INFRA-RED SENSOR SIMULATION

Sanjiv K. Bhatia
Dept. of Math & Computer Science
Univ. of Missouri – St. Louis
St. Louis, MO 63121

George M. Lacy
Visual Simulation Systems
FlightSafety International
St. Louis, MO 63042

Abstract

This paper describes technical/mathematical solutions for simulating infra-red sensor effects. We have implemented our simulation using a PC running Windows NT and off-the-shelf image processing hardware and software. In particular, we describe the computation of the dynamic characteristics of the actual sensor package within the constraints of hardware and software environment. These characteristics can include video polarity, gain, contrast enhancement, noise, blurring, AC coupling, sensor defects, as well as video overlays (reticules/test patterns), and are applied in the post-processor phase. This paper describes the research and development into the Infra-Red Post Processor (IRPP) algorithms needed to support the sensor simulation. The system performs all the operations in real-time with a 30 Hz refresh cycle. The IRPP is modular and can be easily changed by configuration data.

Sanjiv K. Bhatia received his Ph.D. from the University of Nebraska – Lincoln in 1991. He is presently working as an Associate Professor in the Department of Mathematics & Computer Science in the University of Missouri – St. Louis. His primary area of research is Image Databases, Digital Image Processing, and Computer Vision. He has published several papers on image databases and the application of knowledge-based techniques to information retrieval. He is a member of ACM and AAAI.

George M. Lacy is employed by the Visual Simulations Systems (VSS) Division of FlightSafety, International as a Systems Engineer. He is the senior engineer responsible for development of sensor simulations at VSS. Prior to joining VSS, Mr. Lacy spent 20 years as a United States Air Force Officer responsible for various engineering projects within the commands and laboratories. Mr. Lacy received his Masters of Science in Systems Engineering from the Air Force Institute of Technology and a Bachelor of Science in Engineering from Southern Illinois University at Carbondale. He has published papers on digital flight control systems. He is a member of ACM and IEEE.

INFRA-RED SENSOR SIMULATION

Sanjiv K. Bhatia
Dept. of Math & Computer Science
Univ. of Missouri – St. Louis
St. Louis, MO 63121

George M. Lacy
Visual Simulation Systems
FlightSafety International
St. Louis, MO 63042

INTRODUCTION

The Infra-Red Post Processor (IRPP) is part of a real-time infra-red simulation system. The simulated image is rendered by an image generator and the IRPP adds thermal characteristics to the image to mimic the look of the IR sensor being simulated [4]. These characteristics can include AC coupling, sensor defects, noise, blur, and smearing effects. The IRPP also emulates the gain, level, contrast enhancement, and polarity functions of the IR hardware being simulated. For a detailed discussion of sensor characteristics, please see references [5, 6].

Our IRPP implementation is based on performing the sensor functions [1] independent of the image generator. The image generator computes the radiance of objects in a database as a function of wavelength, environment, atmospherics, and thermal characteristics of the objects. However, the dynamic characteristics of the actual sensor are left for the post-processor. These characteristics can include video polarity (white hot/black hot), gain (manual/automatic), contrast enhancement, noise (static/dynamic), blurring, AC coupling, sensor defects (dead pixels/dead lines), as well as video overlays (reticules/test patterns).

In this paper, we describe the research and development into the algorithms needed to support the simulation in the PC environment. The algorithms had to be developed under the constraints imposed by the hardware and software environment, such as the absence of facilities to operate on individual pixels, which had to be simulated by computing convolution kernels.

In the next section, we provide an overview of the post-processor in context of the overall system. This is followed by a description of algorithms. In the last section, we provide the implementation status and the conclusion.

OVERVIEW OF THE POST PROCESSOR

The block diagram of the post processor is presented in Figure 1. The image generator (IG) generates the IR scene at the rate required by the cockpit display in the simulator. The scene is generated as a 640×480 pixel interlaced image at 30 Hz. frame rates. The video input and output are RS-170 format although other formats can be accommodated.

