REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate only, other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (07804-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(LEAVE BLANK)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	5 April 1999	Professional Paper

4. TITLE AND SUBTITLE

Sensor Modeling using a Universal Programmable Interface for Hardware-in-the-Loop IR/EO testing

5. FUNDING NUMBERS

6. AUTHOR(S)

Douglas C. McKee, Daniel B. Howe, Charles F. Ferrara

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Air Warfare Center Aircraft Division
22347 Cedar Point Road, Unit #6
Patuxent River, Maryland 20670-1161

8. PERFORMING ORGANIZATION REPORT NUMBER

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

Naval Air Systems Command
47123 Buse Road, Unit IPT
Patuxent River, Maryland 20670-1547

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A Universal Programmable Interface (UPI) has been developed by Amherst Systems to provide a configurable interface for hardware-in-the-loop (HWIL) testing of infrared and electro-optical (IR/EO) sensor systems. UPI development is funded under a contract to the Air Force Flight Test Center (AFFTC) at Edwards AFB, with additional funding support from the OSD Central Test and Evaluation Investment Program (CTEIP) Infrared Sensor Stimulator (IRSS) enhancement project office at NAWCAD Patuxent River, Maryland. The UPI supports the interface between a scene generation system (SGS) and the unit under test (UUT) through either direct injection of the signals into the system's processing electronics or the projection of in-band scene radiance into the sensor's optical aperture. To properly stimulate the UUT, the UPI should be able to simulate various sensor effects, emulate by-passed sensor components and reformat the data for direct injection or optical projection.

This paper will discuss the sensor modeling capabilities of the UPI and the supporting software and hardware. Sensor modeling capabilities include image blurring due to the sensor's modulation transfer function (MTF) and pixel effects. A sensor modeling and analysis software tool, based on FLI92[1], will be discussed. A technique for modeling other sensor effects will also be presented. This technique, called pixel displacement processing, can model geometric distortion, physical sensor jitter, and other user specified effects. It can also be used to accurately perform latency compensation.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Hardware-in-the-Loop, Sensor modeling, Infrared, Electro-Optic, Simulation	12
	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

DTIC QUALITY INSPECTED 4

19991004 301

Sensor modeling using a universal programmable interface for hardware-in-the-loop IR/EO testing

Douglas C. McKee[a], Daniel B. Howe[b], Charles F. Ferrara[c]

[a,b]Amherst Systems Inc., 30 Wilson Rd., Buffalo, NY 14221

[c]Integrated Sensors, Inc., 502 Court St., Utica, NY 13502

ABSTRACT

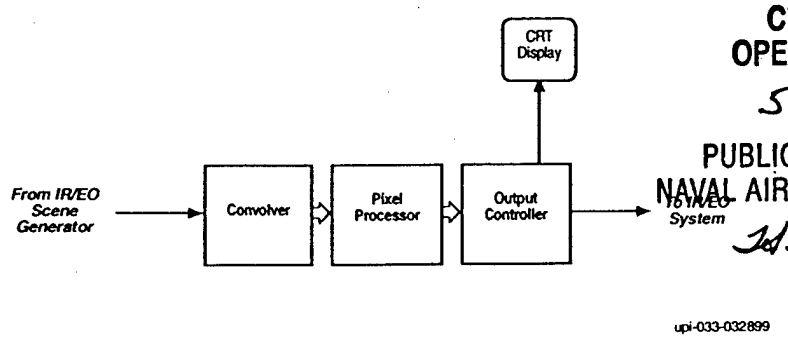
A Universal Programmable Interface (UPI) has been developed by Amherst Systems to provide a configurable interface for hardware-in-the-loop (HWIL) testing of infrared and electro-optical (IR/EO) sensor systems. UPI development is funded under a contract to the Air Force Flight Test Center (AFFTC) at Edwards AFB, with additional funding support from the OSD Central Test and Evaluation Investment Program (CTEIP) Infrared Sensor Stimulator (IRSS) enhancement project office at the Naval Air Warfare Center (NAWC/AD) at Patuxent River, MD. The UPI supports the interface between a scene generation system (SGS) and the unit under test (UUT) through either direct injection of the signals into the system's processing electronics or the projection of in-band scene radiance into the sensor's optical aperture. To properly stimulate the UUT, the UPI should be able to simulate various sensor effects, emulate by-passed sensor components and reformat the data for direct injection or optical projection.

This paper will discuss the sensor modeling capabilities of the UPI and the supporting software and hardware. Sensor modeling capabilities include image blurring due to the sensor's modulation transfer function (MTF) and pixel effects. A sensor modeling and analysis software tool, based on FLIR92[1], will be discussed.. A technique for modeling other sensor effects will also be presented. This technique, called pixel displacement processing, can model geometric distortion, physical sensor jitter, and other user specified effects. It can also be used to accurately perform latency compensation.

Keywords: Hardware-in-the-Loop, Sensor Modeling, Infrared, Electro-Optic, Simulation

1. INTRODUCTION

A UPI has been developed for the purpose of HWIL testing of IR/EO sensor systems. Unlike custom solutions, the UPI provides a re-configurable method for interfacing a wide range of UUTs through either direct injection of the signals into the system's processing electronics or the projection of in-band scene radiance into the sensor's optical aperture. This flexible capability is achieved through the use of a core architecture that provides industry standard interfaces, coupled with minimal custom interface hardware and re-configurable software. The UPI provides the physical interface between a SGS and the UUT, performs sensor modeling, and emulates missing components. The input scene is blurred in the convolver, pixel effects are introduced in the pixel processor, and the resultant image is output to the UUT. The top-level architecture of the UPI is illustrated in Figure 1-1.



