DEVELOPMENT OF AN OBJECT-ORIENTED INFRARED IMAGING SYSTEM SIMULATOR AND ITS APPLICATION TO MULTI-SPECTRAL INFRARED IMAGING

Thesis by

Christopher D. Springfield

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

1998

(Submitted November 11, 1997)

Acknowledgements

To say that I couldn't have done this alone would be a universal understatement. My experiences at Caltech have been extraordinary and I leave here with much more than I came with. At the center of my Caltech life was the McGill Group. If it were not for Dr. Tom McGill's leadership, inspiration, and love of lots of cool toys, much of what I accomplished here would not have been possible. I owe him a debt of gratitude for being patient and giving me the opportunities to expand both my scientific and my creative sides.

One of Tom's many skills is his gift of finding the right people for the right job. Dr. Gerry Picus was the right person in my case. With Gerry's help and guidance, I was able to wrangle this project in and define a clear direction for it to take. At the same time, he showed true interest in the other activities of my life and always seemed eager for a good discussion of how to solve the world's problems. I will always consider him a friend.

Throughout my years in the McGill group, the faces have changed, but the personality of the group remains the same. I have never worked with such a group of broad-minded people. Every single person seems able to pursue brilliant scientific endeavors without losing sight of the goal in life, which is to live. Among those I have had the honor to work with, I wish to thank Wesley Salzillo for helping me get the camera running and finishing those pesky experiments; David Ting for providing that always calm but excited outlook on life; Harold Levy for being the doer of things yet to be done; Rob Miles, Johannes Swenberg, Per-Olav Petterson and Ron Marquardt for expanding my moralistic, economic and political horizons; Mike Wang and Yixin Liu for being patient enough to wait until this thesis was finished to get their wedding videos; and Marcia Hudson who was always there to fix one administrative headache after another. To those still in the McGill group, I thank you for continuing the group's glorious traditions and bringing your own personalities to the mix. Long live the spirit that binds us.

Outside of my McGill group experience, I must thank those individuals who touched my life and pushed me to finish what seemed to be an impossible task. At the top of that list is Kent Bradford, roommate, friend, confidant. Were it not for Kent, I doubt I would be the physicist I am today. He pushed and pulled me through the early years and stood by as friend in the later ones. I am proud to call him one of the best friends of my life. Among the rest, I must thank David Santiago, Mike O'Neal, Richard and Carolyn Doherty, and Shirley Marnaeus. They inspired me to pursue my dreams and were always there to lend me a hand when I needed it. Finally, I must thank everyone at Digital Domain for giving me eight months of pure pleasure and letting me grab hold of the brass ring for awhile. Never in my wildest dreams would I think I would have the chance to work on a potential Academy Award winning film and earn my Ph.D. in the same year. How glorious life can be.

Finally, there are four people I must thank above all the rest. They are ones that made it happen. I must thank my parents, Joe and Alice Springfield. Were it not for their love, support and encouragement, I would have stopped long ago. I must also thank my late grandfather, Richard Smeltzer, who died long before I started this journey. His memory was my inspiration, my drive. Everything I have accomplished is my way of honoring such a wonderful person. Finally, I must thank my fiancée Diana Lavely. As my love, my life, and my soulmate, she made sure this journey came to a happy close. It is to her that I owe so much. I thank God that I have an entire lifetime to make it up to her.

Abstract

This thesis describes research efforts undertaken to investigate passive, multi-spectral infrared imaging. Although many applications of multi-spectral infrared imaging may exist, the high cost of developing viable multi-spectral technologies has limited research into, and subsequent exploitation of, these applications. Our efforts attempt to find a way to minimize the cost of this research while concurrently investigating one of the possible applications of multi-spectral infrared imaging using current infrared imaging technology. The outcome is an object-oriented, infrared imaging system simulator, called IRIMAGE, and a series of experiments and simulations that confirm the viability of gaseous pollution detection using passive, multi-spectral infrared imaging.

IRIMAGE is a flexible tool capable of applications research and basic infrared system design. This combination makes it a cost effective tool for researching the applications of multi-spectral IR imaging and the technological requirements they require. We present the physical and computational concepts that underly the simulation as well as certain computational advances made during IRIMAGE's development. A comprehensive discussion of the primary objects that make up IRIMAGE and how the simulation works is also provided. Since the reliability of a simulation depends on experimental verification of its output, we also present the results of this verification.

Besides verifying IRIMAGE, these experiments investigated detecting gaseous pollutants using passive, multi-spectral IR imaging. The thesis describes the imaging system we used and the theoretical background of these experiments. For each experiment, we describe the experimental setup and how IRIMAGE simulated the experiment. Finally, we compare the experimental and simulation results. Although these experiments verify IRIMAGE and demonstrate how gaseous pollutants can be detected using passive, multi-spectral IR imaging, further research is necessary and certain technological advances must be made before this application can be exploited. More information about IRIMAGE is available on the web at www.ssdp.caltech.edu.

Contents

List of Figures

List of Tables

Glossary of Acronyms

AFGL	Air Force Geophysics Lab
AVS	Advanced Visualization System
BLIP	Background Limited Performance
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CG	Computer Graphics
ECS	Element Coordinate System
FITS	Flexible Image Transport System
FLIR	Forward Looking Infrared imager
FDA	Flexable Data Array
FPA	Focal Plane Array
FTIR	Fourier Transform Infrared spectroscopy
FWHM	Full Width Half Maximum
GCS	Grid Coordinate System
GM	Geometric Model
GMD	Geometric Model Database
GUI	Graphical User Interface
IR	Infrared
NSF	Negative Space Filling
OIA	Object Index Array
OOP	Object Orient Programming
POCS	Primitive Object Coordinate System

QE	Quantum Efficiency
RTE	Relative Temperature Experiment
SOE	Space Occupancy Enumeration
UPO	Universal Primitive Object
VGO	Versatile Grid Object
VIO	Virtual Image Object
WCS	World Coordinate System

Glossary of Terms

Atmosphere Object: A simulation object responsible for atmospheric modeling in IRIMAGE.

Atmospheric Model: A set of parameters that define a specific set of atmospheric conditions.

Detector Object: A simulation object responsible for modeling the four detector types and the background fluctuation noise in IRIMAGE.

Element Coordinate System (ECS): A general coordinate system for a programming object (such as a VGO or UPO) which makes internal operations easier but can transform any location in itself to and from the WCS.

Flexible Data Array (FDA): Either a floating point or integer array that stores the grid point data in the VGO.

FPA Object: A simulation object responsible for modeling each focal plane array and driving the IRIMAGE simulation.

Geometric Model (GM): A three-dimensional representation of the surface of a geometric object such as a cone or sphere.

Grid Coordinate System (GCS): The coordinate system for a VGO.

Object Index Array (OIA): An integer array that associates every grid point in a VGO with the source of the data that it stores.

Optics Object: A simulation object responsible for modeling the optical system for each FPA in an IRIMAGE simulation.

Output Object: A simulation object responsible for outputing the image data for each FPA in an IRIMAGE simulation.

Program Object (PO): A programming class or structure that defines a complete and self-contained piece of programming code.

Scene Object: A simulation object responsible for modeling the background sources in IRIMAGE.

Simulation Object (SO): An abstract element of the simulation such as the Scene object or FPA object.

Universal Primitive Object (UPO): A programming object capable of modeling a bounded surface using a hybrid polygon-edge based surface model. Specifically designed for interpolating onto a VGO.

Versatile Grid Object (VGO): A program object that defines a multi-dimensional orthogonal data array which stores spatially dependent data (e.g., image and volumetric data).

Virtual Image Object (VIO): A program object that stores a set of image transformations which define the location, orientation and scale of the image on the Scene object's imaging VGO.

World Coordinate System (WCS): The coordinate system for the entire IRIMAGE simulation. Located at the front surface of the optical system. All other coordinate systems must be able to transform points to and from the WCS.