The frame grabber digitizes the analog video from the IR channel of the IG. The image is given to the image processor which applies the desired sensor characteristics. Finally, the image is converted back to analog by the image display controller.

The pilot controls are passed as a block of binary data over the ethernet. The controls are received by the IG who constructs the data block and sends it over the ethernet. These controls are used to modify the simulated behavior of the image processor, such as AGC window size adjustment, change in reticule overlay, polarity adjustment, and test pattern overlay.

IMAGE PROCESSING

Image processing is the most important phase in the post processor. It is here that the sensor characteristics are added to the image. The pilot-desired effects, such as polarity and gain and level control [3] are also added in this phase.

Figure 2 presents an overview diagram of the image processing functions used in the post processor.

Analog video from the IG may be gamma corrected. We undo the gamma correction through a 256×16 table lookup. The inverse gamma correction also converts the input data into 16-bit pixels to compensate for the absence of floating point operations in subsequent functions. We have assumed that the IG is applying a gamma of 2. The inverse gamma correction is applied by taking the square of the pixel value. Different gamma values can be accommodated.

In the following subsections, we describe the algorithms for various effects. The software has been designed such that each effect can be turned on or off by configuration controls.

AC Coupling

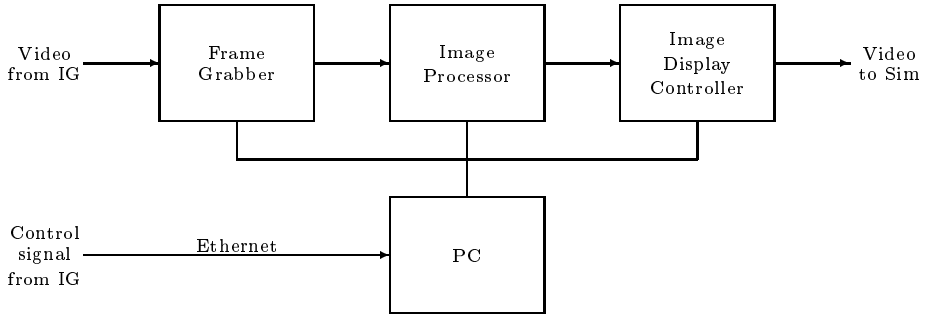

Figure 1: Block diagram of infra-red post processor

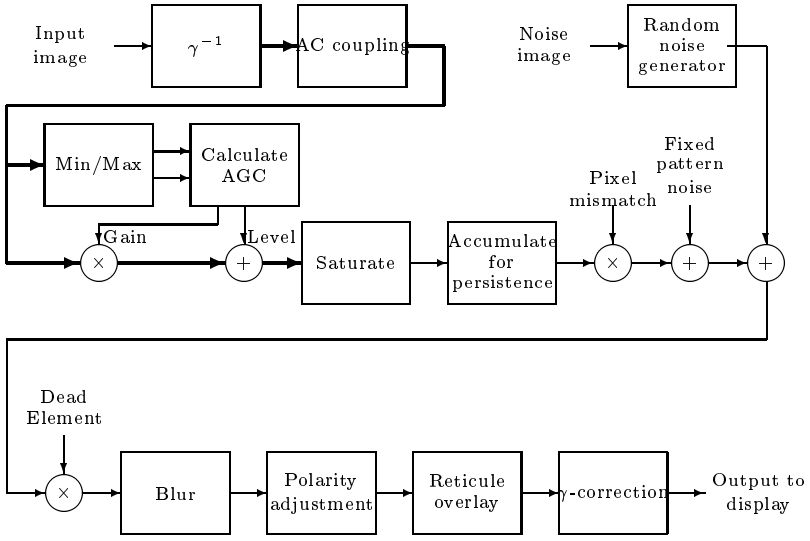

Figure 2: Image processing operations in the post-processor

The video data is processed through an infinite impulse response (IIR) digital filter to simulate AC coupling. AC coupling is a result of capacitive coupling of the IR sensor elements. It is modeled as a high pass RC filter consisting of a series capacitor and a resistor to ground. The IIR digital filter models the analog filter at discrete sample times corresponding to the pixel time. The pixels of the image are passed serially through the filter. The analog filter is described in Figure 3. It has an exponential response with a time constant given as the product of R and C.

