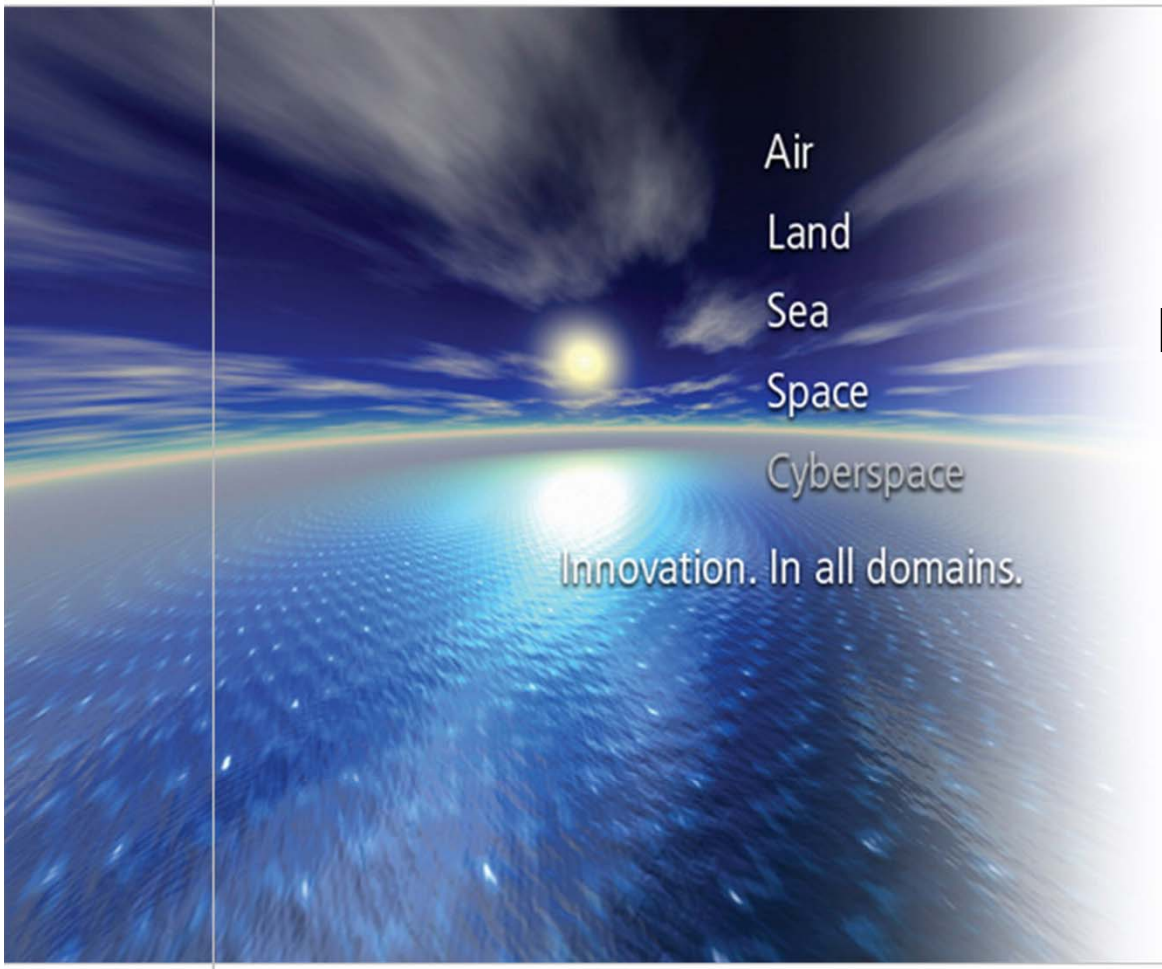
Raytheon

Customer Success Is Our Mission



Physics-Based Performance Analysis with High Fidelity, Dynamic, Real-Time EO Scene Generation

Michael Rivera, Ph.D.
michael.rivera@raytheon.com
Jonathan J. Buchanan
Jonathan_J_Buchanan@raytheon.com
Anibal Morales, MSCpE
anibal@raytheon.com

Modeling, Simulation & Analysis
Systems Design &Performance
Raytheon Missile Systems
Tucson, AZ
November 14-17, 2011

Raytheon

Outline

- Overview
  - Real time problem and challenges
- Motivation
  - Current state-of-the-art and our solution
- Image Chain
  - Missile image chain
- Scene Generation
  - How it fits into the closed loop pipeline
- EO Sensor Model
  - Summary, physics and challenges
- New Paradigm
  - CPU vs. GPU
- Real time pipeline, GPGPUs and CUDA
- Some examples
- Summary

Raytheon

Overview

- Physics–based modeling for any sensor system requires generating a model of the operating environment and modeling of the sensor characteristics
  - Scene generation: What the sensor actually is viewing
  - Sensor model: How the system sees the scene with its particular impairments
- Modeling to varying levels of fidelity is resource constrained
- Our Problem: Missile hardware development
  - Need to inject images into the sensor itself to mimic a dynamic environment in the laboratory
    - **Computer-in-the-loop:** a real time application with unique constraints
- Past solutions were a compromise
  - Realistic scenario simulation to present to the sensors vs. the rate at which simulated images can be feed to the sensors
- Our current solution
  - We developed a physics-based solution with a scene generator and high fidelity EO IR sensor model to exercise our hardware in a real time, closed loop laboratory environment for our computer-in-the-loop environments

Raytheon

Motivation

- A Computer-In-the-Loop (CIL) requires an image generator to run at faster than real time rates for injection into the tactical hardware and consumption by Signal Processing Algorithms
- Multiple deficiencies in present solutions
  - Canned videos: not dynamic
    - Perspective errors introduced and videos are made by running the system simulation or external tools “offline” and are therefore insensitive to dynamic changes
  - Specialized Hardware: not affordable
    - Non-scalable, requires a lot of developer time and effort, cost expensive
  - Model Decimation: not high fidelity
    - Removing model “fidelity,” excluding or simplifying models, not desirable since valuable verification and validation of native models is out the window. Can you now trust the results based upon the new model’s pedigree?
- **Our solution**: Accelerate high fidelity imaging chain models from the all-digital simulation (IFS) via General Purpose Graphical Processing Units (GPGPU) programming techniques
  - Produces dynamic, affordable, high fidelity images at faster than real time rates

Requirements	Canned videos	Specialized Hardware	Model Decimation	GPGPU
COTS				
RT				
DYNAMIC				
HIFI				

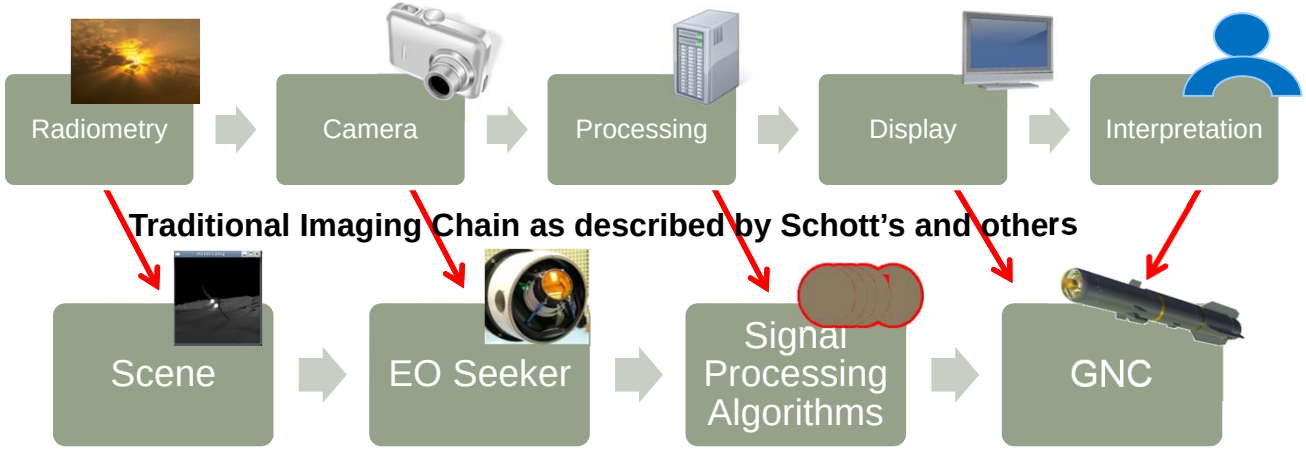
Meets
Compromise
Fails

Current real time imaging solutions are lacking, we fix the drawbacks of each with GPGPU acceleration