Chapter 1 Overview of Research

This thesis discusses the research related to the development of an infrared imaging system simulator and its application to investigations into passive, multi-spectral, infrared imaging. The potential of passive, multi-spectral imaging for environmental monitoring and other scientific and commercial applications has long been a matter for discussion and debate. The development of experimental systems to explore these multi-spectral applications has been hampered by the high cost of such systems and the potentially small market they may represent. Therefore, it was the intention of this project to address the following issues related to multi-spectral infrared imaging:

- What are the possible commercial applications of infrared imaging, specifically multi-spectral infrared imaging?
- Is it possible to use passive, multi-spectral, infrared imaging to detect the presence of gaseous pollutants?

To answer such specifically systems oriented questions it is necessary to inquire into two more general issues.

- Is it possible to develop a cost-effective method of defining a standard scene and use it to compare the performance of different infrared imaging systems?
- Can we apply the same method to judging the performance of a particular imaging system that looks at a variety of different scenes?

We investigated these issues by developing a flexible computer simulation of the infrared imaging process and by conducting a series of multi-spectral experiments. The results of this investigation are an attempt to provide answers and solutions to the questions raised.

As a result of this investigation, this thesis describes several contributions we have made to the infrared imaging community. One of these contributions is the development of a flexible computer simulation, called IRIMAGE, that simulates the entire imaging process from the background scene to the output from an imaging system. Because it simulates the entire imaging process, IRIMAGE is capable of comparing different imaging systems against a standard scene and comparing a specific imaging system against a variety of scenes. IRIMAGE can also serve as an application research tool, investigating different applications of single band and multi-spectral infrared imaging. Using IRIMAGE in this capacity leads to the other significant contribution of our research. In an effort to validate IRIMAGE and investigate passive, multi-spectral detection of gaseous pollutants, we conducted a set of experiments that imaged various plumes of CH_4, CO, and CO_2 against either a hot or cold background. These images were generated using an infrared camera fitted with a filter wheel containing filters to isolate the emission bands of these pollutants. Subsequently, we simulated the approximate conditions of these experiments using IRIMAGE. By comparing the images from the experiments with the simulated images, we are able to demonstrate the validity of IRIMAGE. More importantly, we prove that it is possible to isolate and detect these three gases independently using passive, multi-spectral infrared imaging. However, we also determined that certain technological and performance issues must be resolved before this is a viable method of pollution detection. The rest of this chapter details the motivation for this project, discusses the IRIMAGE simulation, and presents an overview of the multi-spectral experiments discussed above.

1.1 Motivation

With the discovery of the infrared region of the electro-magnetic spectrum by Sir William Herschel in 1800 and the subsequent discovery of the absorption and transmission band nature of infrared radiation by his son, Sir John Herschel, in 1840, a new era in non-visible, optical investigations began. Since its humble beginnings as a research tool, the generation, detection and imaging of infrared radiation has become a multi-million dollar industry that encompasses a vast array of military (night scopes, targeting systems, etc.) and commercial (communications, astronomical observations, motion detectors, etc.) applications. Although a broad array of commercial applications for single detectors abound, very few imaging applications exist. Furthermore, it is becoming increasingly difficult for the industry to bear the cost of supporting efforts to commercialize and improve infrared imaging technology during this current era of government downsizing. Therefore, the infrared imaging community needs to find ways to increase the commercial market for infrared imaging while, at the same time, reducing the costs of manufacturing and supporting current technology as well as developing new technologies.

We believe that many of the applications of infrared imaging will require or be enhanced by some form of multi-spectral capability. The advantages of multi-spectral infrared imaging are analogous to the advantages of color images over black and white images. While a black and white image provides only the intensity of light observed, a color image provides both chromatic and intensity information. The additional chromatic information adds to the total information known about an object being imaged. Similarly, a multi-spectral infrared image contains additional information which also improves our ability to discern the unique characteristics of an object that would otherwise not be obtained from a single-band infrared image. These characteristics include the temperature and emissivity of a surface or the transmission and emission bands of a gas. Active imaging measures the change in the incident radiation of a known source due to the absorption of a substance. For instance, the most common use of active multi-spectral infrared imaging is terrestrial monitoring [1, 2]. These images are captured by airbourne or space-bourne, high resolution, imaging spectrometers (32 to 224 bands) which collect the reflected sunlight (.4 μm- 2.45 μm) from the earth's surface. From these images, we are able to differentiate minerals in the earth's crust; determine ocean temperatures; discriminate between different types of vegetation; and monitor certain characteristics of the atmosphere. Passive multi-spectral detection assumes that the objects of our investigations are either the source of the infrared radiation or attenuate a background source whose own radiative characteristics are not known. Although most of the standard infrared imaging systems are passive imagers, none of these systems are capable of real-time multi-spectral imaging. However, we believe there are certain applications such as temperature measurement, environmental monitoring and medical diagnosis which could benefit from the development of passive, multi-spectral infrared imaging systems [3].

As mentioned above, one of the possible applications of passive multi-spectral IR imaging is environmental monitoring, specifically pollution detection. As concerns about global warming and the enforcement of government mandated air quality standards increase, there is an overwhelming need for remote sensing and imaging of both natural and man-made gaseous emissions. Current methods of remote sensing of gaseous pollutants involve active detection such as FTIR (Fourier Transform IR) spectroscopy [4], laser heterodyne radiometry [5] or longpath photometry [6]. The drawback to these active methods is that they require some type of IR source which may not be convenient or even possible in certain situations. In addition, none of these methods image the emissions, but rely on the pollutants to pass through the active region between the source and the detector. On the other hand, since most gaseous pollutants are emitted at an elevated temperature, it may be possible to passively detect the emission spectra of these gases. A passive, chemical vapor detection system involving a FLIR (Forward Looking IR imager) with several narrow-bandpass filters and a series of detection algorithms was proposed in 1991 by Althouse and Chang [7]. Although their application was military based, they surmised that such a system could also be used in environmental pollution detection. Our experiments and simulations of passive, multi-spectral pollution detection rely on similar filtering concepts.

One increasingly popular method used to design and test systems is simulating how that system will perform under various conditions. In this approach, the entire infrared imaging system is modeled on a computer. Then using a set of test scenes (background sources and atmospheric models), the viability of this system could be compared with other existing imaging systems or new system designs. In addition, different aspects of a system can be adjusted to find the best balance of functionality and performance. Such an approach has been very effective in the design of electrical devices, integrated circuits and optical systems using simulators like Spice3 [8], ANALOG and BEAM. By testing new designs prior to their fabrication, it is possible to minimize the time and cost of the "trial and error" phase of development. In addition, a simulator gives the system designer the freedom to investigate designs that might be too expensive or difficult to fabricate without evidence to support the design.

Besides aiding in the R&D of an infrared imaging system, the same simulation could be used to investigate new applications of current and future infrared imaging technology. Using the information generated by such a simulation, a systems designer could fine tune existing systems for current applications or determine the best technology for a new application. It would also be possible for a researcher to investigate the performance requirements of a proposed application. Ultimately, a computer simulation of the entire imaging process could be used to tailor a complete imaging system for any application. Therefore, a flexible simulation capable of both imaging system design and infrared application research would not only reduce costs but would help stimulate the commercial market with new applications of infrared imaging technology.

Although there are several simulations related to infrared imaging and detection, none of these simulators are comprehensive enough to satisfy the need for a complete infrared imaging simulation. Several scene simulations exist including GTVISIT by Cathcart and Sheffer [9] and DIRSIG by Schott, Raqueño and Salvaggio [10]. These models use first principle computations and three-dimensional models to simulate the infrared radiation emitted by a scene. Despite their sophisticated modeling of background scenes, these simulations fail to properly model a dynamic and heterogeneous atmosphere and use a very idealized imaging system. On the other hand, several detector simulations like Stanford's SUMerCad [11] and Dawn Technologies' SEMICAD DEVICE [12] model the response of individual detectors. These simulations do not attempt to model a background scene, but use a standard gray body source for analysis instead. Between these two extremes lie several simulations that do attempt to model both the imaging system and the background scene [13, 14, 15]. However, these simulations are designed for either a specific application or a specific imager type. Although these simulations model the atmosphere using the Air Force Geophysics Lab (AFGL) LOWTRAN/MODTRAN atmospheric code [16], they only take advantage of its layered nature making the atmospheric model static and purely one-dimensional. In one way or another, each of these simulations falls short of the goal of a design and applications research tool capable of modeling the entire infrared imaging process. They either do not incorporate all of the major parts (source, atmosphere, optics and imaging method) or are designed for a specific type of imaging system or application.