The filter is implemented by

$$y_n = \frac{T}{T+1}(x_n - x_{n-1} + y_{n-1})$$

where T is the pixel time constant and x_n and y_n signify the corresponding pixels in the input and output images, respectively.

At first, the prototype was tested with a recursive function call reflecting the above recurrence. However, the function could not be used as such in the off-the-shelf environment. Therefore, we approximated the above equation through convolution using a one-dimensional kernel of 15×1. The size of 15 pixels was chosen as that is the largest kernel dimension available in the off-the-shelf graphics library. It can be modified to account for larger or smaller kernels.

The convolution is based on the expression for y_n. The major step in the convolution is the calculation of the kernel which is dependent on the time constant T. The computation can be simplified by using the constant c for the fraction such that the above equation is reduced to

$$y_n = c \cdot (x_n - x_{n-1} + y_{n-1})$$

where $c = \frac{T}{T+1}$.

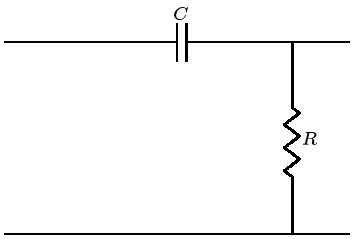

Figure 3: Analog filter for AC coupling effect

Solving the above recurrence yields the kernel for convolution. The equation for y_n, using a kernel of dimension $k \times 1$, is written as

$$y_n = c \cdot x_n - \sum_{i=0}^{k-1} c^{k-i}(1-c) \cdot x_i$$

Gain and Level Processing

Gain and level processing is used to enhance the contrast in the input data. Gain is implemented by multiplying each pixel of the frame by a value, level adds a value to each pixel. The post-processor provides for the options of no gain, manual gain, and automatic gain control. In the manual gain control mode, gain and level values can be specified via pilot controls. Automatic gain control (AGC) is used to optimize gain and level to stretch a specified window within the image for maximum contrast.

For AGC, the gain and level values are calculated from the minimum and maximum intensities of a specified rectangular window in the center of the image frame. The difference between the maximum and minimum pixel value (intensity) is denoted by δ and is used to calculate the gain. δ can be used in the computation as is or it can be reduced by a fractional amount at each end to allow the pixel values to stay between 0 (avoid negative values or underflow) and the maximum intensity value for the pixel (avoid overflow). The fraction of δ to discard at low end and high end of the spectrum are configurable lower limit and upper limit, respectively. After the gain and level are calculated from δ, they are clamped by configurable maximum δs from their past values. The windowed AGC capability is used for local area contrast enhancement in the simulation. We provide for three [configurable] window sizes that can be selected by pilot control.

Next, we present the calculations for gain and level for AGC using Figure 4. This figure shows the output intensity as a function of the input intensity. This is a linear transfer function where gain is given by the slope and level is given by the vertical-axis-intercept [2].

Let $I_{\max}$ be the maximum possible intensity value for any pixel. The minimum and maximum intensity values in the input image (the local window) are denoted by $i_{\min}$ and $i_{\max}$, respectively. $i'_{\min}$ and $i'_{\max}$ are used to denote the minimum and maximum intensity values after discarding part of the intensity range between $i_{\min}$ and $i_{\max}$. The gain G and level L are computed by

$$\begin{aligned}
\delta &= i_{\max} - i_{\min} \\
i'_{\min} &= i_{\min} + \delta \times \text{lower_limit} \\
i'_{\max} &= i_{\max} - \delta \times \text{upper_limit} \\
\delta' &= i'_{\max} - i'_{\min} \\
G &= I_{\max} \div \delta' \\
L &= -i'_{\min} \times I_{\max} \div \delta'
\end{aligned}$$

We are using our gain and level computations for local area contrast enhancement in the image. The contrast enhancement effect is presented in Figure 5 as a rectangular window in the center of the display. This small rectangular window shows contrast enhancement around part of the runway, making the tire marks and the area around the runway darker. This window shows the post gain computations. The rest of the image shows the pre gain computations.