Raytheon

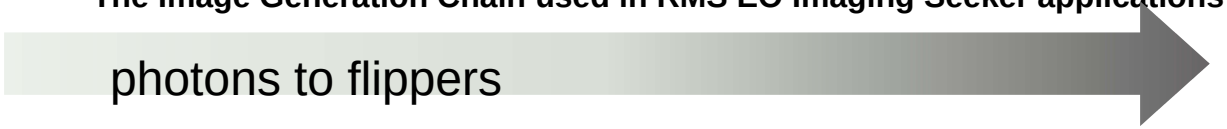
Image Generation Chain

- In our specialization of Schott's imaging chain approach, the image generation chain is an essential part of:
  - Closed Loop Computer In the Loop (CIL) simulations
  - Stand-alone studies
  - Training set generation for Signal Processing algorithm generation
  - All-digital Integrated Flight Simulations (IFS)



Traditional Imaging Chain as described by Schott's and others

The Image Generation Chain used in RMS EO Imaging Seeker applications



Schott, J.R., *Remote Sensing: The Image Chain Approach*, Oxford University Press, 2nd Edition, (May 2007)

The image chain is inherent in any EO/IR remote sensing system, from source to detector

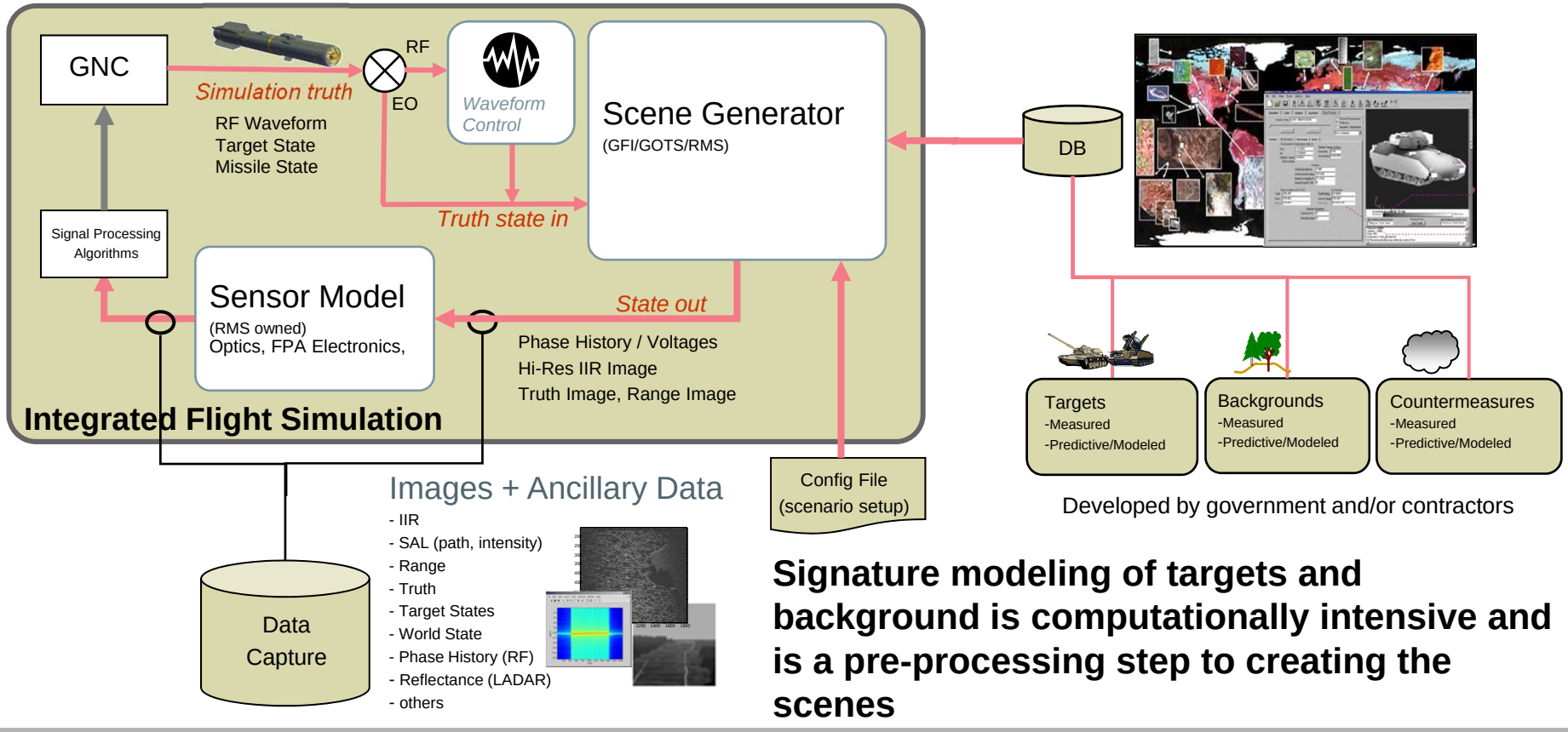
Raytheon

Outline

- Overview
  - Real time problem and challenges
- Motivation
  - Current state-of-the-art and our solution
- Image Chain
  - Missile image chain
- Scene Generation
  - How it fits into the closed loop pipeline
- EO Sensor Model
  - Summary, physics and challenges
- New Paradigm
  - CPU vs. GPU
- Real time pipeline, GPGPUs and CUDA
- Some examples
- Summary

Scene Generation: Defined

- Scene generator: A computer model that creates a **world and environment representation** using phenomenology, physics, and/or behavioral models, to achieve a useful rendition representing a spectral domain of interest

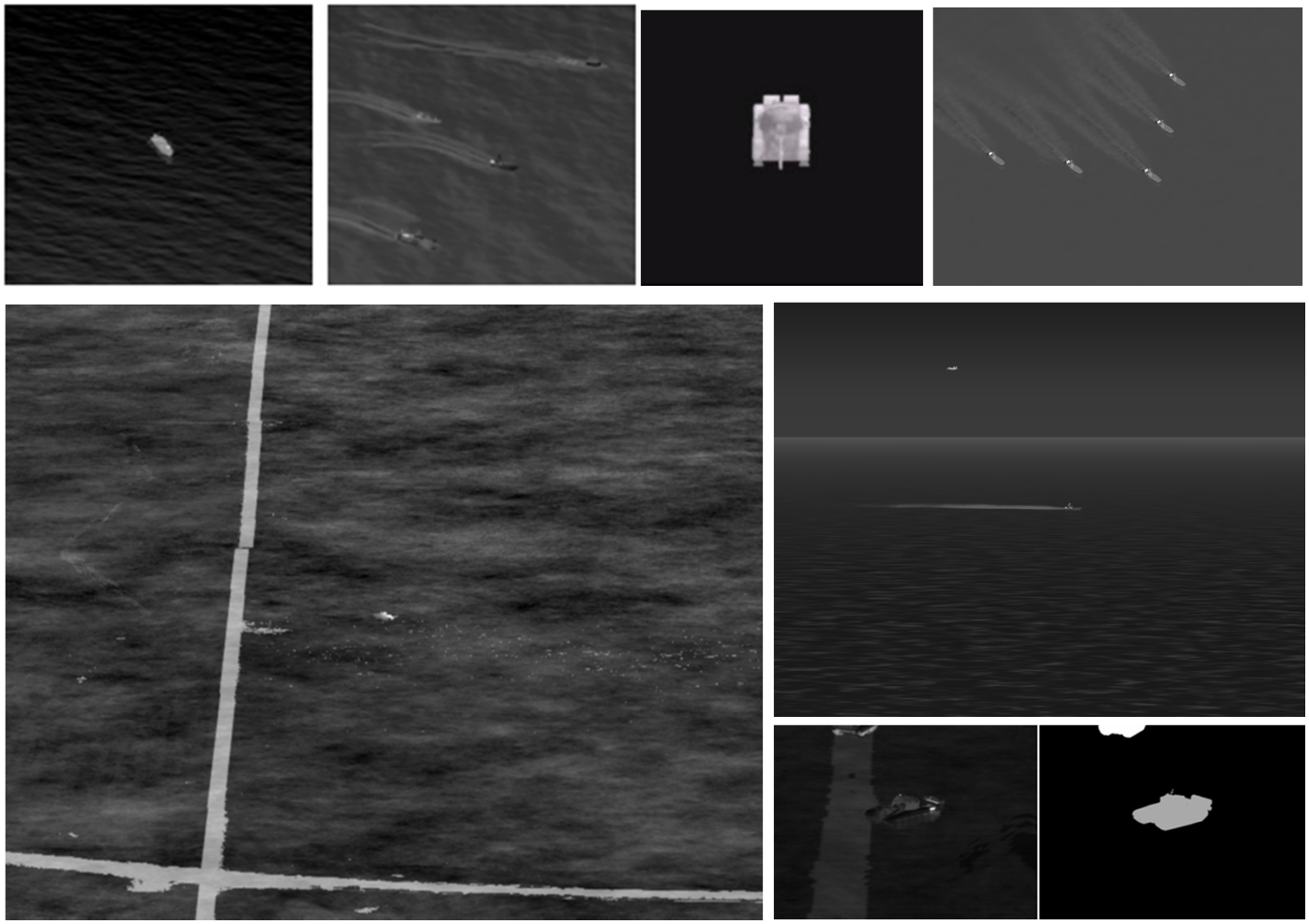


Signature modeling of targets and background is computationally intensive and is a pre-processing step to creating the scenes