CLEARED FOR OPEN PUBLICATION
5 Apr 99
PUBLIC AFFAIRS OFFICE
NAVAL AIR SYSTEMS COMMAND

Figure 1-1. Universal Programmable Interface

Hardware and sensor requirements have been presented in the past.[2,3] This paper focuses on the sensor modeling capabilities supported by the UPI, both in software and in hardware. Section 2 addresses the software used to model image blurring due to the sensor's MTF and pixel effects. A sensor modeling and analysis software tool, based on FLIR92, is used to generate the convolution kernel and simulate the pixel effects. Section 3 discusses pixel displacement processing, a technique that is used to accurately perform latency compensation and to model other sensor effects, including geometric distortion, physical sensor jitter and various additional user-defined effects. Section 4 provides an overview of the hardware used to implement the assorted sensor effects. Conclusions are presented in Section 5.

2. SENSOR MODELING SOFTWARE

A key capability requirement for the UPI is to accurately model the UUT's sensor characteristics, including modulation transfer function (MTF) and pixel effects. The sensor MTF is the frequency representation that describes image blurring, while the pixel effects include various noise sources and pixel non-uniformity. These effects are emulated in hardware components of the UPI, in the convolution and pixel processing subsystems respectively. However, off-line software tools are necessary to build the required inputs for these subsystems.

Software tools are currently being developed for sensor modeling. FLIR92, an industry standard tool for IR sensor characterization and performance modeling, was chosen as a basis for this software. A graphical user interface (GUI) has been developed that provides the capability for a user to enter sensor parameters matching those used by FLIR92. The dialog for required parameter entry is illustrated in Figure 2-1; other dialogs exist for the entry of optional sensor parameters. The UPI sensor modeling tool also supports both the import and export of FLIR92 files to provide file compatibility with this model. After the appropriate parameters are specified, this tool provides the user with the ability to generate the convolution kernels that model the sensor MTF. The details behind kernel generation are provided later in this section.

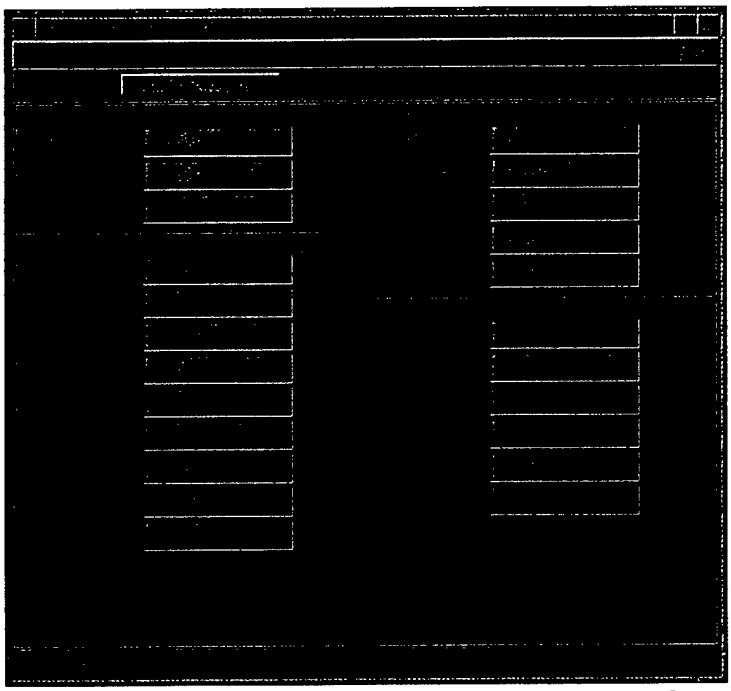


Figure 2-1. User Interface for Sensor Modeling Software

Software generated sensor modeling data files adds flexibility to the system. The tool's present functional capabilities provide the user with the ability to regenerate the kernels with any modifications to the sensor parameters. Therefore, the user can easily model numerous current sensors as well as potential future designs with no cost in hardware. Multiple sets of convolution kernels can be generated and stored for a given sensor system. This supports emulation of a multiple field of view (FOV) FLIR. When the FLIR's FOV is changed during a simulation run, the convolution subsystem will load in the set of kernels generated for the new FOV. Slight variations in each sensor's MTF in a multiple aperture system can also be emulated by generating a specific kernel for each sensor. Unique kernel sets can be generated for each sensor with small

changes in the sensor parameters. Although FLIR92 is used as a baseline for the UPI sensor modeling tool, other models that address capabilities not in FLIR92 can be incorporated in the future, for example NV-THERM. Due to the malleability of software, future sensor modeling refinements can be implemented with no hardware cost.

2.1 Sensor Modeling Prototype

The full implementation of sensor modeling software is presently in development. However, a functional prototype has been developed in PV-Wave. This prototype has been used to model the MTF and per-pixel effects for a second generation FLIR. Additionally, the software has been used for simulated Minimum Resolvable Temperature Difference (MTRD) tests. The experience gained from development and use of this prototype is being applied in the ongoing development effort.