Dynamic Noise

Dynamic noise is a per-pixel offset that simulates the dynamic noise from the imaging array of the IR sensor. The source for noise is a [pre-generated] Gaussian-distributed noise image. Each pixel of the image can be stretched by a random amount (up to 1, 2, 4, or 8 pixels) to simulate the characteristic streaking of the typical IR sensors. The image is then scaled to the appropriate amplitude. Both noise length and noise are configurable parameters. The dynamic nature of noise is simulated by scrolling the pixels by a

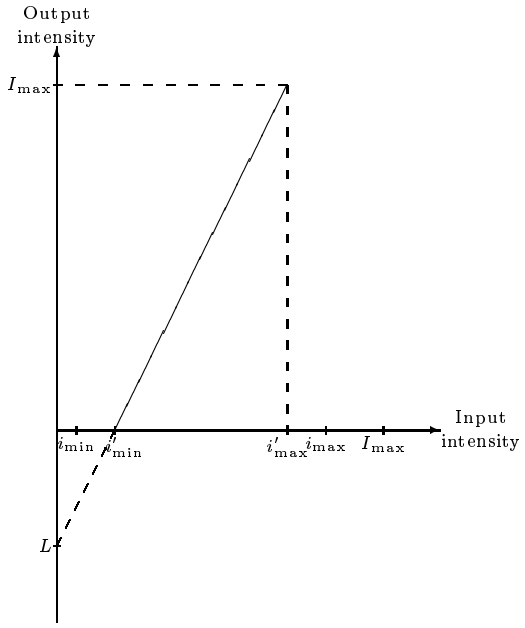

Figure 4: Calculation of gain and level for AGC

random amount for each frame, with the amount of scroll determined by the pseudorandom number generator. The scroll wraps around the bottom of the frame to keep the frame height for the noise constant.

Blur

The blurring effect is defocus through a 7×7 mask convolution. The amount of blur is a configurable and used to calculate the coefficients of the filter kernel.

The blurring effect uses a hard-coded 7×7 kernel though it can be easily changed into a configurable square kernel. The blur radius is specified as a fraction (between 0 and 1) to be used to select the area inside the kernel used for blurring. It is used to specify the weight for different kernel elements.

To compute the weight in the kernel, each pixel is considered to be a square with width 1 so that the 7×7 kernel has the width of each side as 7 pixels. Every element in the kernel is assigned a weight according to the extent to which it is present inside the circle described by the blur radius B_r, shown as the shaded portion in Figure 6. A pixel completely out-

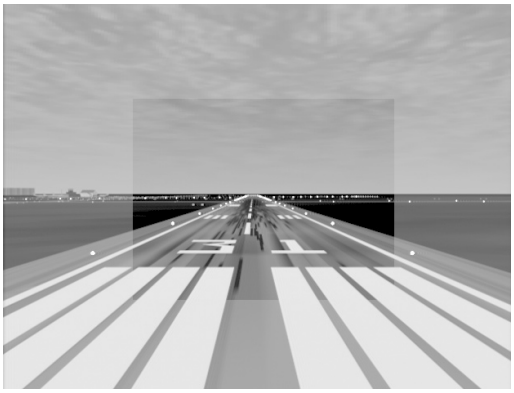

Figure 5: Local area contrast enhancement effect

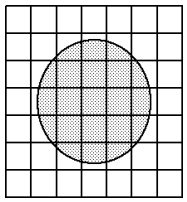

Figure 6: Computing the kernel for blurring

side the circle is assigned weight 0 while a pixel completely inside the circle is assigned weight 1. For the pixels that are partially inside the circle. the weight is computed as the fraction of the pixel inside the circle. The fraction is determined by the distance of the arc of the circle from the corner of the pixel such that the selected corner is nearest to the center of the circle. If D is the distance of the center of the pixel from the center of the kernel, the distance of the arc from the center of the pixel d is given by