1.2 The IRIMAGE Simulation

IRIMAGE bridges the gap between the scene generation programs and the detector simulations. It provides both a tool for designing and comparing different imaging systems and an application research tool for investigating the various applications of infrared imaging. Using a simple ray tracing technique in conjunction with a well defined background scene, atmospheric model and a simple optical model, IRIMAGE computes the infrared radiation that is incident upon the surface of each detector in a focal plane array. Each detector uses its responsivity curve and a background fluctuations model to convert the incident radiation into a signal, noise and total voltage. Following the object-oriented programming paradigm, IRIMAGE organizes these operations into a series of self contained objects that represent the major elements of the infrared imaging process. Data is passed between these objects in such a manner that improvements to any one of the program objects does not effect the other elements of the simulation. The parameters that define each element of a simulation are stored in a set of parameter files which are generated by the Graphical User Interface (GUI). All of these pieces put together result in a flexible simulation package that can generate results now and can be improved in the future.

Figure 1.1 is a physical representation of IRIMAGE. As this figure points out,

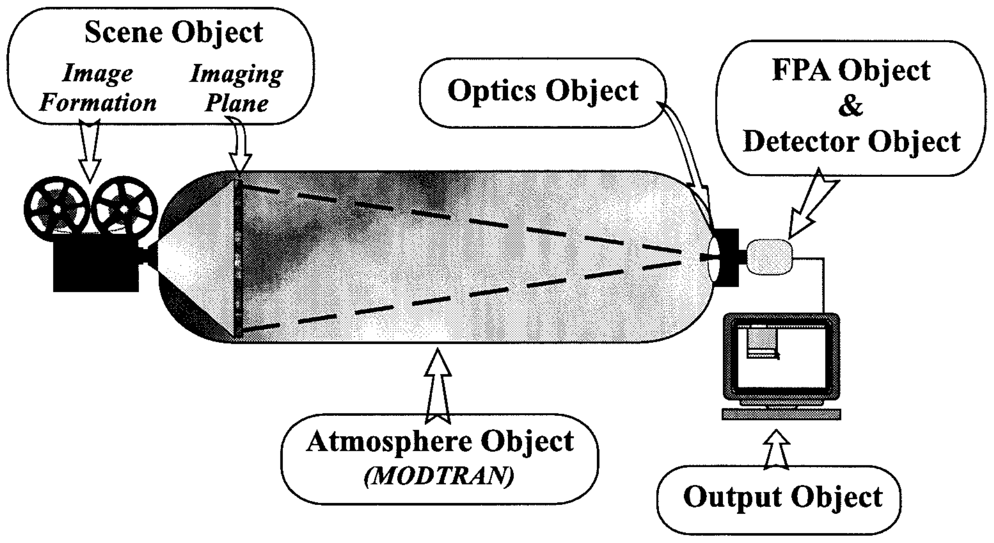


Figure 1.1: Physical representation of IRIMAGE simulator.

there are several major elements that make up the simulation. IRIMAGE attempts to treat each of these major elements as separate objects. By doing so, we are able to easily control and modify each of these elements independently.

The first element, or simulation object, of IRIMAGE defines the temperature and emissivity of background infrared sources. The *Scene simulation object* represents the non-atmospheric sources of the scene using a two-dimension background surface. This background surface is a two-dimensional grid called the imaging VGO (Versatile Grid Object). The imaging VGO allows us to reduce the source objects to a series of two-dimensional temperature and emissivity maps that may be interpolated onto the imaging VGO. Using a manipulation object called the Virtual Image Object (VIO), each of these maps can be translated, rotated and scaled in the 2D plane before being interpolated. Furthermore, these transformations can be animated using a series of key frames to store the manipulations for specific frames. Using the key frames, it is then possible to interpolate the transformation values between two key frames. In addition, multiple images can be overlaid on the imaging VGO using a simple "matte" to define what part of each image to interpolate. This two-dimensional approach is a first order approximation of a full three-dimensional scene simulation. Since such 3D simulations already exist, our intention was to provide a simple method of modeling a scene while being flexible enough to be able to incorporate an existing 3D scene generator in the future.

The other element responsible for modeling the scene is the *Atmosphere simulation object.* This element models the atmosphere between the Scene object and the imaging system. In this case, the atmosphere is modeled using a three-dimensional form of the VGO. The atmospheric VGO is used to define specific regions of the atmosphere that may have different characteristics such as a polluted plume or a hot gas cloud. Each of these polluted regions is represented by a bounding surface model, a gas model and an aerosol model. The parameters for each gas and aerosol model are stored in a set of databases. The bounding surface is linked to these models via their database index. Using an interpolation algorithm we developed, the Atmosphere object interpolates each bounding surface onto the VGO. By doing so, every grid point inside the bounding surface is then assigned the surface's corresponding gas and aerosol model index. Like the images in the Scene object, each of these bounding surfaces may also be animated using the key frame procedure defined above. Once all of these surfaces have been interpolated for a given frame, the Atmosphere object uses another interpolation alogrithm to generate the atmospheric profile along each ray that passes through the VGO. It combines this atmospheric profile with the temperature and emissivity values of the ray's interception point on the imaging VGO and passes this data to the MODTRAN object, a modified version of the atmospheric model developed by the Air Force Geophysics Lab. The MODTRAN object generates the spectral radiance array for the ray using the ray's background source information and atmospheric profile. Once the spectral radiance array is computed, it is passed onto the imaging system.

The IRIMAGE imaging system is made of four major elements: the *Focal Plane Array(FPA) object*, the *Optics object*, the *Detector object*, and the *Output object.* The

FPA simulation object acts as the overall control object for the imaging system. It lays out the detectors onto the focal plane by tiling a unit cell across the array. It also generates the rays used to compute the radiance incident on each detector and passes them onto the other major elements. Once the radiance is computed, the FPA object passes this information onto the Optics object and then the Detector object. Finally, the FPA retrieves the output voltages, photon power and photon arrival rates from the Detector object and passes them onto the Output object to be output into a file. IRIMAGE is also capable of defining multiple FPA's that look at the same scene. In some cases, this means that each FPA may need its own Optics object and Output object. Therefore, these objects are defined to be part of the FPA object.

The Optics simulation object defines the optical system of the imaging system. Currently, we use the pin-hole approximation in conjunction with the nodal points of the optical system. This approximation uses the concept that a ray passing through one of the nodal points of the system will appear to exit from the other nodal point with the same direction vector. Therefore, the Optics object stores the location of the cardinal points of the optical system (focal, principle and nodal points) and the location of the front and back surfaces along the optical axis. In addition, it stores the F-number ($F/\#$) of the system and maintains a database of transmission curves for different filters used in the optical system. When a ray is generated by the FPA, it is passed to the Optics object. It determines the point where the ray exits the optical system, adjusts the ray and returns it to the FPA object. Once the spectral radiance array is calculated for the ray, this array is passed back to the optical system which then uses the filter transmission curves to attenuate the incident radiance. After this attenuation is complete, the Optics object waits for the next ray.

The individual detector models are stored in the Detector simulation object. This object maintains a database of detectors which are based on a general detector model. Currently, we model four different detector types: Photoconductors, Photovoltaics, Pyrometers, and Bolometers. Although the physical processes involved with detection may be different, each of these detectors can be represented by a *spectral responsivity curve*. This curve defines what the resultant voltage is for an incident number of photons of a particular frequency. Since the radiance data is also given in the form of photons per frequency interval, this approach is an excellent method for generalizing each detector system as well as computing the output voltage that results from a given radiance array. This object also defines a model for generating the random noise associated with the background photon fluctuation. This time-dependent noise model approximates the random noise due to the random fluctuations in the incident number of photons for each frame. This noise model combined with the responsivity curve representation generates the signal, noise and output voltages for a particular detector.