2.1.1 Convolution Kernel Generation

A primary function of the prototype sensor modeling software is the generation of the convolution kernel that models the sensor's MTF. Similar to FLIR92, this software models the overall system MTF as the combination of component MTFs. Each component MTF is mathematically generated, based on the sensor parameters, using the same equations as FLIR92. However, an exception does exist with the electronics low pass MTF. FLIR92 only provides the magnitude of this MTF, but phase information is also needed to build a convolution kernel. Since the component MTFs are built in the frequency domain, combination is performed by multiplying them together as illustrated in Figure 2-2. The software currently being developed will additionally permit the input of measured MTF components in either the frequency or the spatial domain. If a component is specified in the spatial domain, it will be converted to the frequency domain through a Fast Fourier Transform (FFT).

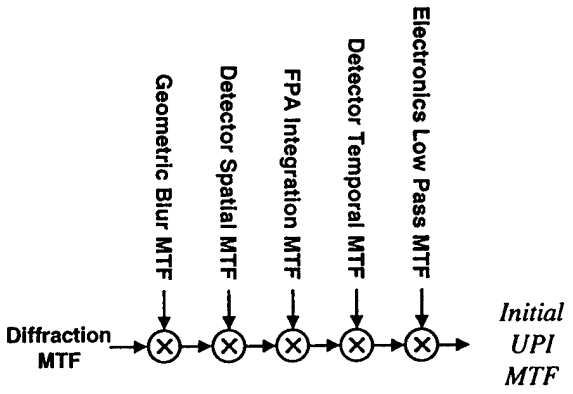


Figure 2-2. Sensor MTF Model

The convolution subsystem utilizes a spatial domain convolution kernel. Therefore, after the MTF is generated, it is converted to the spatial domain through an inverse FFT, as illustrated in Figure 2-3. The MTF is generated using a large array so that it can be accurately represented across multiple frequencies. This approach results in a large convolution kernel. However, practical UPI implementation constraints dictate that the kernel size be limited to 16 × 16 due to computational cost. Therefore, the kernel is truncated to the desired size and quantized to the appropriate fixed-point resolution.

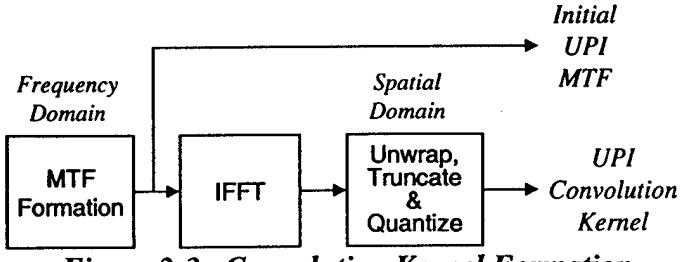


Figure 2-3. Convolution Kernel Formation

Although truncating the kernel lowers the computational cost, it can lead to inaccuracies and aliasing. The UPI sensor modeling tool provides the capability for the user to vary the kernel size and generate plots that display the effects of truncation. An example plot in Figure 2-4 illustrates the comparison of the vertical MTF profile pre-truncation (Initial MTF) and the MTF post-truncation (UPI MTF). The post-truncation MTF was created by transforming the truncated kernel back

into the frequency domain through an FFT. Clearly, the truncated kernel does not perfectly reproduce the MTF modeled with a non-truncated kernel. The slower roll-off of attenuation at the lower frequencies results in less blurring of the image than the actual sensor would encounter. Additionally, the ringing at higher frequencies can lead to aliasing distortions.

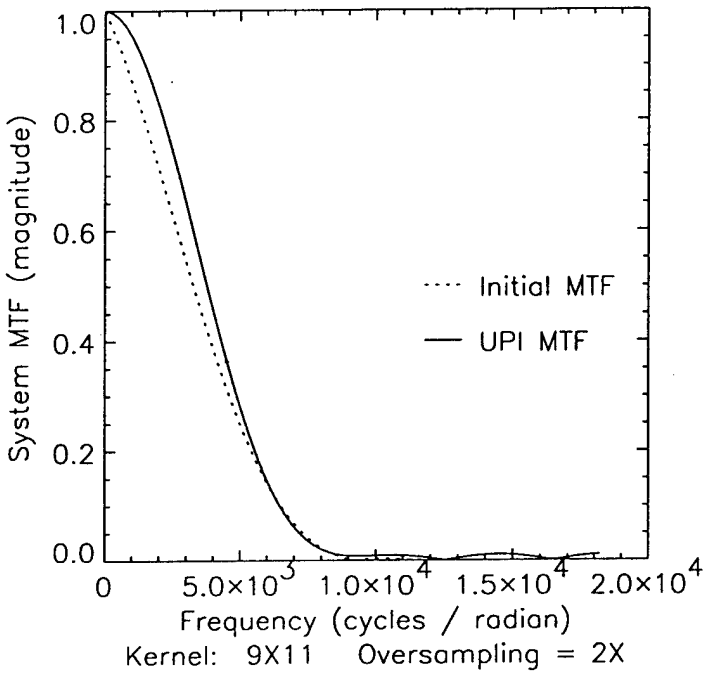
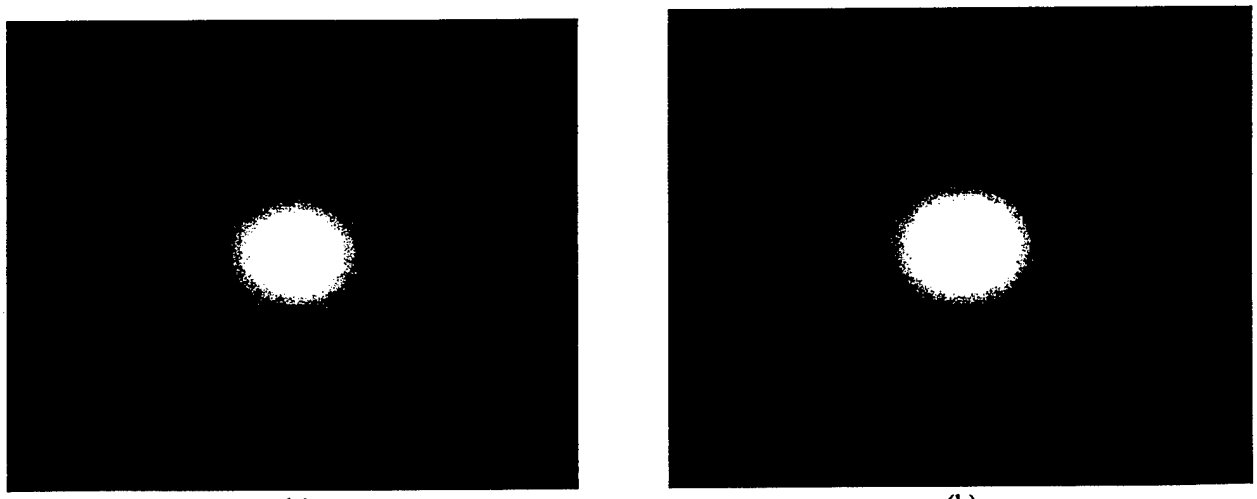


