, 18, 18, 12
 RTEXT "STARTED RIGHT:", -1, 7, 20, 60, 8
```

```
 EDITTEXT 303, 175, 18, 18, 12

 EDITTEXT 304, 175, 41, 18, 12

 RTEXT "GUESSED RIGHT:", -1, 104, 21, 64, 8

 RTEXT "GUESSED LEFT:", -1, 104, 44, 60, 8

 EDITTEXT 301, 78, 42, 18, 12

 RTEXT "STARTED LEFT:", -1, 7, 44, 60, 8

 EDITTEXT 305, 78, 69, 18, 12

 RTEXT "CORRECT:", -1, 8, 72, 60, 8

 //CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 0,   60, 198, 10

 EDITTEXT 308, 78, 105, 18, 12

 RTEXT "silent:", -1, 8, 108, 60, 8

 EDITTEXT 310, 78, 122, 18, 12

 RTEXT "sound-in-sync:", -1, 8, 125, 60, 8

 EDITTEXT 312, 78, 139, 18, 12

 RTEXT "sound-out-of-sync:", -1, 8, 142, 60, 8

 RTEXT "By Test Mode:", -1, 8, 90, 60, 8

 EDITTEXT 309, 175, 104, 18, 12

 RTEXT "silent:", -1, 105, 107, 60, 8

 EDITTEXT 311, 175, 121, 18, 12

 RTEXT "sound-in-sync:", -1, 105, 124, 60, 8

 EDITTEXT 313, 175, 138, 18, 12

 RTEXT "sound-out-of-sync:", -1, 105, 141, 60, 8

}
```

```
MAINMENU MENU
{
 POPUP "&Graph"
 {
  MENUITEM "&Change Parameters", 101
  MENUITEM "&Test Status", 102
  MENUITEM "&Reset Parameters", 105
  MENUITEM "E&xit", 103
 }

}

apmotion ICON
{
'00 00 01 00 01 00 20 20 10 00 00 00 00 00 E8 02'
'00 00 16 00 00 00 28 00 00 00 20 00 00 00 40 00'
'00 00 01 00 04 00 00 00 00 00 80 02 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 80 00 00 80 00 00 00 80 80 00 80 00'
'00 00 80 00 80 00 80 80 00 00 C0 C0 C0 00 80 80'
'80 00 00 00 FF 00 00 FF 00 00 00 FF FF 00 FF 00'
```

```
'00 00 FF 00 FF 00 FF FF 00 00 FF FF FF 00 99 99'
'99 99 99 99 99 99 99 99 99 99 99 99 99 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF 00 00 00 FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF 00 00 00 FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF 00 00 00 FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF 00 00 00 FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF 00 00 00 FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF F0 0F FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF 00 FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF 00 FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'00 00 00 0F FF FF FF FF FF FF FF FF FF F9 9F FF'
'00 00 00 0F FF FF FF FF FF FF FF FF FF F9 9F FF'
'00 00 00 0F FF FF FF FF FF FF FF FF FF F9 9F FF'
'00 00 00 0F FF FF FF FF FF FF FF FF FF F9 9F FF'
```

```
'00 00 00 0F FF FF FF FF FF FF FF FF FF F9 9F FF'
'00 00 00 0F FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 9F FF'
'FF FF FF FF FF FF FF FF FF FF FF FF FF F9 99 99'
'99 99 99 99 99 99 99 99 99 99 99 99 99 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
}
```

*G.3 *.DEF Code*

```
NAME          Ap_test


DESCRIPTION 'Multi-Modal Apparent Motion Experiment'