Finally, the last element of IRIMAGE is the Output object. This object simply takes the output values from the Detector object and puts them in an ASCII data file. Each frame of the simulation is stored as a separate file. The object-oriented nature of this object will allow future versions of IRIMAGE to output the data in various image formats as well as onto the screen.

The parameters that define each of the major elements of the IRIMAGE simulation are stored in a set of parameter files. The Graphical User Interface (GUI) generates a parameter file for each of the six major objects of IRIMAGE. In addition to the six parameter files for these objects, the GUI also generates a general parameter file which sets up a few of the animation parameters, the monitoring flags, and links each major object with its own parameter file name and location. Each of these parameter files are written in ASCII format so they are portable to virtually any operating system that IRIMAGE supports. We avoid a direct link between the GUI and IRIMAGE because we prefered to be able to run our simulations in batch mode (multiple simulations at a time) which is difficult to implement if the GUI is linked directly to the simulation.

The actual simulation process in IRIMAGE is very straightforward. It begins by loading and initializing each of the major objects. Once each object is prepared, the imaging and atmosphere VGO's are set up for the current frame. With the scene and atmosphere prepared, the FPA generates each ray and passes it to the Optics object to compute the exit ray. This ray is then passed to the Scene and Atmosphere objects to generate the profile and then compute the spectral radiance array using MODTRAN. This spectral radiance array is passed back to the Optics object to be attenuated and is then passed on to the Detector object. Using the detector associated with the given detection element, the Detector object computes the signal, noise and total voltages. These values are passed onto the Output object which loads them into the ASCII file for the current frame. This process is repeated for each detector in the FPA and each frame of the simulation.

1.3 Multi-Spectral Infrared Imaging Experiments

The experimental phase of this project had two primary goals. The first goal was to investigate the viability of passive pollution detection using multi-spectral methods, one of the primary issues that we intended to investigate further with this project. The second goal was to verify the results of the IRIMAGE simulation using a set of simple experiments that test the various parts of IRIMAGE. In an effort to codify the verification process, we laid out three criteria that IRIMAGE should satisfy in order to be considered a valid simulator:

1. The Scene object accurately emulates a simple background scene that has regions with varying temperatures.

2. The Atmosphere object reasonably approximates the absorptive and radiative effects of a three-dimensional heterogeneous atmosphere.

3. The Imaging system (made up of the FPA, Optics, Detector and Output objects) generates a reasonable and realistic output based on the actual physical parameters of the experimental setup.

Using these criteria and the desire to investigate passive pollution detection, we designed three experiments that would satisfy both goals. Each of these experiments uses an Amber AE-256 infrared camera consisting of a 256x256 InSb array and a five position filter wheel in a LN_2 dewar. By using several different narrow band-pass filters in the filter wheel, we were able to incorporate multi-spectral imaging into these experiments.

The first of these experiments was the Relative Temperature Experiment (RTE). Designed to test the first and third validation criterion, the RTE consisted of a large aluminum plate and a smaller aluminum disk located in the center of this plate. The small disk was thermally isolated from the large plate using styrofoam and had a small heater attached to its backside. The surfaces of both plates were painted with a high emissivity black paint to maximize the generated infrared radiation. Using the heater, we controlled the temperature difference between the large plate (at room temperature) and the heated disk. We used a thermocouple attached to both plates to measure their temperature difference and used a standard surface thermocouple to measure the absolute temperature of the large plate. Using several different filters, we captured images of various temperature differences. The inherent two-dimensional nature made this source ideal for testing the 2D source modeling in the Scene object. For the simulation, the measured temperature data is applied to a set of images that approximate the large plate and the small disk. Using these images and each filter's transmission curve, IRIMAGE generated a set of simulated images for each filter. The results of these simulations were images that were very close to the experiment's images, with some minor variance from the experiment. This variance may be attributed to temperature variance across the surface of the large plate (especially near the disk/plate interface) which could effect the relative and absolute temperature measurement. Since this behavior was not consistent between filters, we also suspect that the approximations made when digitizing the filter transmission curves may also be affecting the image, especially in the narrow bands. Overall, we felt IRIMAGE easily satisfied both the first and third criteria for validation.

Although the first experiment was exclusively for verfiying IRIMAGE, the second experiment had broader implications. Referred to as the "methane experiment," this experiment introduced a plume of highly concentrated methane (92.88%) and a trace amount of CO_2 (.22%) into a clean atmosphere against a fixed temperature background. The plume was generated by passing natural gas from the laboratory gas line through a copper coil placed in a water bath. By adjusting the water temperature, we controlled the temperature of the plume. Using the absorption/emission spectra of CH_4 and CO_2, we selected three filters to isolate the band-pass of the camera to the methane band, the carbon dioxide band and a clean band (not containing either gas). For each filter we examined the four possible cases of hot and cold backgrounds versus hot and cold plumes. We obtained the gas concentration data from the gas company and recorded the temperatures of the plume and clean atmosphere as well as the background surface for each case and filter. From this data and the camera setup data, we conducted a series of simulations of these experiments as well. The most important result was the ability to passively detect methane and CO_2 in their respective spectral bands. In addition, we did not detect any gas in the clean region, which implies that the plume was not part of the background plate and that the CH_4 and CO_2 bands were not overlapping. The simulation results were as encouraging. The simulated images compared very well to the experiment images. The major variances occurred when the cone that approximated the plume grew wider but still maintained the same concentration (not physically correct). This caused this simulated region to appear brighter when it was imaged against a cold background and vice versa against a hot background. However, near the tip of the exhaust nozzle, the images looked very similar. In addition, the background sources and noise modeling appeared to be much more consistant than in the RTE images. The excellent corroboration between experiment and simulation proves that IRIMAGE satifies the second and third criteria for validity. Furthermore, the experimental evidence and simulated images not only demonstrate the viability of detecting a highly concentrated gas, but also the ability to image a very minor consituent of that gas as well.

The third experiment proved to be the most difficult. In this experiment, we chose to examine a real world application of pollution detection using multi-spectral methods. Using a setup similar to the methane experiment, this experiment generated plumes nitrogen containing 10% CO_2, 1% CO and a mixture of 6.7% CO_2/ 1% CO. The poisonous nature of carbon monoxide (CO) necessitated the use of a gas cell that would isolate the polluted atmosphere from the laboratory atmosphere. The constraints of the imaging system, in combination with the desire to minimize depth of field problems, forced us to build a large gas cell which required large windows that were transmissive in the infrared. Unfortunately, the only windows transmissive enough to allow the detection of the plumes were made of poly-ethylene based plastic wrap. Although very transmissive, the non-uniformity of the surface caused noticable differences in transmission across the surface of the window. Therefore, we incorporated an image processing technique to try to negate the effects of these windows. The resulting images showed marginal improvement, but introduced artifacts that seemed to wash out most of the expected detail. Again, we used three filters to isolate the band-pass of the camera to the CO_2 band, the CO band, and a clean band. However, in this experiment we decided to examine only the cases of a hot plume against a cold background and cold plume against a hot background. For each filter and case, we recorded the temperatures of the plume, the clean atmosphere and the background surface. We also recorded the concentration of the gases in the plume. Using this data and the camera data, we defined the simulations for these experiments.

Unlike the methane experiment, the experimental results of the gas cell experiments were mixed. For the hot background/cold plume case, we were able to detect the carbon dioxide in a still frame, but carbon monoxide was much fainter and could only be seen by viewing successive frames. For the other case, we were able to see the plumes for each gas, but again very faintly. We attribute these problems to the problems with the windows. In all cases, the clean band showed no trace of either gas. Comparing the simulations to the experimental images, the simulation predicted that the plumes would be more visible. The results of the methane experiment agree with these simulations as well since the concentration of CO_2 in that experiment was between 25 and 50 times smaller and that plume was still visible. In general, these experiments demonstrate the possibility of independently detecting CO and CO_2 using passive, multi-spectral IR imaging. Since the results of the simulations predict that both plumes should be easily detected, we believe further experiments should be conducted using a better experimental apperatus.