Scene generation provides a virtual world based on modeled phenomena

Raytheon

Sample Scenes



Scene Generation

Scene generators create a radiance map using IR phenomenology and behavioral models

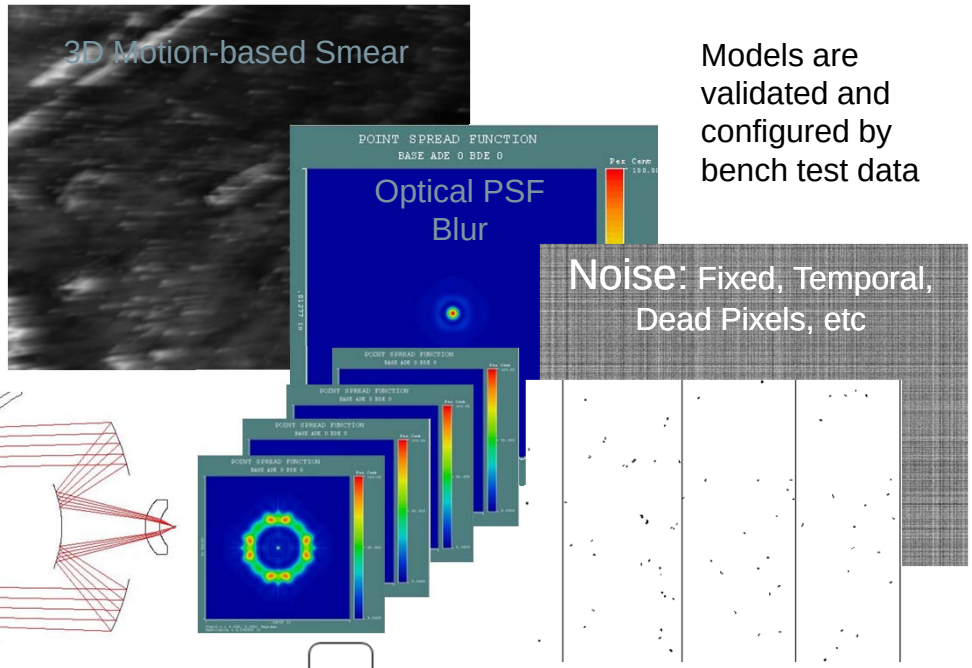
Raytheon

Outline

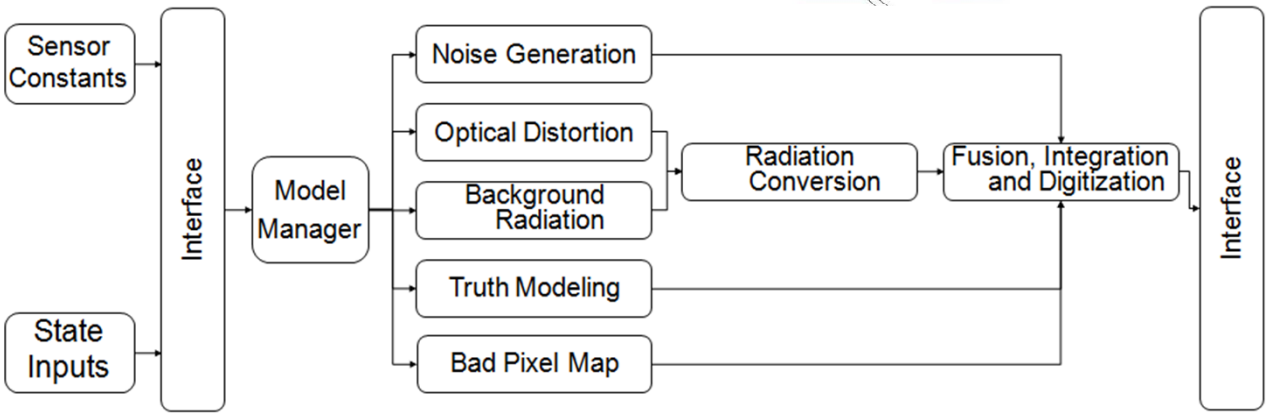
- Overview
  - Real time problem and challenges
- Motivation
  - Current state-of-the-art and our solution
- Image Chain
  - Missile image chain
- Scene Generation
  - How it fits into the closed loop pipeline
- EO Sensor Model
  - Summary, physics and challenges
- New Paradigm
  - CPU vs. GPU
- Real time pipeline, GPGPUs and CUDA
- Some examples
- Summary

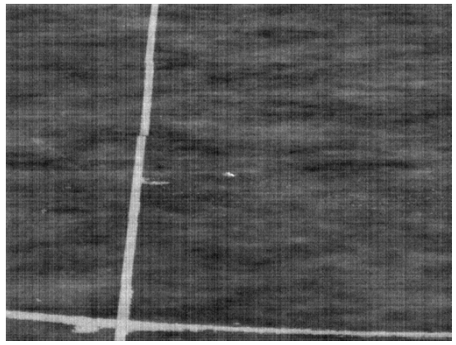
EO Sensor Model: Defined

- High Fidelity EO IIR Hardware Model (**genIR™**)
  - Smear
  - Optical Blur
  - Electronic Noise
  - Bad Pixels
  - ADC: Radiance to Counts
  - Others



Models are validated and configured by bench test data





The EO sensor model degrades the image by applying optical and detector effects

Raytheon

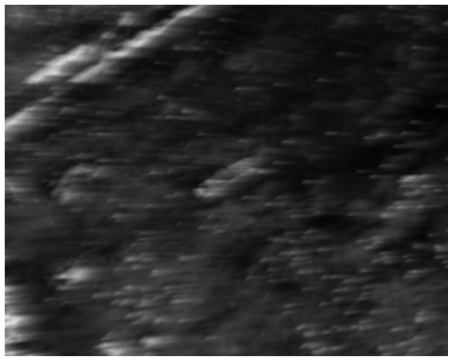
genIR™ Models

- System level models that efficiently apply physics-based effects without going into extensive mathematical effort
  - Better than real time performance is necessary to model high fidelity sensor effects at real time frame rates
- Sub-models currently used
  - SMEAR (motion blur)
    - Using platform and gimbal motion determine the line-of-sight rates of all of the pixels in the two dimensional focal plane array (FPA)
  - Point Spread Function (PSF)
    - Applied as a spatial filter either directly or in frequency domain
  - Temporal and Fixed Pattern Noise
    - Uses statistical characteristics of measured data from FPAs
  - Bad Pixel Map (BPM)
    - Uses statistical distribution of factory dead pixels of typical measured sensor
    - New ones are created per instantiation
  - Bad Pixel Replacement (BPR)
    - Uses algorithms to correct for factory dead or dynamically unusable pixels

The EO sensor is comprised of multiple sub-models each affecting a part of the image chain

Raytheon

SMEAR Model



- Motion blur (SMEAR) model
  - Implemented with knowledge of FPA pixel motion
    - Requires knowledge of missile motion
  - Applied at the local (per pixel) level (not globally) on the image
  - The effects on the image are integrated over the sensor integration time
  - Can *ONLY* be done in the spatial domain
- Implementation
  - Operate on original image for every integration step by moving the image
  - Requires knowledge of row/column locations for every sub-image
  - Need to recreate the propagation matrix for every integration step

Smear is fundamentally a spatial effect which is involved to implement

Raytheon

SMEAR: Derivation of Image Motion

Moving point D relative to stationary missile in missile coordinates

$$\vec{r}_{DB}^{\,P} = \frac{\left(\vec{r}_{DB}^{\,E}\right)^t \vec{u}_B^{\,E}}{\left(\vec{r}_{CB}^{\,P}\right)^t \vec{u}_B^{\,P}} \vec{r}_{CB}^{\,P} = \frac{-a}{\left(\vec{r}_{CB}^{\,P}\right)^t \vec{u}_B^{\,P}} \vec{r}_{CB}^{\,P}$$

$a \equiv$ Altitude of missile body

$\vec{u}_B^{\,P} \equiv$ Unit Ground UP vector

Transform from missile to platform coordinates

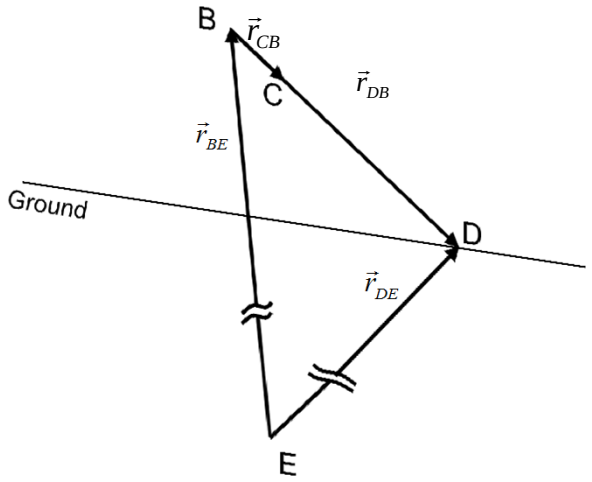
Moving point D relative to stationary platform coordinates

$$\dot{\vec{r}}_{DB}^{\,P} = -\vec{\omega}_{PE}^{\,P} \times \vec{r}_{DB}^{\,P} - \dot{\vec{r}}_{BE}^{\,P}$$

Stationary point D relative to moving platform coordinates

$$\dot{\vec{r}}_{DB}^{\,P} = \vec{\omega}_{PE}^{\,P} \times \vec{r}_{DB}^{\,P} + \dot{\vec{r}}_{BE}^{\,P}$$

Need pixel motion relative to stationary target position D!!