Figure 2-4. MTF Magnitude Vertical Profiles

The MTF can also be displayed fully in 2D, as shown in Figure 2-5. This figure illustrates that most of the ringing is along the horizontal and vertical axes with only minor ringing along the diagonals.



(a) (b)

Figure 2-5. UPI MTF. a) Pre-truncation, b) Post-truncation, raised to 0.25

The user also has the option to plot the MTF absolute differences, and/or use log scales to fully analyze the differences due to truncation. Although differences will exist, this tool can be used to help determine the best trade-off between kernel size and error. Convolution is performed on an over-sampled input scene. Therefore, the over-sampling factor can additionally be adjusted and its impact can be analyzed.

2.1.2 Pixel Effects Modeling

A variety of noise sources and response non-linearities that are present in sensors can affect system performance. Therefore, these pixel effects are modeled, based on user specified sensor parameters, in the pixel processing subsystem to add to the validity of the HWIL test. Since the pixel effects are implemented as software executing on general-purpose processors, a wide variety of effects can be modeled in the UPI. User-defined effects, within the limits of the UPI, can easily be added with no change or cost in hardware.

Pixel effects modeling is simulated in the present prototype software. As illustrated in Figure 2-6, a subset of pixel effects were chosen and implemented in this simulation. The radiance data from the convolution subsystem is first converted to photons. This is followed by the introduction of various noise sources, both multiplicative and additive. Linear responsivity models the system's analog-to-digital (A/D) converter by converting the photon count to fixed point units measurable by the A/D. Finally, the automatic gain control (AGC) normalizes the display to the intensity range of the image.

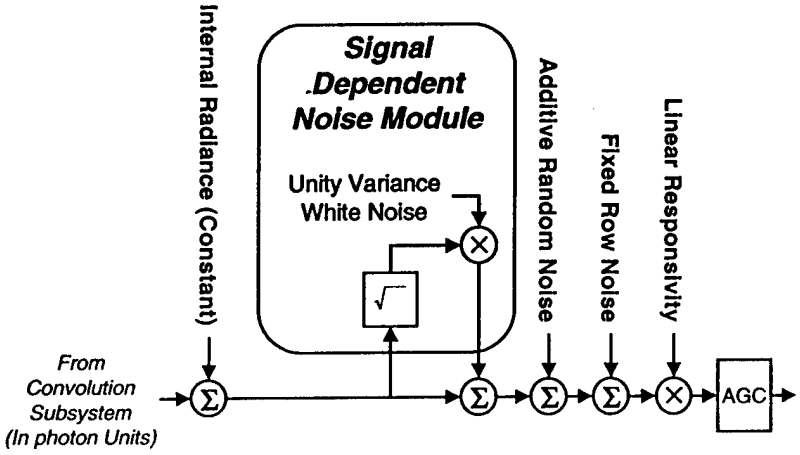


Figure 2-6. Simulated Pixel Processing Algorithm

2.1.3 Minimum Resolvable Temperature Difference (MRTD) Testing

The sensor modeling software supports simulated MRTD testing. A synthetic image consisting of a set of 40 four-bar targets is used as input. Convolution is performed on the image followed by pixel processing, resulting in a degraded image as illustrated in Figure 2-7. This process is repeated, varying ΔT, thereby, permitting the user to observe the sensor effects and to perform simulated MRTD testing. Preliminary comparisons between simulated testing and FLIR92 predicted results have shown very close agreement.