Besides the detailed conclusions about each experiment and its relationship to IRIMAGE, there are some general conclusions that can be made. First, using a system like the AE-256 has significant problems as a true multi-spectral imager. Since the camera has to be recalibrated for each filter, it is impossible to generate near-real time multi-spectral images using this system. It might be possible if each of the filters in the wheel had central wavelengths and bandwidths that correspond to approximately the same number of collected photons over the desired temperature range, but this would put severe limitations on these filters. Another solution might be to use a set of microfilters with each filter sitting over a single detector. Such an arrangement would allow the camera to be calibrated for all of the bands simulataneously as well as provide true real time multi-spectral imaging. However, if the scene has a lot of spatial variance, there will be problems where two adjacent pixels may be looking at two very different sources. Such a solution could be ineffective for pollution imaging. Therefore, the further technological advances and research are neccessary before passive, multi-spectral infrared imaging is economically feasible.

Another conclusion from these experiments is that although IRIMAGE generates reasonably accurate images, there is still room for improvement. The atmospheric modeling in the methane experiment demonstrates the fundemental problem of modeling an expanding gas in IRIMAGE. As the bounding surface grows, the concentration remains the same. Although this is physically impossible, it is possible to simulate a series of sucessive surfaces that do decrease in concentration as they get larger (our two cylinder gas for example). However, it would be better if IRIMAGE were to define a method of diluting the gas inside a volume as it grows. Such a solution would require certain modifications to be made to MODTRAN as well as IRIMAGE. Another place for improvement is detector noise modeling. Initially, we assumed that a detector would be operating under background-limited conditions (BLIP). This is a reasonable approximation when looking at a wide spectral band, but as the band is narrowed for multi-spectral imaging, the detector noise begins to dominate the system. This would especially be true at shorter wavelengths (less than 4μm) were the radiation of a near room temeprature source gets considerably smaller than for the longer wavelengths. In these experiments, the effects of the simulated noise model on the simulated images appeared to be much smaller than in the corresponding experimental images for the filters centered around $3.5\mu m$ and having very small bandwidths ($\approx .07\mu m$). In these cases, the experimental images looked considerably noisier than the simulated counterparts. Although these improvements would make IRIMAGE an even better simulation, the results of these experiments prove that IRIMAGE is already a valuable design and application research tool.

1.4 Outline of Thesis

The rest of this thesis discusses the concepts and experiments introduced in this chapter in greater detail. Chapter 2 provides a detailed discussion of the IRIMAGE simulation. This includes a comprehensive discussion of the physical basis and computational methods used by IRIMAGE. It also expands on the six major elements of the simualtion and how they fit together. Finally, it presents the results of two example simulations.

Chapter 3 presents a comprehensive discussion of the experiments used to investigate passive multi-spectral pollution detection and validate the simulation. The chapter includes a complete description of the Amber imaging system and how the IRIMAGE simulator modeled this system. There is a brief overview of multi-spectral pollution detection. The three following sections describe each experiment and make comparisons between the experimental and simulation results.

There is also a three part appendix. Part A provides a set of block diagrams that show how the different objects in IRIMAGE are related. Part B is a very detailed discussion of the computational methods employed to interpolate bounded volumes onto a versatile grid object. This discussion expands on concepts introduced in chapter 2. Part C is an example of the parameter files used by IRIMAGE to simulate the car example in chapter 2.

Finally, a web site exists to support the IRIMAGE software. The latest versions of the source code and the GUI are available at www.ssdp.caltech.edu.

Bibliography

[1] A.F.H. Goetz, "Imaging spectrometry for Earth Observations," *Episodes*, 15(1), March 1992, 7-14.

[2] For an excellent discussion of terrestrial monitoring using imaging spectroscopy: G. Vane and A.F.H. Goetz, "Terrestrial Imaging Spectroscopy," *Remote Sensing of Environment*, 24, 1988, 1-29.

[3] S. Hejazi, D.C. Wobschall, R.A. Spangler, and M. Anbar, "Scope and limitations of thermal imaging using multiwavelength infrared detection," *Optical Engineering*, 31(11), Nov. 1992, 2383-2393.

[4] H. Phan and J. Auth, "Measurements of chemical emissions using FTIR spectroscopy," *American Laboratory News*, Aug. 1993

[5] D. Courtois, D. Delahaigue, and C. Thiebeaux, "Detection of thermal emission from atmospheric gases by laser heterodyne radiometry," *Infrared Physics*, 34(1), 1993, 407-413.

[6] G.A. Bishop, J.R. Starkey, A. Ihlenfeldt, W.J. Williams, and D.H. Stedman, "IR longpath photometry, a remote sensing tool for automobile enissions", *Anal. Chem.*, 61(10), Oct. 1989, pp. 671A.

[7] M.L.G. Althouse and C.I. Chang, "Chemical vapor detection with a multispectral thermal imager," *Optical Engineering*, 30(11), Nov. 1991, pp. 1725-1733.

[8] Spice was developed by the Computer-Aided Design Group, Department of Electrical Engineering and Computer Sciences, University of California-Berkeley. For more information, contact them via the web at www-cad.eecs.berkeley.edu.

[9] A.D. Sheffer and J.M. Cathcart, "Computer generated IR imagery: a first principles approach," *Proc. SPIE*, 933, 1988, 199-206.

[10] J. R. Schott, R. Raqueño, and C. Salvaggio, "Incorporation of a time-dependent thermodynamic model and radiation propagation model into infrared three-dimensional synthetic image generation," *Optical Engineering*, 31(7), July 1992, 1505-1516.

[11] J.L. Meléndez and C.R.Helms, "Process Modeling and Simulation for $Hg_{1-x}Cd_xTe$. Part I:Status of Stanford University Mercury Cadmium Telluride Process Simulator", *Journal of Electronic Materials*, 24(5), May 1995, 565-572.

[12] For more information contact: Dawn Technologies Inc., 491 Macara Avenue, Sunnyvale, CA 94086.

[13] W.T. Kreiss, A. Tchoublneh, and W.A. Lanich, "Model for infrared sensor performance evaluation: applications and results," *Optical Engineering*, 30(11), Nov. 1991, 1797-1803.

[14] H.V. Kennedy, "Modeling second-generation thermal imaging systems," *Optical Engineering*, 30(11), Nov. 1991, 1771-1778.

[15] G.H. Kornfeld, "Digital simulation of precise sensor degredations including non-linearities and shift variance," *Proc. SPIE: IR Image Process. and Enhance.*, 781, 1987, 63-70.

[16] F.X. Kneizys, L.W. Abreu, G.P. Anderson, J.H. Chetwynd, E.P. Shettle, A. Berk, L.S. Bernstein, D.C. Robertson, P.K. Acharya, L.S. Rothman, J.E.A. Selby, W.O. Gallery, and S.A. Clough, "The MODTRAN 2/3 Report and LOWTRAN 7 Model," *prepared for PL/GPOS*, 1996.

Chapter 2 IRIMAGE: An Infrared Imaging System Simulator

2.1 Introduction

Since the beginning of the computer age, scientists and engineers have used computer-based simulations to investigate numerous theories, designs and ideas. Initially, simulations performed the computational drudgery of numerical analysis and other simple algebraic calculations. As time progressed and computers became more sophisticated, more advanced numerical methods could be employed to solve linear differential equations. With advent of Computer Graphics (CG), the simulation community grew exponentially. By adding the visual component to simulations, one could "view" a simulation's results and quickly make adjustments or even adjust the parameters while the simulation was still running. With the advent of cheaper and more powerful computers, the concept of Computer Aided Engineering (CAE) took hold. Using CAE, solutions to real world problems could be tested on the computer, saving the time and money that might be spent attempting to find the solution in other ways. Computer Aided-Design (CAD) evolved from the CAE model. Using a CAD system, scientists and engineers could completely design and conduct simple tests on various models or equipment. Not only did this save time and money, it allowed a greater sense of freedom. A scientist could explore various cases without spending months adjusting an experiment. An engineer could not only tailor a specific product to an application but also look for alternative solutions to a particular problem. With acceptance of the Object-Oriented approach to problem solving and programming, new CAD systems allowed users to pick and choose what elements to incorporate in their models and which ones to disregard. It made creating CAD systems easier and improved the ability to tailor a CAD system to a particular application without losing its flexibility and maintainability. Using this new approach, we have created the IRIMAGE Infrared System Simulator to provide an object oriented CAD model for designing, testing and experimenting with various IR imaging systems.