Trucco, E. and Verri, A., *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, (March 1998)

Smear requires direct knowledge of pixel motion with respect to the scene

SMEAR: Derivation of Image Motion II

Stationary point D relative to moving platform

$$\dot{\vec{r}}^{P}_{DB} = \vec{\omega}^{p}_{PE} \times \vec{r}^{P}_{DB} + \dot{\vec{r}}^{P}_{BE}$$

Projection onto image plane

$$\vec{r}^{I} = \begin{bmatrix} y^{I} \\ z^{I} \end{bmatrix} = \frac{1}{x^{P}_{DB}} \begin{bmatrix} y^{P}_{DB} \\ z^{P}_{DB} \end{bmatrix}$$

Velocity of the projected points

$$\dot{\vec{r}}^{I} = \frac{1}{x^{P}_{DB}} \left(\begin{bmatrix} \dot{y}^{P}_{DB} \\ \dot{z}^{P}_{DB} \end{bmatrix} - \dot{x}^{P}_{DB} \vec{r}^{I} \right)$$

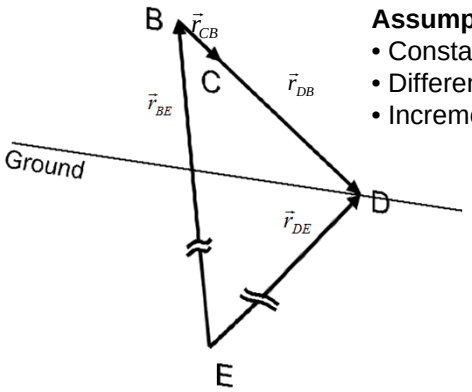
(Angular) Image rates

$$\dot{y}^{I} = \alpha_{00} + \alpha_{10} y^{I} + \alpha_{01} z^{I} + \alpha_{20} \left(y^{I} \right)^{2} + \alpha_{11} y^{I} z^{I}$$

$$\dot{z}^{I} = \beta_{00} + \beta_{10} y^{I} + \beta_{01} z^{I} + \beta_{11} y^{I} z^{I} + \beta_{02} \left(z^{I} \right)^{2}$$

$\vec{\omega}^{P}_{PE} = \left[\dot{\phi}^{P}_{PE} \quad \dot{\theta}^{P}_{PE} \quad \dot{\psi}^{P}_{PE} \right]^{t}$	Inertial angular rate vector (roll, pitch, yaw)
$\vec{u}^{P}_{B} = \left[x^{P}_{U} \quad y^{P}_{U} \quad z^{P}_{U} \right]^{t}$	Unit ground UP vector
$\dot{\vec{r}}^{P}_{BE} = \left[\dot{x}^{P}_{BE} \quad \dot{y}^{P}_{BE} \quad \dot{z}^{P}_{BE} \right]^{t}$	Missile rate vector
a	Altitude of missile body
α_{ij} and β_{ij}	Coefficients in terms of previous parameters

IMPT: All in platform (camera) coordinates



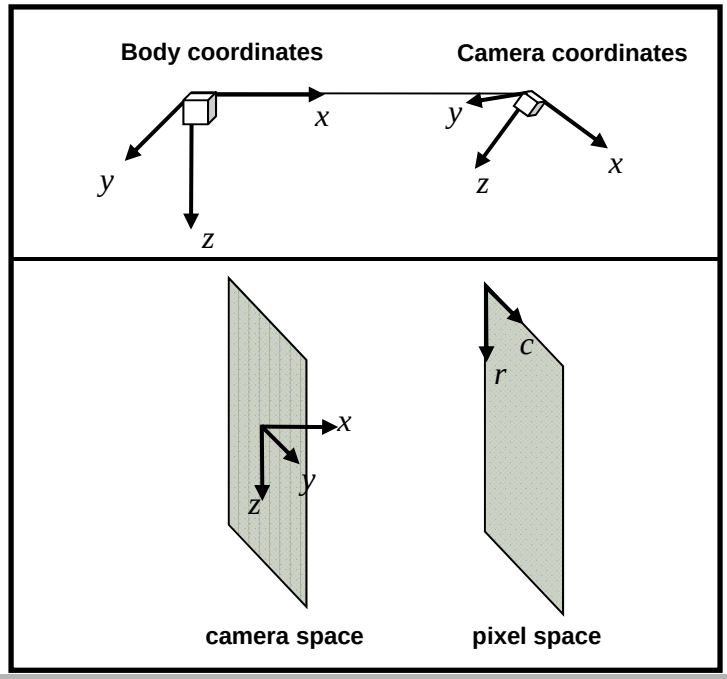
Assumptions
- Constant rate change for each pixel
- Different linear velocities from pixel-to-pixel
- Incremental change at sub-pixel resolution

B – Body

E – Earth

C – Camera (P - platform)

D – Stationary point



Smear is highly dependent of ancillary data from the platform and gimbal sub-systems

Raytheon

Point Spread Function: Definition

- The point spread function (PSF) is the two dimensional, incoherent, spatial impulse response of an optical system for direct detection
  - For a known an aperture stop, $p(x,y)$, and its geometric image,$(1/m)p(x/m,y/m)$, the exit pupil, we define

$$h(x,y) = m \cdot \Im\Im\left\{\left(\frac{1}{\lambda f}\right)^2 p\left(\frac{x}{\lambda f}, \frac{y}{\lambda f}\right)\right\} = m \cdot \Im\Im\{\hat{p}(x,y)\} \quad coherent \quad impulse \quad response$$

$$\gamma_H(\xi,\eta) = H(\xi,\eta) \otimes\otimes H^*(\xi,\eta), \qquad H(\xi,\eta) = m \cdot p(-\lambda f \xi, -\lambda f \eta) \qquad coherent \quad transfer \quad function$$

$$\mathcal{h}(x,y) = \frac{|h(x,y)|^2}{\gamma_H(0,0)}, \quad point \quad spread \quad function$$

- The optical transfer function (OTF) is the equivalent in the frequency domain
  - Magnitude is the modulation transfer function (MTF)

$$\mathcal{H}(\xi,\eta) = \Im\Im\{\mathcal{h}(x,y)\} = \Im\Im\left\{\frac{|h(x,y)|^2}{\gamma_H(0,0)}\right\}, \qquad \mathcal{H}(\xi,\eta) = \frac{\gamma_H(\xi,\eta)}{\gamma_H(0,0)} = \frac{\left[\hat{p}(x,y) \otimes\otimes \hat{p}^*(x,y)\right]_{x=-\lambda f\xi, y=-\lambda f\eta}}{area \quad of \quad aperture \quad stop}$$

  - MTF is the normalized autocorrelation of the exit pupil

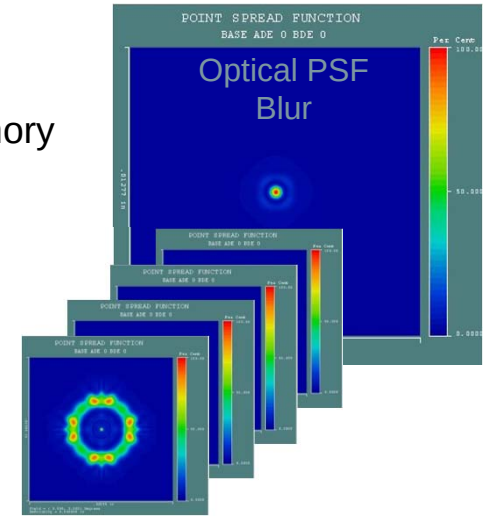
Gaskill, J.D, *Linear Systems, Fourier Transforms and Optics*, Wiley Interscience, (June 1978)