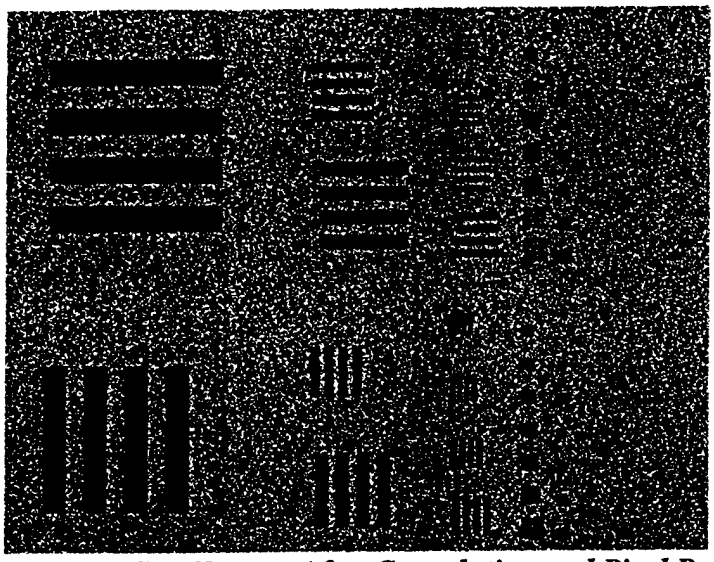


Figure 2-7. Four-Bar Targets After Convolution and Pixel Processing

3. PIXEL DISPLACEMENT PROCESSING

In the convolution subsystem, the over-sampled rendered scene is convolved with the MTF kernel, producing the image as seen by the sensor. Ideally, a static mapping between the sensor pixels and the scene pixels would exist. However, multiple factors can contribute to sensor pixel displacement from the perfectly rendered image to the correctly sensed image, including scene rendering latency, optics induced geometric distortion, and physical sensor jitter. The following discussion describes how these effects can be modeled in the UPI through pixel displacement processing. Accuracy concerns will be addressed including a method for sub-scene pixel mapping.

Displacement processing requires a mapping from each sensor pixel into scene space. Each map entry contains an x, y pair specifying where this sensor pixel is centered, in scene coordinates. During convolution, the kernel is applied to the location in the scene based on the map entry for the current sensor pixel.

An initial map must be generated which is subsequently modified by displacement processing. This map can either be stored explicitly or expressed as a mathematical function. An example map is shown in Figure 3-1 in which the sensor pixels are mapped into the center of the generated scene. The area covered by the sensor is less than that covered by the scene to allow for the displacement processing. The sensor pixels are more sparse than the scene pixels due to the over-sampling of the scene. If there were no pixel displacement, the initial map would be used directly for convolution processing. However, modifications to the map are necessary to model the above mentioned effects.

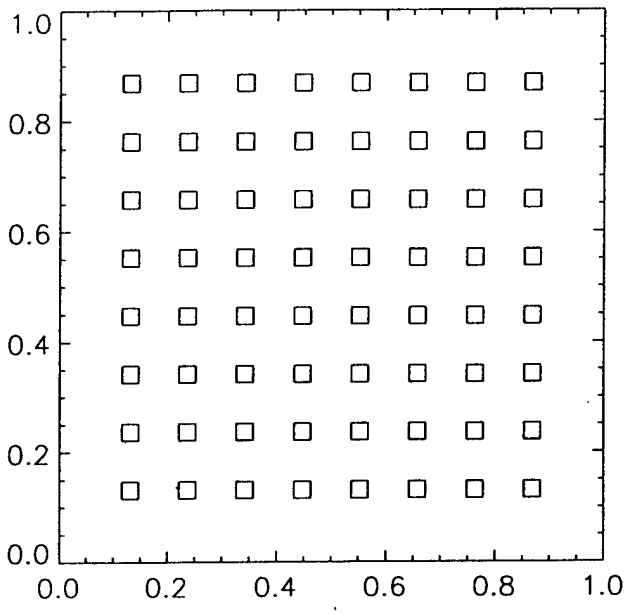


Figure 3-1. Initial Sensor to Scene Pixel Displacement Map

3.1 Latency Compensation

In a HWIL system, latency can occur between the time when positional data is received and when the scene is rendered. Latency can create errors in x and y shift, and in rotation, which can consequently affect the ability to accurately test the system under test (SUT). Therefore, the UPI performs latency compensation to extract the correctly located sensor image from within the oversized scene image. This can be accomplished by mathematically rotating and shifting the x, y pairs in the displacement map. Rotation is performed through the following equations:

$$\begin{aligned} x' &= x\cos(\theta) - y\sin(\theta) \\ y' &= x\sin(\theta) + y\cos(\theta) \end{aligned} \tag{1}$$

where x, y are the current coordinates stored in the displacement map, x', y' are the new coordinates and theta is the angle of rotation. Care must be taken to insure that the rotation takes place around the bore-sight of the sensor. The implementation may require a shift operation to accommodate this requirement.

Image shift is calculated as:

$$\begin{aligned} x' &= x + d_x \\ y' &= y + d_y \end{aligned} \tag{2}$$

where x, y are the current coordinates stored in the displacement map, x', y' are the new coordinates and d_x, d_y are the shifts in x and y respectively. Any post-rotation shift can be combined with this shift.

Figure 3-2 shows the result of rotation and shift of the displacement map coordinates.