In the past, IR system simulation has been approached from two directions. The first was inspired by the desire to simulate IR backgrounds and scenes. Programs like those created by Schott, Raqueno and Salvaggio [1] or by Cathcart and Scheffer [2] were created to use 3-D scene information and generate images that would represent what an IR scene would look like under various conditions. Although these simulations use sophisticated computer graphics models and first principle calculations to model the final scene, the atmospheric modeling is not as sophisticated, usually using standard atmospheric models like those in LOWTRAN [3]. In addition, the imaging system is largely ignored for simplicity sake or is treated as an image processing task. While these approaches certainly simulate what a scene would look like to the perfect imager looking through a standard atmosphere, they do not adequately approach how one would actually generate such images using an imaging system. The second approach involves the simulation of various detection devices. Programs like SUMerCad [4] or SEMICAD DEVICE by Dawn Technologies [5] simulate device performance using the physical parameters of a device in combination with equations for various detector types and material systems. The results would be the expected performance of the detector. Although a very effective method for testing detector performance, these simulations do not test the detector in an imaging system or test it against a realistic background.

IRIMAGE bridges the gap between the scene generation programs and detector simulations. It generates a simple background scene and uses a backward ray tracing approach to determine the incident IR radiation on the optical system of the imaging system. Using the spectral character of the incident radiation, the simulation computes the output signal and noise of an FPA based on the response curve of each detector. Thus, IRIMAGE generates output images of the background scene and intervening atmosphere using an actual imaging system.

The IRIMAGE simulator uses a straightforward Object-Oriented Programming

(OOP) approach instead of the classical linear programming methodology. Under the OOP paradigm, a basic object contains both data structures to store information and routines which can manipulate this information. Some basic objects define standardized methods for storing and manipulating specific types of data, i.e., vectors, matrices, grids, etc., while others are more specialized to perform specific tasks. Larger objects are constructed from these smaller objects while incorporating new variables and routines as well. Eventually, the program becomes a collection of coupled, self-contained objects that pass information, or other objects, between each other in order to perform the various tasks of the simulation. In terms of IRIMAGE, the OOP approach allows us to treat the various physical elements of the infrared imaging process as "objects" which interact with each other to generate an output image.

In chapter 1, figure 1.1 presented a physical representation of IRIMAGE. As this figure points out, there are several major elements that make up the simulation. IRIMAGE tries to treat each of these major elements as a separate object. By doing so, we are able to easily control each of these elements independently.

The first two elements of the simulation generate the IR source information. The background *Scene Object* defines a 2D background surface for the scene. The surface represents objects in the background by projecting various temperature and emissivity maps onto an image plane represented by a 2D grid approach. The atmosphere between the Scene object and the imaging system is represented by the *Atmosphere Object* which models the 3D nature of the atmosphere using a three-dimensional grid and the MODTRAN atmospheric model. IRIMAGE uses a backward ray tracing method in conjunction with these grid-based objects to compute the spectral radiation incident upon the optics of imaging system.

The IRIMAGE imaging system is made of four major elements. The *Focal Plane Array (FPA) Object* acts as the overall control object for the imaging system. It lays out the detectors onto the focal plane; generates and passes on the rays used to compute the radiance for each detector; passes the incident radiance information to the optics and detector objects; and passes the output voltages to the output object. With this in mind, each FPA object is associated with an *Optics Object* which stores the parameters which define the optical system. The Optics object manipulates the rays generated by the FPA and attenuates the collected radiance generated by MODTRAN. The FPA passes the radiant power returned by the Optics object to specific detectors stored in the *Detector Object.* Using the radiant power and the responsivity of the corresponding detector(s), the Detector object generates the signal output as well as the noise output which uses a noise modeling algorithm. Once the FPA receives the output from the corresponding detector element, it passes the data to the *Output Object* to be written out to a data file.

The parameters that define an IRIMAGE simulation are loaded by each of these objects using a set of seven parameter files generated by a separate Graphical User Interface (GUI). When IRIMAGE begins, it loads the general parameter file which sets up a few of the animation parameters and the monitoring flags. In addition, it links each major object with is own parameter file which contains all of the information necessary to define the corresponding object. These parameter files are written in ASCII format so they are portable to virtually an operating system that IRIMAGE supports. We briefly discuss the GUI and these scripts at the end of the paper.

Besides the obvious conceptual advantage of creating a program like one builds a house, the other advantages to the OOP approach include code reuse and minimal code upkeep. Since many of the basic objects used by IRIMAGE perform standard tasks or define standard data structures and operations, their code can be reused over and over by the various larger objects in the simulation. Several examples of code reuse are discussed in the paper. More importantly, most of the objects in IRIMAGE are self contained. Thus, changes in one object do not require changing the entire program which minimizes the upkeep of the code and provides maximum flexibility in the future for upgrades. In a simulation like IRIMAGE, this benefit is essential to the long term usefulness of the program.

Before discussing the simulation itself, we briefly discuss the major physical principles utilized by IRIMAGE in section 2.2. In section 2.3, we introduce several computational constructs used by the various objects of IRIMAGE. With these fundamentals introduced, we go over each major object of IRIMAGE in some detail in section 2.4. In addition, this section contains a simple walk through how an image is generated by the simulation. Finally, in section 2.5 we discuss a a few test simulations using IRIMAGE and discuss the long-term usefulness of IRIMAGE as both a research and design tool.

2.2 Physical Basis of IRIMAGE

Several major physical relationships need to be reviewed before a clear discussion of IRIMAGE can take place. In any infrared simulation one must discuss blackbody radiation and the effects of the atmosphere. Our simulation employs a ray tracing method to determine atmospheric profiles and background surface temperatures and emissivities, thus a brief introduction to ray tracing is also presented. Since we simulate an imaging system, a clear discussion of the optical collection of radiation and its conversion to an output voltage from a detector scheme is necessary. Finally, we discuss the concept of simulating the background fluctuations of radiation that create the incident photon noise. These ideas form the physical basis for the IRIMAGE simulation. Most of these concepts have been discussed thoroughly by others; we only wish to point out the pertinent equations and ideas that directly relate to IRIMAGE.

2.2.1 Blackbody Radiation

Several principles govern the radiation of energy from an object. Kirchhoff's Law defines the two basic principles regarding the interaction of electro-magnetic radiation and objects:

1. All objects may only transmit (τ), reflect (ρ), and absorb (α) incident radiation and their three coefficients must sum to one: $\tau + \rho + \alpha = 1$

2. Good absorbers are also good emitters meaning that at thermal equilibrium all energy absorbed by an object must be radiated.