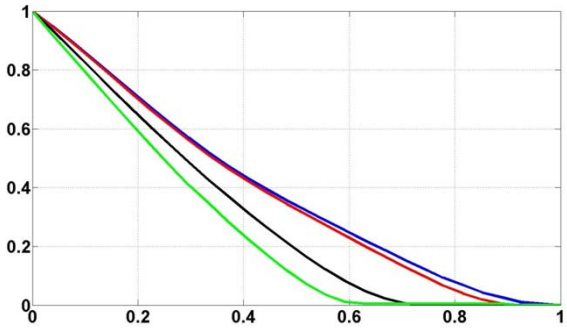
The point spread function represents the optic blur as a spatial filter

Raytheon

Point Spread Function: Implementation

- Via convolution (spatial filter) or Fourier space (FFTW/CUFFT)
- Point Spread Function (PSF) model
  - PSF data (lens design software)
    - As a function of range/off boresight angle/FOV imported into memory
  - Analytical PSF
    - Estimating deviations from diffraction limited PSF
      - Obscurations, defocus and other aberrations
- PSF via Modulation Transfer Function (MTF)
  - MTF data (lens design software)
    - Assume MTF separable and determine PSF (approximation)
  - Determine MTF from PSF
    - Use directly as two dimensional frequency filter
- Additional PSF contributions
  - Jitter, detector, etc…
    - Can be directly inserted into the model





Boreman, G.D., *Modulation Transfer Function in Optical and Electro-Optical Systems*, SPIE Tutorial, TT52, (July 2001)

PSF models can be implemented either in the spatial or frequency domain

Raytheon

3D Noise Model

- Developed by the US Army Night Vision and Electronic Sensor Directorate (NVESD)
  - Designed to provide a characterization of the noise found in many classes of electronic imaging sensors
- Each noise component is a zero mean Gaussian random process with a corresponding standard deviation (σ_{vh}, …etc.)
- These noise components are only orthogonal for infinite dimensions vertical, horizontal and temporal (*v*,*h*,*t*)
  - Non-orthogonality scales as $1/\sqrt{N}$
  - The total noise image is decomposed as follows:

$$U(t,v,h) = S + N_t(t) + N_v(v) + N_h(h) + N_{t,v}(t,v) + N_{t,h}(t,h) + N_{v,h}(v,h) + N_{t,v,h}(t,v,h)$$

  - We create the N's by using the following operators

$$D_t = \frac{1}{t}\sum_t U(t,v,h) \qquad D_h = \frac{1}{h}\sum_h U(t,v,h) \qquad D_v = \frac{1}{v}\sum_v U(t,v,h)$$

O'Shea, P. and Sousk, S., "Practical Issues with 3D-Noise Measurements and Application to Modern Infrared Sensor", SPIE Proceedings Vol. 5784, pp 262-271

The 3D noise model comprehensively describes the temporal/spatial noise in any EO/IR sensor system

Raytheon

3D Noise Model: Components

- These noise components are characterized by the standard deviations of the noise terms of the previous slide

Noise Term	Description	Potential Source
σ_{TVH}	Random spatial-temporal noise	Detector temporal noise (per pixel)
σ_{TV}	Temporal row noise	Line processing, 1/*f*, *readout*
σ_{TH}	Temporal column noise	Scanning
σ_{VH}	Fixed spatially random noise	Pixel processing, detector nonuniformity
σ_{V}	Fixed row noise	Detector nonuniformity, 1/*f*
σ_{H}	Fixed column noise	Detector nonuniformity, scanning
σ_{T}	Frame bounce	Frame processing
σ_{Total}	Total Noise	RSS of all Noises

Total Temporal Noise

Total Spatial Noise

3D Noise Model: Typical Noise Block

Procedure to extract standard deviation of components from measured data

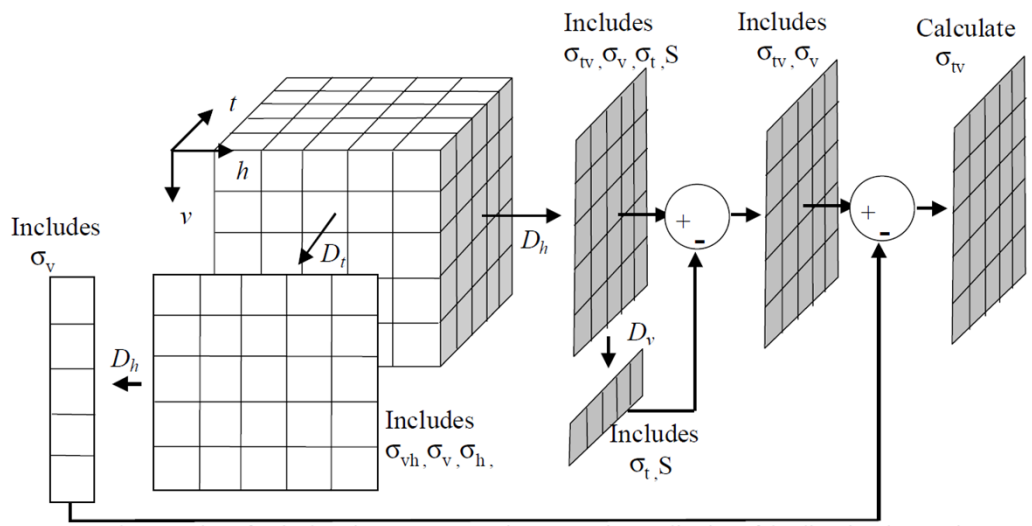
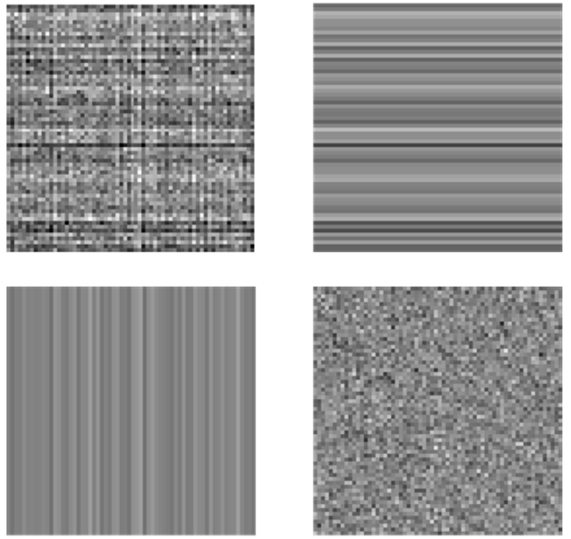


Figure 1. Sample extraction of a single noise component using successive application of the directional averaging operator.



3D Fixed Noise Term Representations:(Clockwise from upper left:) combined, σ_V, σ_{VH} and σ_h

$$D_t \cdot D_v \cdot D_h \cdot U(t,v,h) = \text{ImageMean} = S = 0$$

$$N_t = D_v \cdot D_h \cdot U(t,v,h) \qquad N_{vh} = D_t \cdot U(t,v,h) - N_v - N_h$$

$$N_v = D_h \cdot D_t \cdot U(t,v,h) \qquad N_{vt} = D_h \cdot U(t,v,h) - N_v - N_t$$

$$N_h = D_v \cdot D_t \cdot U(t,v,h) \qquad N_{th} = D_v \cdot U(t,v,h) - N_h - N_t$$

$$N_{tvh} = U(t,v,h) - D_t \cdot U(t,v,h) - D_v \cdot U(t,v,h) - D_h \cdot U(t,v,h) + N_v + N_h + N_t$$

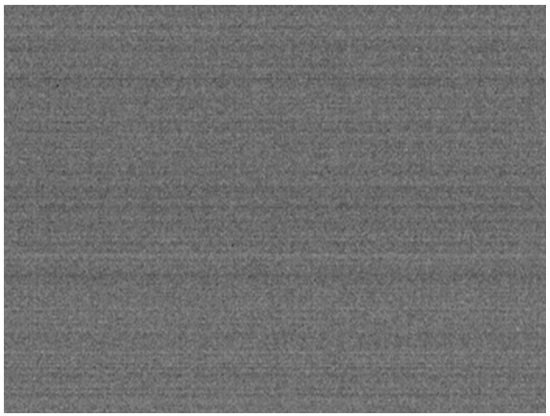
Measured data from a typical sensor represents the entire class of manufactured sensors