CODE          PRELOAD MOVEABLE DISCARDABLE

DATA          PRELOAD MOVEABLE MULTIPLE


EXETYPE       WINDOWS


SEGMENTS

        _RES  PRELOAD MOVEABLE DISCARDABLE


HEAPSIZE      128

STACKSIZE     8192
```

Bibliography

1. Boff, K.R. and J. E. Lincoln. *Engineering Data Compendium: Human Perception and Performance.*. AAMRL, Wright-Patterson AFB, OH, 1988.
2. Broussard, Randy. "PCNN Code." Code written as part of minor exam requirement toward AFIT PHD desertation, July 1995.
3. Burns, Thomas J. "Stephen Grossberg's Boundary Contour System." PHD Minor Exam Requirement, 1992.
4. Coren, Stanley, Clare Porac and Lawrence M. Ward. *Sensation and Perception* (2nd Edition). Academic Press, Inc.
5. Crick, Francis and Christoff Koch. "The Problem of Consciousness." *Scientific American* chapter 11, 125, W.H. Freeman and Company, 1993.
6. Eckhorn, R., H. J. Reitboeck M. Arndt and P. Dicke. "Feature Linking via Synchronization Among Distributed Assemblies: Simulations of Results from Cat Visual Cortex," *Neural Computation*, (2):293–307 (1990).
7. Francis, Gregory and Stephen Grossberg. "Cortical Dynamics of Form and Motion Integration: Persistence, Apparent Motion, and Illusory Contours." Soon to be published in Vision Research, 1994.
8. Gazzaniga. "Brain Mechanisms and Conscious Experience," *Experimental and Theoretical Studies of Consciousness*, 247–262 (1993).
9. Gray, Charles M., Peter Konig Andreas K. Engel and Wolf Singer. "Oscillatory Responses in Cat Visual Cortex Exhibit Inter-Columnar Synchronization Which Reflects Global Stimulus Properties," *Nature*, *338*:334–337 (March 1989).
10. Grossberg, Stephen. *Neural Networks and Natural Intelligence.* Cambridge, Massachusetts: MIT Press, 1988.
11. Grossberg, Stephen and Ennio Mingolla. "Neural Dynamics of Motion Perception: Direction Fields, Apertures, and Resonant Grouping," *Perception and Psychophysics*, *53*(3):243–278 (1993).
12. Grossberg, Stephen and Michael E. Rudd. "A Neural Architecture for Visual Motion Perception: Group and Element Apparent Motion," *Neural Networks*, *2*:421–450 (1989).
13. Grossberg, Stephen and Michael E. Rudd. "Cortical Dynamics of Visual Motion Perception: Short Range and Long Range Apparent Motion," *Psychological Review*, *99*:78–121 (1992).
14. Hinkle, Dennis E., William Wiersma and Stephen G. Jurs. *Applied Statistics for the Behavioral Sciences.* Chicago: Rand McNally College Publishing Company, 1979.
15. Johnson, John L. *Pulse Coupled Neural Networks.* Technical Report, U.S. Army Missile Command, Weapons Sciences Directorate, Redstone Arsenal, 1989.
16. Kabrisky, Dr Mathew, September 1995. contained in speech given at monthly SigArt meeting.
17. Korzeniewska, W. "Multisensory Convergence in the Thalamus of the Pigeon," *Neuroscience Letters*, *80*(1):55–60 (1987).

18. Libet, Benjamin. "Cortical Activation in Conscious and Unconscious Experience," *Perspectives in Biology and Medicine*, *9*:77–86 (1965).

19. Libet, Benjamin. "Responses of Human Somatosensory Cortex to Stimuli below Threshold for Conscious Sensation," *Science*, *158*:1597–1600 (1967).

20. Libet, Benjamin. "Conscious Subjective Experience vs. Unconscious Mental Functions: A Theory of the Cerebral Processes Involved." *Neurophysiology of Consciousness* 325–337, Birkhauser, 1993.

21. Libet, Benjamin. "The Neural Time Factor in Conscious and Unconscious Events," *Experimental and Theoretical Studies of Consciousness*, 123–146 (1993).

22. Libet, Benjamin. "The Neural Time-Factor in Perception, Volition and Free Will." *Neurophysiology of Consciousness* 367–383, Birkhauser, 1993.

23. Libet, Benjamin et al. "Production of Threshold Levels of Conscious Sensations by Electrical Stimulation of Human Somatosensory Cortex." *Neurophysiology of Consciousness* 2–33, Birkhauser, 1993.