Using Kirchhoff's law, a *blackbody* is defined to be a perfect absorber over all wavelengths and angles (i.e., $\alpha = 1, \tau = \rho = 0$). Therefore, at thermal equilibrium, a blackbody must also be a perfect emitter. The Stefan-Boltzman relation further establishes that the total energy absorbed or emitted by a blackbody only depends on the temperature of the object. From these principles, the general nature of a black body is defined. However, none of these properties describe the spectral nature of a blackbody. Max Planck used the quantized energy states of light and the statistical nature of quantized energy states to predict the spectral dependence of blackbody radiation. Planck's Radiation Formula demonstrates how the *radiance*, or radiated power per unit area per solid angle, of a blackbody depends solely on the temperature of an object (T) and frequency (ν) of the light it radiates:

$$L_{BB}(\nu, T)d\Omega = \frac{h\nu^5}{c^3\left(e^{h\nu/kT} - 1\right)}d\Omega \tag{2.2.1}$$

From (2.2.1), one can derive Wien's Displacement law which describes the relationship between the temperature of a blackbody and the peak wavelength of the radiance curve:

$$\lambda_{max}T = 2897\mu m K \tag{2.2.2}$$

Using (2.2.2), we find that for a blackbody at room temperature ($T = 293K$) the maximum point of the radiance curve occurs at a wavelength near 10 μm. As the temperature increases this wavelength decreases. Since most objects on earth have temperatures in excess of 200 K, one can see that the peak wavelengths will remain well within the infrared (IR) region of the electro-magnetic spectrum ($\sim$ 1 μm $\rightarrow$ 200 μm). However, most objects are not true black bodies, but are more characteristically "gray bodies" which can be approximated using the blackbody radiation formula (2.2.1) and the object's emissivity (ε):

$$\varepsilon = \frac{\text{Radiance of Object}}{\text{Radiance of a Blackbody}} \cong \alpha \qquad (0 \leq \varepsilon \leq 1)$$

The emissivity is the percentage of blackbody radiation that a gray body will radiate at a particular spectral frequency and, according to the second principle of Kirchoff's Law, should be equal to the absorption coefficient. In some cases the emissivity may be a function of temperature as well as frequency. The general "gray body" radiance equation can be defined using the properties of emissivity and equation (2.2.1):

$$\begin{aligned} L_{GB}(\nu,T)d\Omega &= \varepsilon(\nu,T)L_{BB}(\nu,T)d\Omega \\ &= \varepsilon(\nu,T)\frac{h\nu^5}{c^3(e^{h\nu/kT}-1)}d\Omega \end{aligned} \tag{2.2.3}$$

In most cases, an infrared imager will be capable of seeing in one of the two main windows of the infrared regions (3-5 and 8-12 μm). Since most materials can be approximated by a constant emissivity over these common wavelength ranges, IRIMAGE currently treats the emissivity as a constant with respect to frequency when modeling objects.

2.2.2 Modeling the Atmosphere using MODTRAN

Various methods have been developed to model the atmosphere. Most of them are designed to simulate the atmosphere along the line of sight of a single detector. A popular model of this kind is the LOWTRAN model. In an effort to correct for the atmospheric effects in aircraft and satellite observations, the Air Force Geophysics Lab (AFGL) developed LOWTRAN to accurately compute the transmission and radiance characteristics of various atmospheric conditions using a band-model approach with layer pressure as the only parameter. It is capable of modeling several common gaseous pollutants as well as various aerosol backgrounds. LOWTRAN contains several pre-defined atmospheric models as well as a method for defining a user-defined atmosphere using multiple horizontal layers. Using LOWTRAN as a basis, the AFGL derived MODTRAN [6] as a more accurate model for the infrared and visible regions. MODTRAN improved the band model to include three parameters (pressure, temperature and line width) all of which utilize a database of values derived from the HITRAN database [7]. Both models use wave numbers (cm^{-1}) as the spectral unit with the spectral bandwidth resolution of LOWTRAN being 20 cm^{-1} and MODTRAN reducing it to 2 cm^{-1}. The smaller bandwidth makes MODTRAN a superior model in the infrared region of the spectrum. Since MODTRAN also retains all of the major features and reliability of LOWTRAN, it is a good model to use as the basis for simulating the atmosphere in the IRIMAGE environment.

As mentioned, the MODTRAN model is not only capable of computing the spectral transmission of a path in an atmosphere, but it also computes the spectral radiance of the gases along the path and the background surface. Using the transmission coefficient with the radiance calculations, MODTRAN can compute a total spectral radiance value along a path. Although we discuss how we use MODTRAN to compute the incident radiance upon the imaging system later, the following discusses how MODTRAN computes the transmission coefficient, the background radiance, the atmospheric radiance and the final radiance along a specific path looking through the atmosphere.

Atmospheric Transmission Coefficient (τ)

The computation of the atmospheric transmission coefficient uses the MODTRAN band model in conjunction with its detailed parameter database. We will not discuss the computation of the individual gas transmission coefficients because this is detailed quite well by Anderson et al. [8]. One important note is that each individual gas transmission coefficient for a specific layer depends upon the path length, l, through that layer. If we assume the transmission coefficient for each gas in an atmospheric layer is computed, then the total atmospheric transmission coefficient for the n^{th} layer is simply the product of the individual gas transmission coefficients for this layer:

$$\tau_n(\nu, l) = [\tau_{O_2}(\nu, l)\tau_{N_2}(\nu, l)\tau_{CO_2}(\nu, l)\dots]_n \tag{2.2.4}$$

Once τ_n is computed for each layer along the path, one can compute the total transmission coefficient for the path. Again, the transmission coefficient along the path is simply the product of all of the individual layer transmission coefficients along that path. For a path passing through N layers of an atmosphere, the total transmission coefficient is defined by the following equation:

$$\tau_{atm}(\nu) = \prod_{n=0}^{N} \tau_n(\nu) \tag{2.2.5}$$

Background Surface Radiance

MODTRAN uses a simple background surface model to calculate the spectral radiance of the surface, $L_{surf}(\nu)$. It assumes the background surface is a gray body, opaque ($\tau_{surf} = 0$) and is Lambertian in natue. Using (2.2.3) with temperature, T_{surf}, and albedo, ρ, for the surface, MODTRAN computes the spectral radiance of the background surface.

$$L_{surf}(\nu) = (1-\rho)L_{BB}(T_s,\nu) \text{ where } \varepsilon = 1-\rho \tag{2.2.6}$$

Atmospheric Radiance

MODTRAN assumes the gas radiance can also be computed using the Planck Radiation formula (2.2.1) in conjunction with the spectral transmittance of each layer. As part of the calculations to determine each layer's transmission coefficient, MODTRAN computes the layer's Curtis-Godson density weighted temperature, θ. This temperature is part of the Curtis-Godson approximation which treats each atmospheric layer as if it were homogenous in temperature and pressure [9, 10]. MODTRAN extends this approximation to the radiance calculation by using θ in the blackbody calculation. With all this in mind, MODTRAN uses the simple, spectral, radiative transfer function for a single layer [11]:

$$L_{atm}(\theta,\nu) = L_{BB}(\theta,\nu)\cdot(1-\tau(\nu)) \tag{2.2.7}$$

We extend the above case to multiple layers by computing each layer's individual radiance contribution, L_n, and then multiplying the radiance value by the product of all of the transmission coefficients for layers between current layer and the imaging system.

$$\begin{aligned} L_n(\nu) &= L_{BB}(\theta_n, \nu)(1 - \tau_n(\nu))\Upsilon_{n-1}(\nu) \\ &= L_{BB}(\theta_n, \nu)(\Upsilon_{n-1}(\nu) - \Upsilon_n(\nu)) \end{aligned} \tag{2.2.8}$$

where

$$\Upsilon_n(\nu) = \begin{cases} 1 & \text{if } n = 0, \\ \prod_{j=1}^{n} \tau_j(\nu) & \text{if } n > 0. \end{cases}$$

Then summing over each layer we can get the total contribution of the atmosphere.

$$L_{atm}(\nu) = \sum_{n=1}^{N} L_n = \sum_{n=1}^{N} L_{BB}(\theta_n, \nu)(\Upsilon_{n-1}(\nu) - \Upsilon_n(\nu)) \tag{2.2.9}$$

Total Radiance along the Path

Using the various radiance equations and the total transmission coefficient, MODTRAN computes the final spectral radiance for any chosen path to be:

$$L_{total}(\nu) = L_{atm}(\nu) + L_{surf}(\nu)\tau_{atm}(\nu) \tag{2.2.10}$$

Despite all of its functionality, MODTRAN does have some drawbacks. Although considerably faster than a line-by-line model, MODTRAN is still computationally intense. The calculations in conjunction with the necessary database access causes MODTRAN to take up to several seconds to compute a single path. There is no automatic procedure for defining multiple paths through the atmospheric model. Any imaging simulation must supplement MODTRAN with external routines which define individual paths and pass them to MODTRAN. In addition, MODTRAN can only define an atmospheric model in terms of horizontal layers of gas. More complex atmospheric models, i.e., blobs of polluted gas, require external routines to generate specific profiles for each path. These routines must interpolate the path with the atmosphere in order to generate an acceptable profile for MODTRAN. These problems are solved by IRIMAGE.