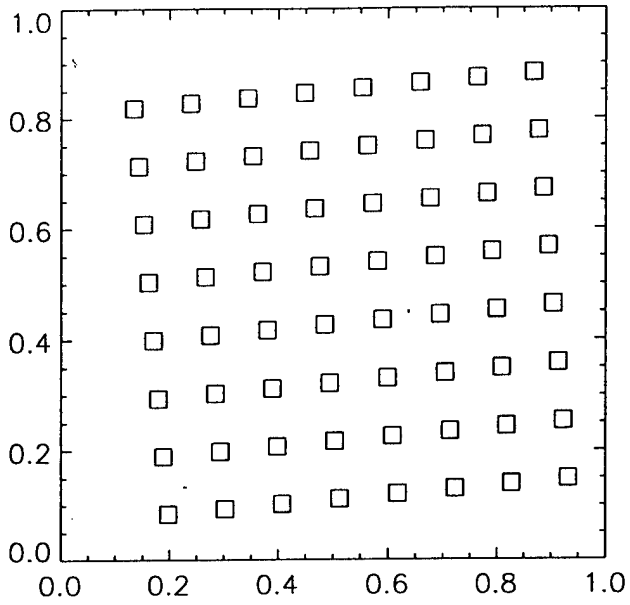


Figure 3-2. Rotated and Shifted Displacement Map

3.2 Geometric Distortion

Within a real sensor, the optics can generate geometric distortions in the resulting scene. Edges that are geometrically straight can appear curved, an effect that can be especially pronounced with wide field-of-view optics. If the scene generation system (SGS) does not perform the geometric distortion, then the compensation can be performed by the UPI as an additional sensor effect that can be modeled by this system.

Two common geometric distortions are pin-cushion and barrel effects. In both cases, there is a mapping between the true off-axis distance and the imaged off-axis distance. A function that generates these effects in polar coordinates is:

$$\rho' = (k\rho)^{\alpha}\rho \tag{3}$$

where ρ is the off-axis distance, k is a scaling factor, α is the distortion exponent, and ρ' is the resultant off-axis distance. The distance, ρ, is multiplied by the scaling factor, k: setting k equal to one over the maximum distance normalizes this factor to between 0 and 1. The distortion exponent, α, determines both the type and the amount of distortion. A positive α generates a pin-cushioning of the displacement map, while a negative α generates a barreling of the map. A zero valued α creates no distortion because any number raised to zero equals one. The magnitude of α determines the amount of distortion. An example displacement map after distortion is depicted in Figure 3-3. It should be noted that a pin-cushion displacement map yields a barrel effect image, and a barrel distortion map yields a pin-cushioned image. This is the result of the displacement map defining the mapping from the sensor to the scene. In the case of the pin-cushion displacement map, the inward bowing in scene coordinates of the regularly spaced sensor pixels creates an outward bowing of the resultant image.

A performance gain can be achieved if the geometric distortion is pre-computed for the specified UUT and stored in a displacement map. This is used as an initial displacement map prior to roll and shift latency compensation.

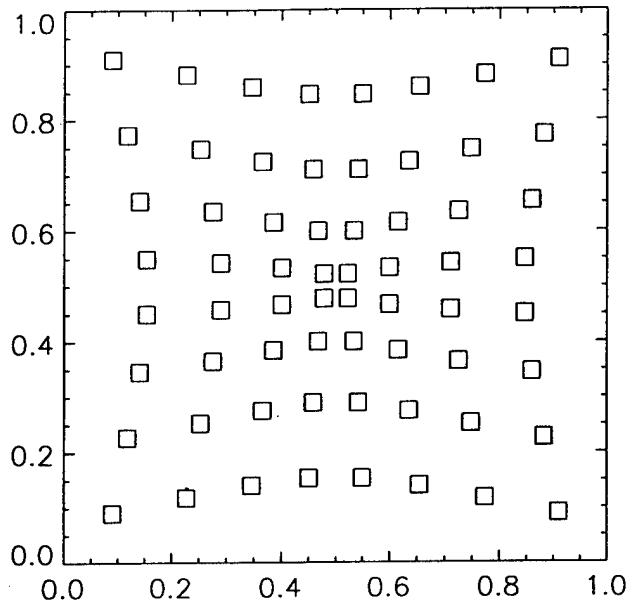


Figure 3-3. Pin-Cushioned Displacement Map

3.3 Other Displacement Functions

Sensors can experience physical jitter when mounted on a moving platform. For a scanning system, this could lead to small perturbations in the location of each successive scan line. Assuming the jitter function can be mathematically modeled, it too can be handled by the UPI. Potential jitter models can include sinusoidal or random per-scan-line or per-pixel effects. Pre-computed jitter offset values are sent to the convolution subsystem and added into the pixel displacement processing.

Displacement processing permits the implementation of any user-defined function that yields new x, y pairs based on the current x, y. Such user-defined functions can be combined with the other displacement effects. However, care should be taken in the order that the functions are executed. For example, functions that are not rotationally invariant must be applied before the rotational latency compensation function is applied. Additionally, the computational complexity of the function should be considered because these operations are performed on a per-sensor pixel basis.

3.4 Displacement Accuracy

This technique as described above allows for displacements with scene pixel accuracy; however, this may not be sufficient depending on the application. A solution is to increase the over-sampling factor used in scene generation. This process will increase the sub-sensor pixel accuracy, but it is limited by the computational resources of both the SGS and UPI. Therefore, a method is presented for displacements with sub-scene pixel accuracy.