$$d = B_r - D$$

$d > 0.5$ indicates that the pixel is completely inside the circle while $d < -0.5$ indicates that the pixel is completely outside the circle. For partially covered pixels, the weight is simply calculated to be $(d+0.5)$. If the weight for the pixel is denoted by w_{ij}, the kernel element is given by the expression

$$\frac{w_{ij}}{\sum w_{ij}}$$

Persistence

Persistence causes a smearing in the image when the sensor moves. This is simulated by computing a weighted average of the current frame and the previous frames. The weights applied to each frame is computed from an exponential decay function. The persistence amount is specified as a configurable time. We accumulate the weighted average of frames in a *history frame*. The computation of persistence coefficient with the exponential decay function gives the history frame F_n in terms of the current frame f_n and the last history frame F_{n-1} as

$$F_n = f_n + e^{-1 \cdot \frac{t}{T}} \cdot F_{n-1}$$

The above recurrence allows us to compute the persistence for the previous frames from the last frame displayed. It may be noted that

$$\lim_{i \to \infty} e^{-i} = 0$$

which implies that the accumulated frames can be normalized by dividing each pixel by a constant given by $\sum_{i=0}^{\infty} e^{-i\frac{t}{T}}$. This value can be easily computed for a given t and T.

The persistence effect is presented in Figure 7. It shows the lightpoint traces as the field of view shifts towards left.

Other effects

This subsection describes the effects shown in Figure 2 that have not been covered in the above subsections.

Saturation is used to convert the image from 16-bit pixel frame to unsigned 8-bit pixels. It is performed by right-shifting each pixel value by 7 bits. The pixels are shifted by 7 bits instead of 8 bits to account for one sign bit.

Pixel mismatch is a per-pixel gain that is used to simulate the non-uniform response of the imaging array of the IR sensor. This operation is performed by pixel-wise multiplication of the frame with a pre-generated frame.

The fixed pattern noise is a per-pixel offset that is used to simulate the fixed noise pattern of the imaging array of the IR sensor. It is performed by adding the [pre-generated] fixed noise buffer to the frame.

The dead element is an artifact in the sensing array that leads to a pixel not being scanned. The dead element appears as a horizontal one pixel-wide black scan line on the display frame. We have implemented the dead element as a configurable with random distribution. The number of dead elements is also a configurable. We generate a dead pixel frame where the dead pixel elements are denoted by 0. The frame being processed is pixel-wise multiplied by the dead pixel frame to simulate dead elements. We also have provisions for the dead elements being distributed across the frame instead of over scan lines (salt and pepper distribution).

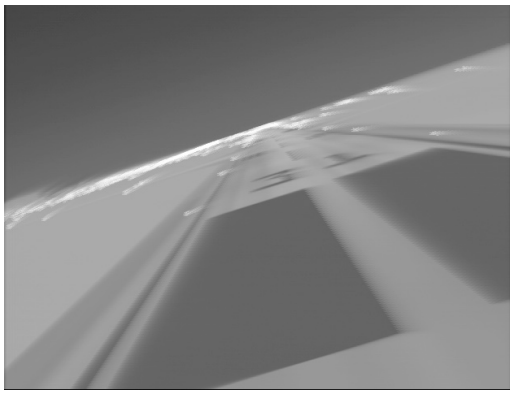

Figure 7: Persistence effect

Polarity inversion simulates the capability of the IR hardware to invert the video from "white hot" to "black hot". The inversion is invoked by the crew through ethernet interface. It is easily accomplished by taking a one's complement of each pixel in the frame.