Raytheon

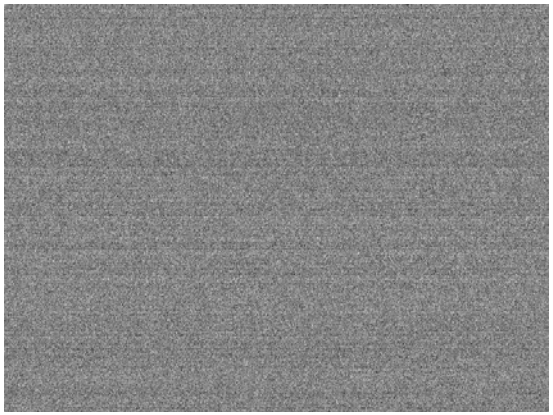
3D Noise Model: Typical Noise Frame

- The components described permit the simulation of realistic noise images
  - Some noise components are vertically or horizontally correlated
  - There is added spatially uncorrelated noise
- The 3D noise model accounts for these spatial components (fixed pattern) and their temporal variation

Test Image



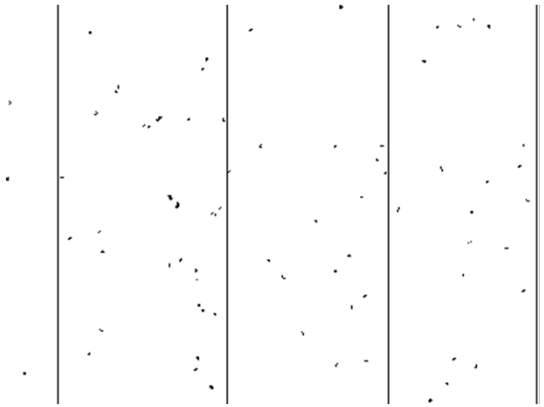
Simulation Image

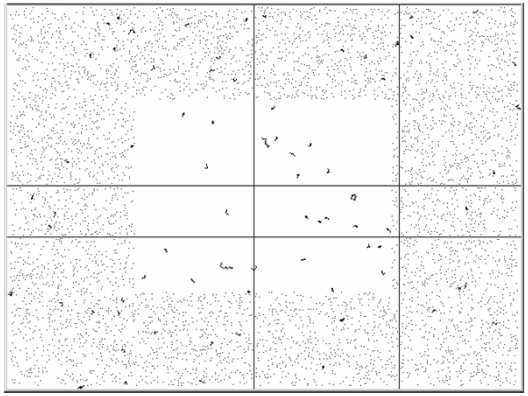


Raytheon

Bad Pixel Map

- Bad Pixel Map Creation
  - Based on statistics from measured sensor data
    - For the entire FPA or various regions of interest
  - Logged in a separate file or encoded within the image
  - If separated by region each can have different dead pixel densities
    - Typically you see fewer dead pixels in a region centered in the FPA and a larger distribution outside of the center

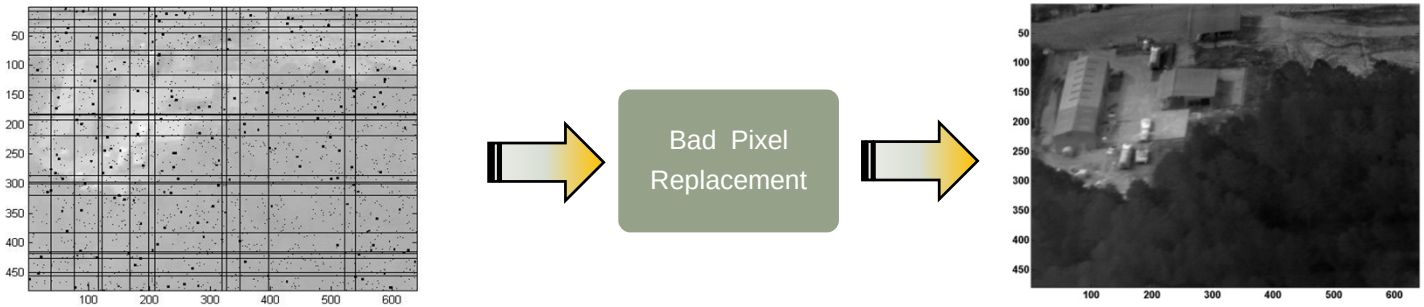




Bad Pixel Maps identify factory dead pixels

Bad Pixel Replacement

- Accomplished with a portion of non-uniformity correction (NUC) algorithm
  - With its the ability to adjust for gains and offsets, the NUC can ensure that we can estimate the correction for each dead pixel on a per frame basis
- Two Point NUC
  - Blackbody source at two temperatures create uniform backgrounds that can be used to determine adjustments to gains and offsets at both temperatures
  - Done apriori in the lab for each sensor
- Scene-based adaptive NUC
  - Using a uniform (bland) scene we can determine the adjustment to gains and offsets on a per frame basis



Bad Pixel Replacement corrects factory dead or dynamically unusable pixels in the FPA

Raytheon

Other Models

- Vignetting
  - Applied as a field-of-view dependent transmission loss
- Stray light
  - Applied as a time and position dependent external background source
- Internal background radiation
  - Radiometrically accurate and temporally varying internal self emission
    - Dome radiation
    - Optics and internal housing
- Time dependent noise effects
  - Electromagnetic interference

Various models can be included to make the EO sensor model more realistic and robust

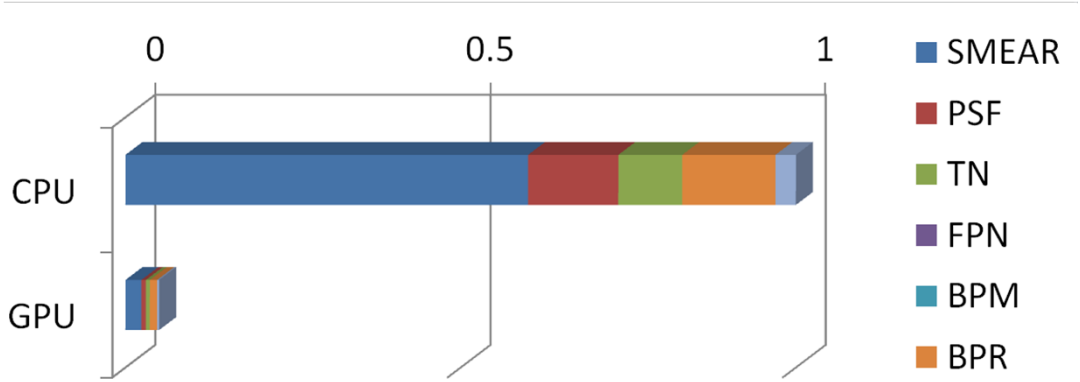
Raytheon

Outline

- Overview
  - Real time problem and challenges
- Motivation
  - Current state-of-the-art and our solution
- Image Chain
  - Missile image chain
- Scene Generation
  - How it fits into the closed loop pipeline
- EO Sensor Model
  - Summary, physics and challenges
- New Paradigm
  - CPU vs. GPU
- Real time pipeline, GPGPUs and CUDA
- Some examples
- Summary

The New Paradigm

- High fidelity scene generation and EO IR sensor modeling is computationally intensive
- Our efforts to create a real time requires a new way of thinking
- General purpose graphical processing units (GPGPUs) enable reductions in execution time and the potential to achieve real time



- We have succeeded in reducing the execution time of our combined scene generation and EO IR sensor model

Raytheon

Outline

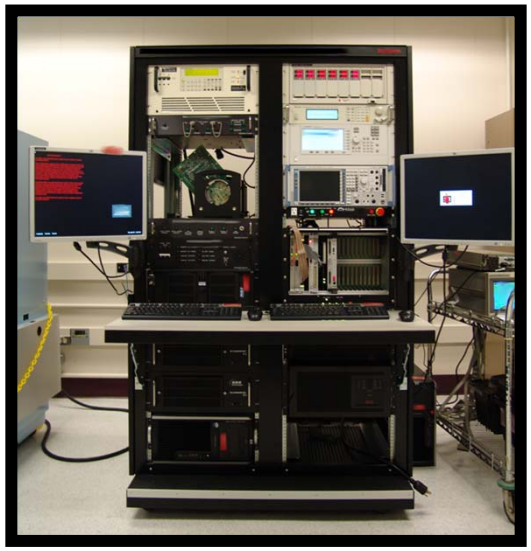
- Overview
  - Real time problem and challenges
- Motivation
  - Current state-of-the-art and our solution
- Image Chain
  - Missile image chain
- Scene Generation
  - How it fits into the closed loop pipeline
- EO Sensor Model
  - Summary, physics and challenges
- New Paradigm
  - CPU vs. GPU
- Real time pipeline, GPGPUs and CUDA
- Some examples
- Summary

Raytheon

Real Time Pipeline: CIL Application EO Imaging

- In a CIL, the flight computer executing the tactical image processing algorithms cannot wait for image generation to complete
- Scene Generator + EO sensor model, must be faster than real time. GPGPU acceleration gives us this capability