Although convolution is often meant to blur objects, it is typically not intended to displace an object by a fractional scene pixel. However, sub-scene pixel displacement is desirable for displacement accuracy. Therefore, the goal is to generate a set of kernels that intentionally produce controlled sub-scene pixel displacements. A kernel is generated by sampling the point spread function (PSF) at intervals equal to the scene pixel width. Since the PSF can be considered a continuous function, it can be re-sampled starting at any chosen sub-scene pixel offset, allowing for control of sub-scene pixel displacements.

Ideally, an infinite number of kernels could be generated to account for the infinite number of sub-scene pixel offsets, but this obviously is not feasible. Therefore, an analysis was performed to determine how the number of kernels, n_k, affects the maximum error due to displacement. Two identical signals were generated: the first, the baseline, was generated with a sub-scene pixel shift and the second signal without. A convolution kernel was chosen for the second signal shifted to match the shift between the two signals. Convolution was performed on both signals and the maximum difference between the resultant

signals was calculated. This process was repeated with a varying number of kernels and a plot of the maximum error was generated as shown in Figure 3-4. The sub-scene pixel resolution is an inverse function of n_k; therefore, the error dropped off inversely with an increase in n_k as expected. Although the decrease in error is limited by an asymptote, it should be noted that this is maximum error calculated over a wide range of sub-scene pixel offsets. Some offsets produce less error for the given n_k, but none produce more than the plotted results.

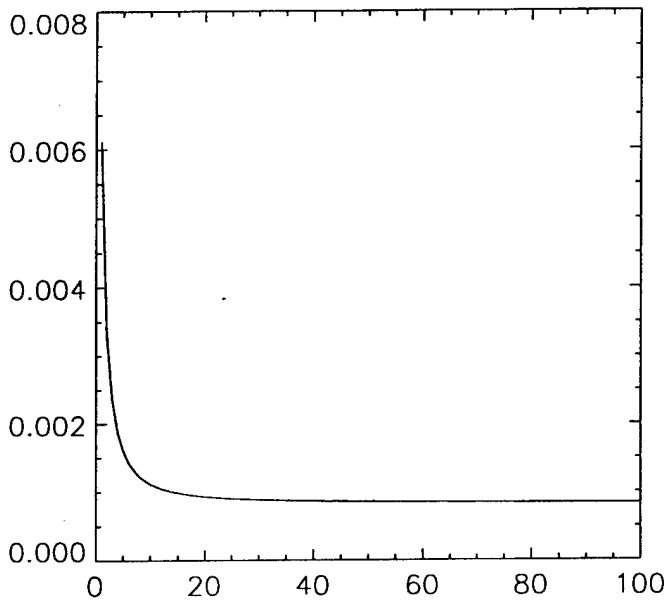


Figure 3-4. Maximum Error as a Function of the Number of Kernels

4. UPI SYSTEM ARCHITECTURE

The basic system architecture devised to support the sensor emulation processing discussed in the previous sections has evolved throughout the development process, but still adheres to the original concept. The primary design goals included cost-effectiveness per sensor channel, scalability, and algorithm flexibility. Figure 4-1 illustrates the resulting UPI architecture.

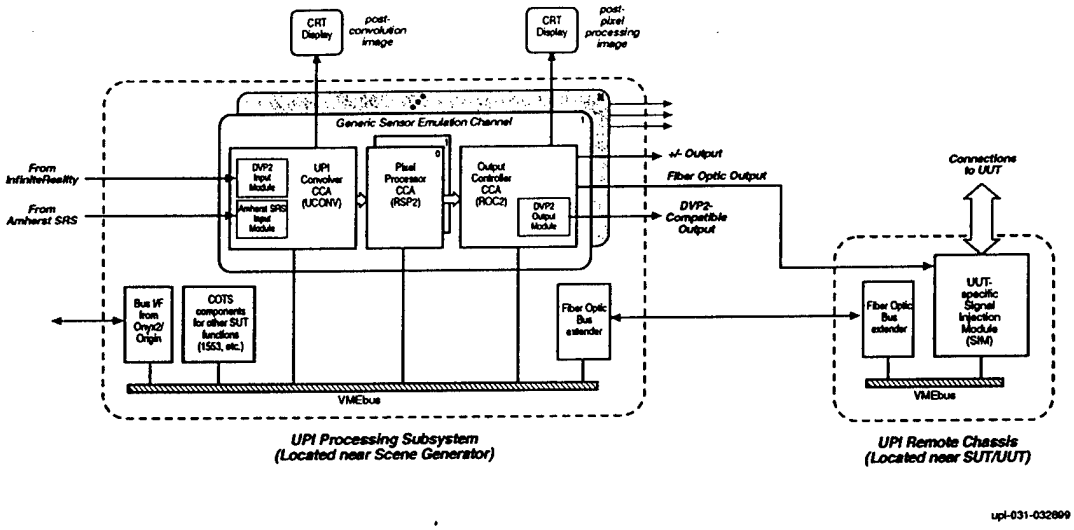


Figure 4-1. UPI System Architecture

In the context of the UPI, System Under Test (SUT) is generally used to refer to the entire system (i.e., aircraft, tank, ship, etc.) being tested, while Unit Under Test (UUT) is used to refer to a specific sensor aperture being stimulated. Even this distinction can become blurred in some cases, since a particular IR/EO subsystem on an aircraft (a missile warning system for example) might itself have multiple sensor apertures. Consequently, in these cases SUT might refer to the missile warning system as an entity, as well as the entire aircraft.