Reticule overlay is performed by simply overlaying a pre-generated reticule frame on the image. Currently, we have two reticule frames that are generated during system initialization. The pilot can choose either one of the frames during simulation, or remove the reticule altogether. Addition of noise, reticule, and dead element is illustrated in Figure 8.

The gamma correction is a nonlinear mapping of the video to account for the nonlinear transfer characteristics of display devices in terms of light output versus the commanded intensity. A standard power law mapping is assumed. This is applied to the processed buffer using a lookup table as the last step before sending the frame to display.

IMPLEMENTATION

The post-processor is implemented using a Matrox Genesis graphics board using two circuit cards hosted on a standard Windows NT workstation. The graphics board provides the operations of frame-grabbing, image processing, and display control.

The current configuration is made up of three Texas Instruments C-80 DSP processing elements, and a neighborhood operations accelerator (NOA).

The post-processor divides the 640×480 pixel input video signal into three horizontal strips (each 640×120) as the frame is grabbed. Further processing is localized to the separate processing elements for each strip for most part. Some of the processing, such as frame grabbing and image processing, is multithreaded even within the same processing element.

The system has been designed such that the number of processing elements is determined during system initialization time. This allows the system to be ported on to a different graphics board configuration, with fewer or more processing elements.

The implementation has involved creation of higher-level interface to low-level library function calls using object-oriented concepts in C++. Thus, to move to a different hardware/software combination, we only need to modify the lower-level interface.

CONCLUSION

In this paper, we have described the algorithm development and implementation of an infra-red sensor simulation using a PC and off-the-shelf image processing environment. The algorithm development included software emulation.

The use of off-the-shelf hardware and software led us to make a number of decisions that affect the performance of the system. The most notable among those is the AC coupling effect which had to be implemented with an approximation kernel instead of a recursive routine.

To achieve acceptable processing throughput, we divided each frame into multiple strips that are processed separately on different processing nodes. This decision was complicated because of the use of kernels to refer to pixels on a strip assigned to different processing nodes which are not accessible to the current node, and also due to the multithreading aspects.

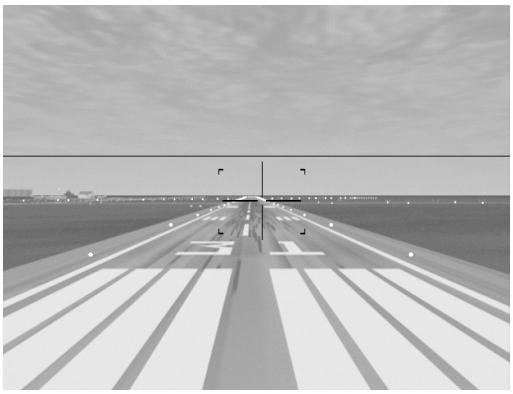

Figure 8: Addition of noise, reticule, and dead element

We had to minimize the number of synchronization calls between threads on different nodes as the speed advantage would be lost.

The system is implemented and is undergoing testing at the present time.

References

[1] G. E. Ball. New infrared opportunities in commercial aviation. *Advanced Imaging*, pages 22–24, April 1992.

[2] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison Wesley, Reading, MA, 1992.

[3] D. L. Peters. FLIR – A different world. In *Proceedings of the IMAGE V Conference*, pages 165–171, Phoenix, AZ, June 1990.

[4] B. J. Russo. Real time thermal imaging sensor simulation. In *Proceedings of the 1996 Image Conference*, Scottsdale, AZ, June 1996.

[5] D. L. Shumaker, J. T. Wood, and C. R. Thacker. *Infrared Imaging Systems Analysis.* DCS Corporation, 1988.

[6] W. L. Wolfe and G. J. Zissis. *The Infrared Handbook (revised ed.).* Office of Naval Research, 1985.