CIL test station



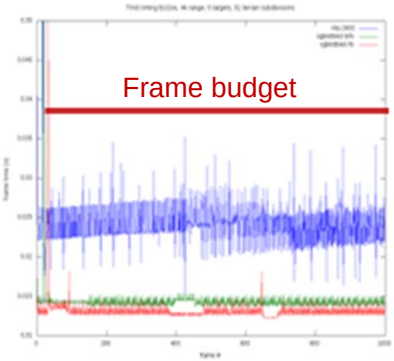
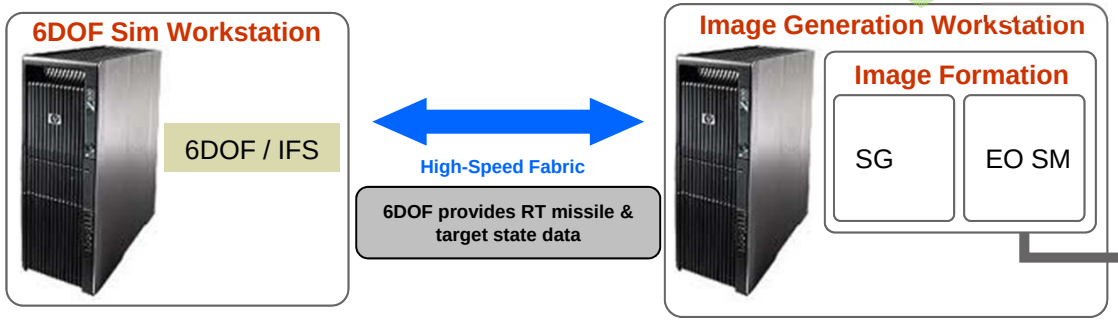


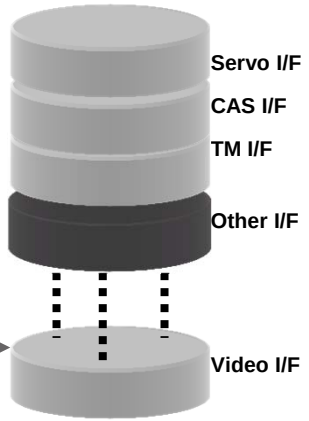
Image generation frame budget is a very tough requirement in a CIL simulation. GPGPUs working with real time LINUX systems help meet and beat this

Nvidia Tesla c2070



Electronics Unit (EU) Stack

The tactical algorithms are executed on the actual flight hardware.



GPGPU acceleration was the key to execute SG+SM at faster than real time rates for the CIL

Raytheon

Getting to GPGPU

- "GPGPU stands for *General-Purpose computation on Graphics Processing Units*. Graphics Processing Units (GPUs) are high-performance many-core processors that can be used to accelerate a wide range of applications." *from* http://gpgpu.org
- Why consider GPGPU in the first place?
  - **COTS:** you can go buy a graphics card at your local computer parts shop
  - **Community:** there is a large developer community (academia, oil exploration, seismic research, medical imaging, etc.) focused on data parallel programming
  - **Cost:** relatively inexpensive over the lifecycle when compared to FPGAs

Increase in cores, throughput, and programmability

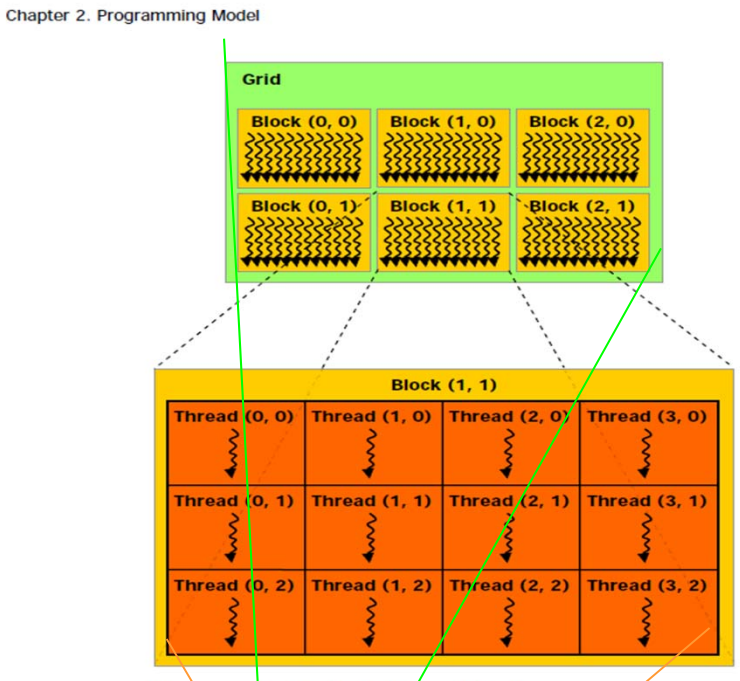
Cg, CUDA and GeForce are products of NVIDIA®

GPGPU technology and programmability has matured and is continuing to advance

Raytheon

GPGPU Using CUDA

- The Compute Unified Software Architecture (CUDA) from NVIDIA®
  - Provides an architecture and software Application Programming Interfaces (APIs) to exploit the massively parallel computation ability of Graphical Processing Units (GPUs) in commodity graphics cards
- The CUDA APIs offer its users GPU-based acceleration from a higher level of abstraction, without resorting to hand-optimized code
  - CUFFT: FFT library patterned after FFTW3
  - CUBLAS: Basic Linear Algebra functions
  - CUSP: Sparse matrix multiplication
  - CURAND: pseudorandom/quasirandom NG
- Avoids the common pitfalls of other approaches for multi-core development
  - Performing hardware-based thread management
- CUDA develops seamlessly with C/C++
  - Also supports Fortran, Python, Matlab



vectorAdd.cu

```
dim3 block(1, blockSize, 1);
dim3 grid(3, (int)(N/block.y) + 1, 1);

for(float time=0.0; time<runTime; time=time + dt){
  VecAdd<<<grid, block>>>(d_pos, d_vel, d_accel, N, dt);
  cutilSafeCall( cudaThreadSynchronize() );
}
```

NVIDIA's CUDA makes it possible to harness the processing throughput of GPUs

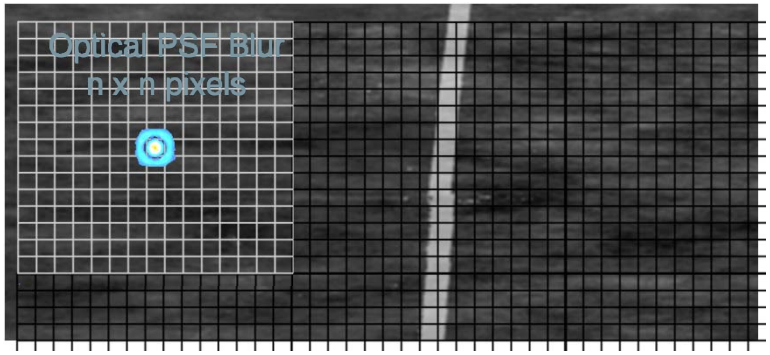
Raytheon

Outline

- Overview
  - Real time problem and challenges
- Motivation
  - Current state-of-the-art and our solution
- Image Chain
  - Missile image chain
- Scene Generation
  - How it fits into the closed loop pipeline
- EO Sensor Model
  - Summary, physics and challenges
- New Paradigm
  - CPU vs. GPU
- Real time pipeline, GPGPUs and CUDA
- Some examples
- Summary