24. Mangun, G. R. R. and S.A. Hillyard. "The Spatial Allocation of Visual Attention as Indexed by Event-Related Brain Potentials," *Human Factors*, *29*(2):195–211 (1987).

25. Ogmen, H. and S. Gagne. "Neural Network Architectures for Motion Perception and Elementary Motion Detection in the Fly Visual System," *Neural Networks*, *3*:487–505 (May 1990).

26. Rosenstengel, John. "Pulse Coupled Neural Networks." Paper presented to AFIT EENG 817 class during summer quarter, 1995.

27. Swanson, David E. *Retinal Modeling: Segmenting Motion from Spatio-Temporal Inputs Using Neural Networks*. MS thesis, Air Force Institute of Technology, 1992.

28. Wallace, Mark T., M. Alex Meredith and Barry E. Stein. "Integration of Multiple Sensory Modalities in Cat Cortex," *Experimental Brain Research*, *91*(3):484–488 (1992).

29. Warren, Richard M. and Roslyn P. Warren. "Auditory Illusions and Confusions," *Scientific American*, *223*:30–36 (December 1970).

30. Weiskrantz, Lawrence. "Unconscious Vision," *Sciences*, *32*(5):22–28 (September 1992).

31. Werblin, Frank S. "The Control of Sensitivity in the Retina," *Scientific American*, 70–79 (January 1973).

Vita

Captain Don E. Hill [redacted] a. He graduated from Chapin High School in 1987 and entered undergraduate studies at the United States Air Force Academy at Colorado Springs, Colorado. Captain Hill received the Bachelor of Science in Electrical Engineering degree from USAFA in 1991. Upon graduation, he received a Regular commission and was assigned to Los Angeles AFB, California where he worked initially as a Satellite Production Engineer in the Defense Support Program and later as an Engineering Intern in the Education with Aerospace Program. In May 1994, Captain Hill entered the School of Engineering, Air Force Institute of Technology.

[redacted]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1995	Master's Thesis

4. TITLE AND SUBTITLE
TEMPORAL INFLUENCE ON AWARENESS

5. FUNDING NUMBERS

6. AUTHOR(S)
Don E. Hill, Capt, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/ENG/95D-07

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Grossberg's Motion Oriented Contrast Filter(MOC) was extensively analyzed. The output from the filter's "global motion" neuronal layer was compared to a noncausal post-processing filter developed by AFIT. Both filters were shown to incorporate a weighted, noncausal temporal range of input data in processed output. The global motion framework was then implemented using a physiologically motivated pulsed neural model – the Pulse Coupled Neural Network(PCNN). By incorporating both spatial and temporal data, the PCNN was shown to exhibit a common visual illusion, apparent motion. The existence of a physiological temporal processing range was further investigated through implementation of two multi-modal experiments which integrated visual and auditory stimulus input channels. Results from the first experiment reinforce earlier findings from literature of a temporal window for perception of simultaneous activity. (Events occurring within this window are considered simultaneous; events which span more than one window are considered temporally separate.) Data collected from the second experiment suggests future inputs from an accessory auditory stimulus impact current perception of a visual stimulus. The influence of the auditory accessory stimulus decreases as the temporal delay between visual and auditory stimulus presentation is increased up to a maximum value of approximately 40 milliseconds. These tests results suggest the existence of perceptual noncausality in the mind – awareness as a function of past, current, and future perceptual inputs.

14. SUBJECT TERMS	15. NUMBER OF PAGES
PERCEPTION, APPARENT MOTION, NEURAL NETWORKS, MULTI-MODALITY	200
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.