Whatever the case may be, the UPI architecture can be physically configured to support combinations of SUT/UUT subsystems. For example, to stimulate a missile warning system with four individual sensor heads, all four UPI generic sensor emulation channels might reside in the same processing chassis, along with any common functions needed to stimulate the missile warning system as a whole. The four individual Signal Injection Modules (SIMs) could then reside either in a single remote enclosure, or in separate ones, depending on the separation distances between sensor heads as installed on the SUT.

A principal goal of the UPI system/hardware development has been to develop the basic building blocks of the architecture, and to integrate and demonstrate these building blocks in an incremental fashion. This approach establishes a progression of demonstrable functionality, providing crucial proof of concept as well as design feedback. Toward this end, the first major functional milestone in the UPI development has recently been achieved and delivered to a customer for preliminary software development purposes, and is shown in Figure 4-2. Note that the UPI architecture re-uses circuit card assemblies from the Amherst Systems Real-time IR/EO Scene Simulator (RISS), to reduce risk and development costs.

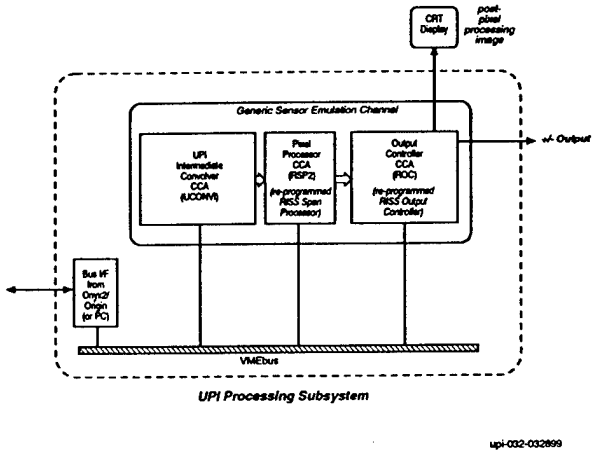


Figure 4-2. UPI Development and Integration - 1st Functional System

This system has been tested by injecting static images into the UPI Intermediate Convolver (UCONVI) circuit card assembly (CCA), loading test pixel processing software into the pixel processor (RSP2) CCA, and subsequently running the UPI channel at 200 frames per second with a 512 × 512 pixel image. Although working with a static image (and thus not exercising the input interfaces on the UCONVI), the remainder of the UPI channel is fully functioning at the frame rate. Various generic pixel processing algorithms were added to exercise loading the pixel processor processing elements, and the channel continued to operate at 200 frames per second. To test image size/rate scalability, the image size was then reduced to 512 × 128 pixels. The UPI channel was then able to sustain a frame rate of 1000 frames per second. Note that during any of these modes, the CRT display still functions. The display logic is programmed to display every other frame, every fourth frame, etc., as needed to provide a visual display independent of the UUT frame rate.

The next planned milestone is to integrate the Silicon Graphics Digital Video Port (DVP2)-compatible input module into the UPI, and process dynamic scenes through the UPI channel. The full UPI Convolver CCA (UCONV) currently in development will be added next (replacing the UCONVI), at which point the generic sensor emulation channel will be complete.

5. CONCLUSIONS

The UPI provides a re-configurable, cost-effective, modeling solution for HWIL IR/EO sensor system testing. A sensor modeling and analysis tool, based on FLIR92, provides the ability to model sensor blurring and per-pixel effects. Sensor parameters can be edited to model these effects in a wide range of current and future sensor systems. Pixel displacement processing can accurately perform latency compensation and model other sensor effects, including geometric distortion, physical sensor jitter and various additional user-defined effects. These modeling capabilities are afforded through a combination of both UPI software and hardware. It should additionally be mentioned that the UPI has been selected as the Signal Injection Subsystem (SIS) for the joint U.S. Navy and U.S. Air Force Infrared Sensor Simulator (IRSS) program. The SIS will be used to inject a scene into a UUT's digital data stream behind the optical sensor.

6. ACKNOWLEDGMENTS

This work was supported by the U.S. Air Force Flight Test Center (AFFTC) at Edwards Air Force Base, under Contract F04611-97-C-0077. Additional CTEIP funding of this work has been provided via the Infrared Sensor Stimulator (IRSS) Program Office at the Naval Air Warfare Center (NAWC/AD) at Patuxent River, MD.

7. REFERENCES

1. L. Scott, "Analysts Reference Guide", *FLIR92 Thermal Imaging Systems Performance Model*, U.S. Army NVESD report RG5008993
2. P. A. Acevedo, B. E. O'Toole, and T. E. Eisenhauer, "Prototype development of a universal programmable interface for hardware-in-the-loop sensor emulation", *Technologies for Synthetic Environments: Hardware-in-the-Loop Testing III*, Robert Lee Murrer, Editor, Proceedings of SPIE Vol. 3368, pp. 354-365, 1998.
3. D. B. Howe, P. A. Acevedo, and M. E. Nuwer, "Hardware-in-the-loop sensor emulation using a universal programmable interface", *Technologies for Synthetic Environments: Hardware-in-the-Loop Testing II*, Robert Lee Murrer, Editor, Proceedings of SPIE Vol. 3084, pp. 292-303, 1997.