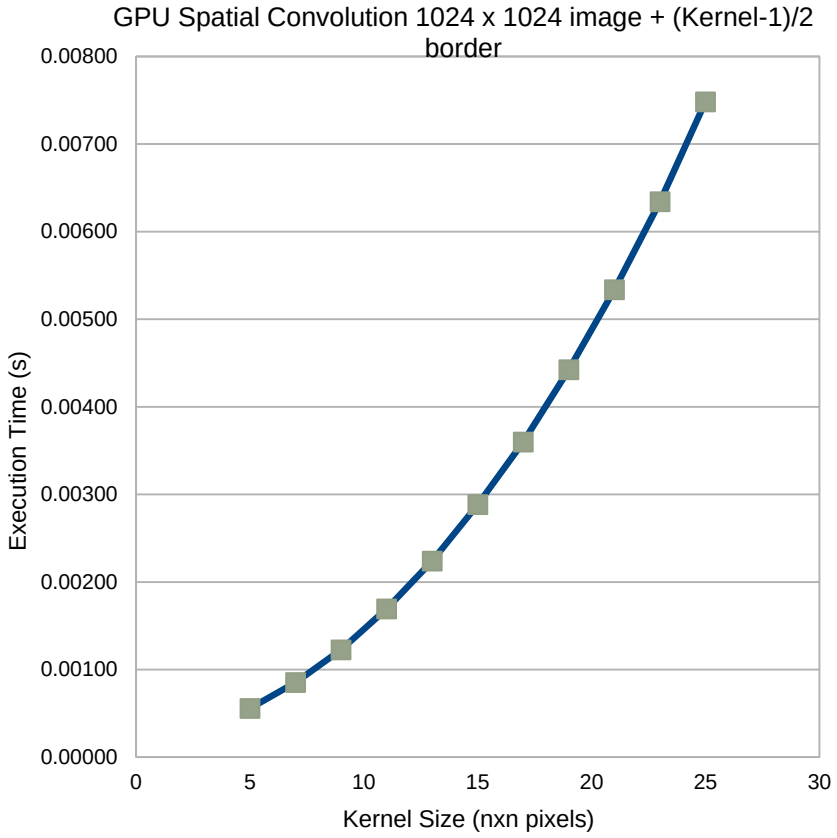
Raytheon

Spatial Convolution GPGPU Acceleration



1024x1024

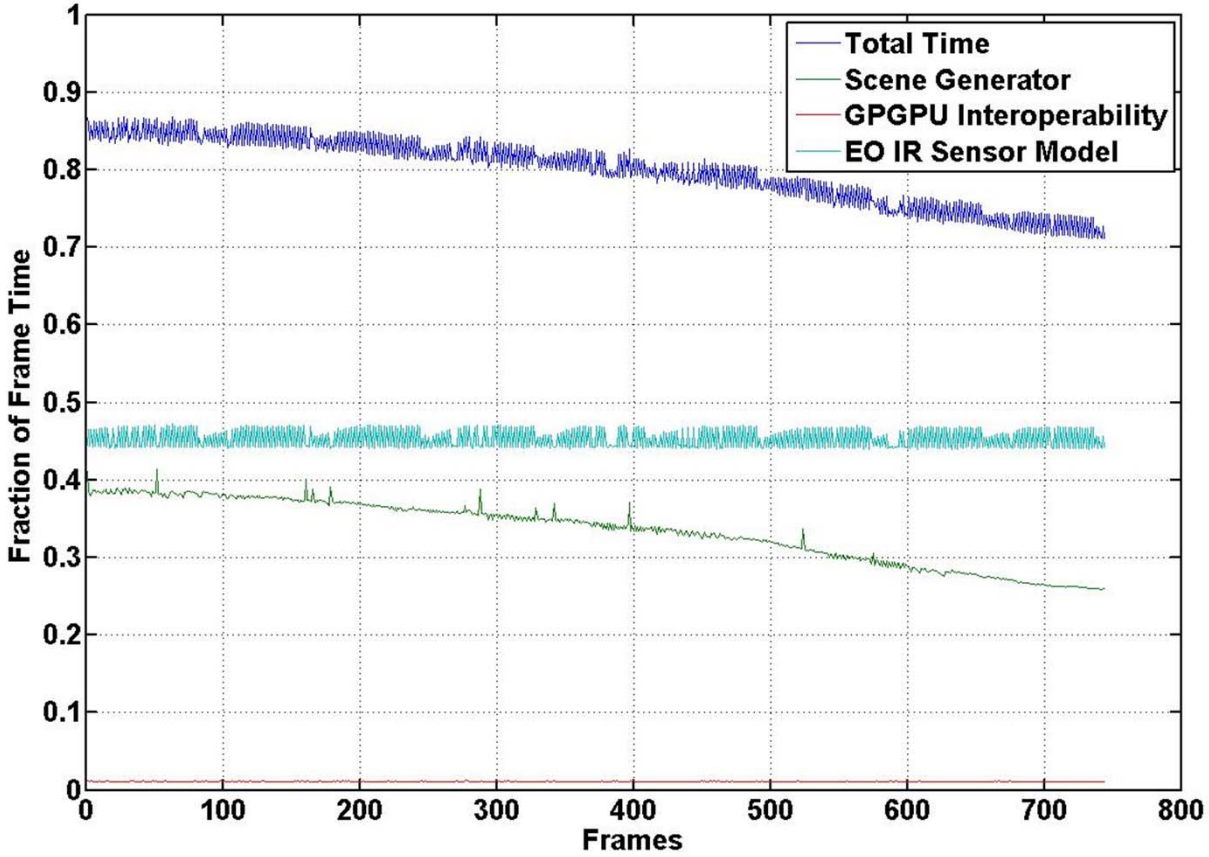
Kernel Size (nxn)	CPU Execution Time (s)	GPU Execution Time (s)	Speedup (x)
5	0.17	0.00056	311
7	0.31	0.00085	363
9	0.47	0.00123	387
11	0.70	0.00169	414
13	0.97	0.00224	432
15	1.73	0.00289	601
17	3.26	0.00360	905
19	4.78	0.00443	1080
21	9.07	0.00534	1699
23	9.92	0.00634	1564
25	10.80	0.00748	1444



CUDA allowed us to accelerate a model by almost 1700x

Raytheon

Scene Generator + EO Sensor Timing I

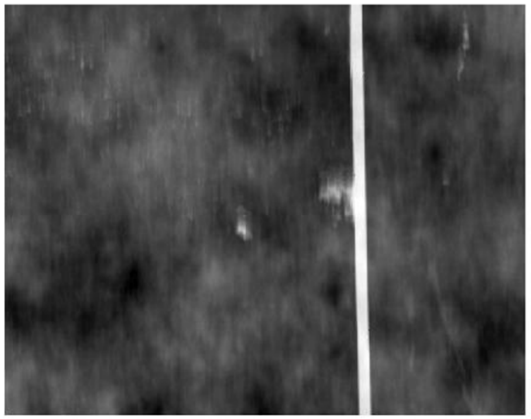


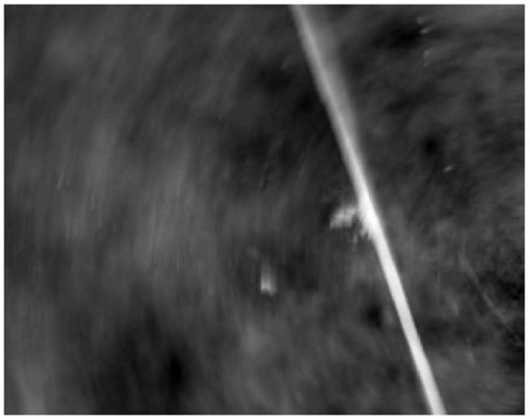
Timing analysis on a real time LINUX system

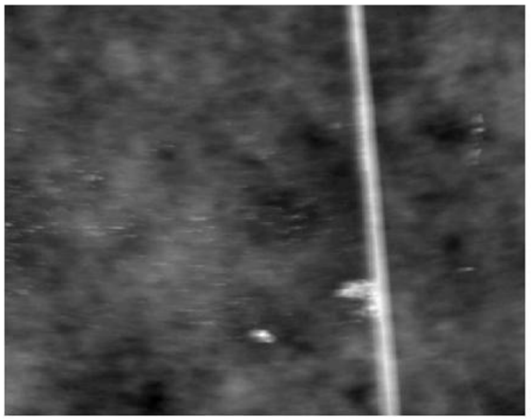
Raytheon

Examples

- IR scene generator and sensor model images







The more computationally intensive scenarios contain motion blur

Raytheon

Outline

- Overview
  - Real time problem and challenges
- Motivation
  - Current state-of-the-art and our solution
- Image Chain
  - Missile image chain
- Scene Generation
  - How it fits into the closed loop pipeline
- EO Sensor Model
  - Summary, physics and challenges
- New Paradigm
  - CPU vs. GPU
- Real time pipeline, GPGPUs and CUDA
- Some examples
- Summary

Raytheon

Summary

- A fully developed scene generator and high-fidelity EO Imaging IR sensor model were implemented with GPGPUs demonstrating real time performance
- Do not confuse with simple high-rate “Real Time Scene Generators”
  - Our solution innovates by implementing a hardware-specific, high-fidelity EO sensor model, along with a scene generator to run at faster than real time rates
- Our sensor model accelerated with CUDA retains and reproduces the results of the verified and validated high fidelity, all-digital sensor model

Raytheon

References

- Schott, J.R., *Remote Sensing: The Image Chain Approach*, Oxford University Press, 2nd Edition, (May 2007)
- Gaskill, J.D, *Linear Systems, Fourier Transforms and Optics*, Wiley Interscience, (June 1978)
- Boreman, G.D., *Modulation Transfer Function in Optical and Electro-Optical Systems*, SPIE Tutorial Texts, TT52, (July 2001)
- O'Shea, P. and Sousk, S., "Practical Issues with 3D-Noise Measurements and Application to Modern Infrared Sensor", SPIE Proceedings, **5784**, pp 262-271
- Trucco, E. and Verri, A., *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, (March 1998)
- CUDA: http://developer.nvidia.com/category/zone/cuda-zone
- GPGPU: http://gpgpu.org
- Cg, CUDA and GeForce are products of NVIDIA®