2.2.3 Determining Optical Path using Ray Tracing

Backward ray tracing, considered to be an accepted method for generating physically accurate images, traces rays from a detector, through the optical system, into the background scene to determine the exact sources of radiation as well as how each source contributes to the overall intensity. In addition, ray tracing correctly models the refractive and reflective properties of an optical system. These properties make it ideal for determining the optical path through the atmosphere used by MODTRAN.

Various methods may be employed to generate rays [12]. The monte-carlo method uses a probability function to generate a set of rays for each detector. In figure 2.1b, we see that rays are fired off in different directions. Each detector might generate just a few rays or several hundred rays. Since each ray is computed separately and averaged with the others, the time involved is considerable. The accuracy of the model depends on the probability function that defines the ray distribution and the number of rays generated. Another method involves generating a single ray per detector and selecting the most likely path it will take as seen in figure 2.1c. Although considerably faster, this method is less accurate and depends greatly on determining the most likely path.

In essence, the path method employed by MODTRAN is just a ray tracing calculation. The user defines the path, i.e., ray, through the atmosphere with an opaque surface at the end. Using the intercepted profile and the surface characteristics, MODTRAN determines the radiance along that path. The drawback to MODTRAN is its long computation time. Although a single ray calculation may take a fraction of a second, when one considers that a 128x128 array has 16,384 detectors, it is clear that a Monte Carlo method is not feasible. Therefore, IRIMAGE employs a single ray per detector method with the possibility of sub dividing a detector into a set of sub-areas. As shown in figure 2.1d, each sub-area also only represents a single ray, but it allows each detector to sample multiple regions of the background to generate

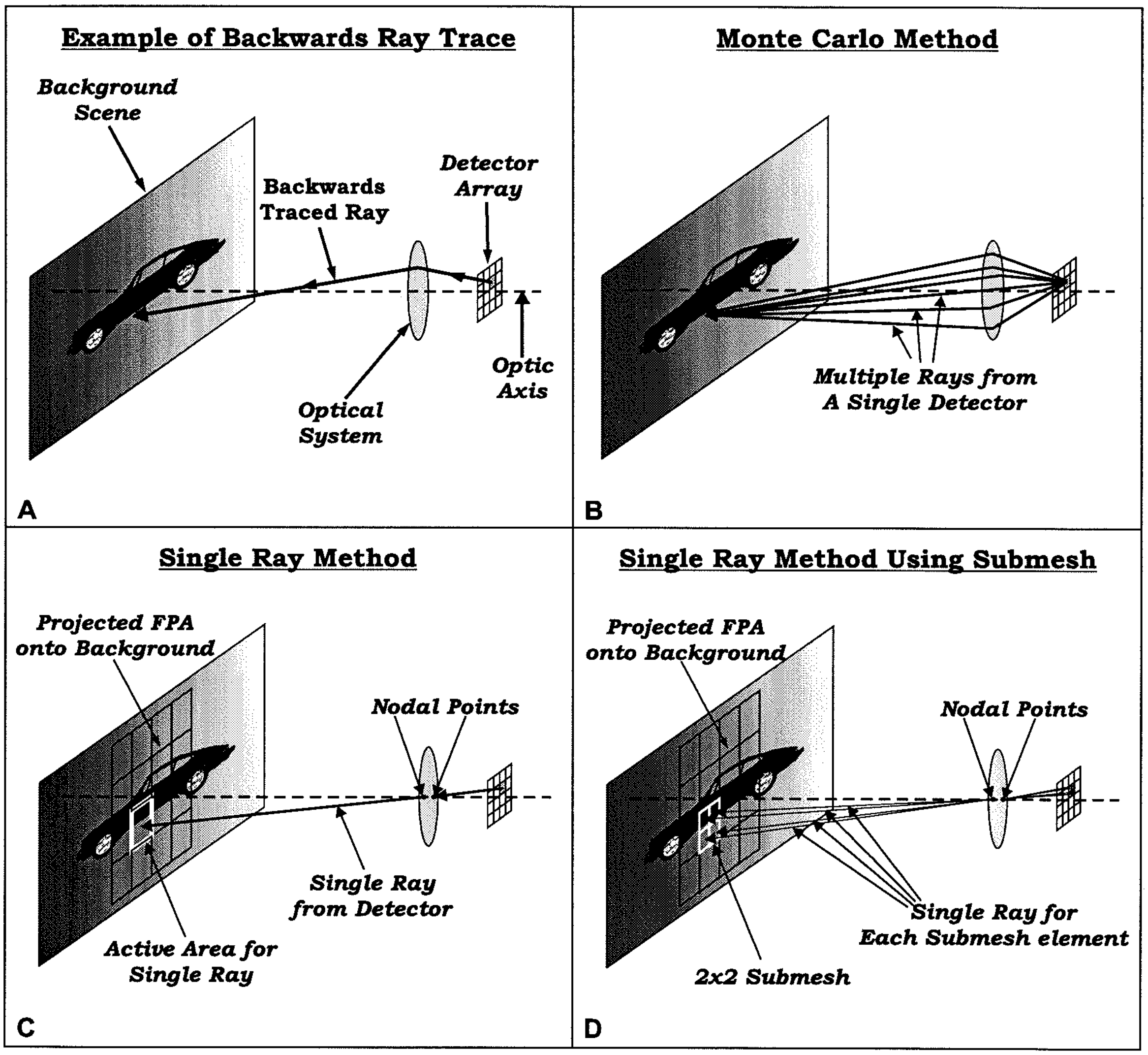

Example of Backwards Ray Trace
Background Scene
Backwards Traced Ray
Detector Array
Optic Axis
Optical System
A
Monte Carlo Method
Multiple Rays from A Single Detector
B
Single Ray Method
Projected FPA onto Background
Nodal Points
Single Ray from Detector
Active Area for Single Ray
C
Single Ray Method Using Submesh
Projected FPA onto Background
Nodal Points
Single Ray for Each Submesh element
2x2 Submesh
D

Figure 2.1: Backwards ray tracing methods: (a) Example of a backwards ray trace. (b) Monte Carlo method. (c) Single ray per pixel method. (d) Single ray method using sub-mesh.

a more accurate image.

Although various algorithms, equations and transfer matrices have been developed to transform an incident ray on an optical system into an exiting ray, the single ray per detector nature of the IRIMAGE calculation requires a less sophisticated approach. The nodal points of an optical system are defined as the points on the optical axis where a ray enters and exits with the same direction vector. If one assumes the system obeys the paraxial approximation, then the most likely path for the ray can be assumed to pass through the nodal points. Since a pin-hole camera follows this same principle, we refer to this method as the *Pin-Hole Approximation.* The beauty of this approximation is that the attenuation properties of the system are preserved provided they are not spatially dependent (e.g., Fourier Optics). In addition, the nodal points can be calculated using the transfer matrices of the optical elements mentioned above. One simply solves for the point along the optical axis where the incoming and outgoing rays have the same direction vector. There are numerous optical design packages which will compute the cardinal points of an optical system including the nodal points, so IRIMAGE does not compute them. It relies on the user to enter the coordinates of these points into the optics parameter file.

2.2.4 Computing the Incident Radiation upon the FPA

The ray tracing approach defines a ray's path through the atmosphere as well as the incident point on the background surface. Using this information, one can use MODTRAN to compute the radiance value associated with the ray. However, this value does not account for the orientation of the background surface, or the solid angle between an element of the background surface and the surface of the aperture of the optical system (see figure 2.2). These two quantities are necessary to compute the radiation collected by the optical system and focused on the focal plane.

Figure 2.2 shows the generalized situation of an arbitrary surface (S_1) emitting radiation onto surface (S_2), the imaging system apeture. The ray from S_1 to S_2 is usually not along either surface's normal, but tends to be at some angle. If we

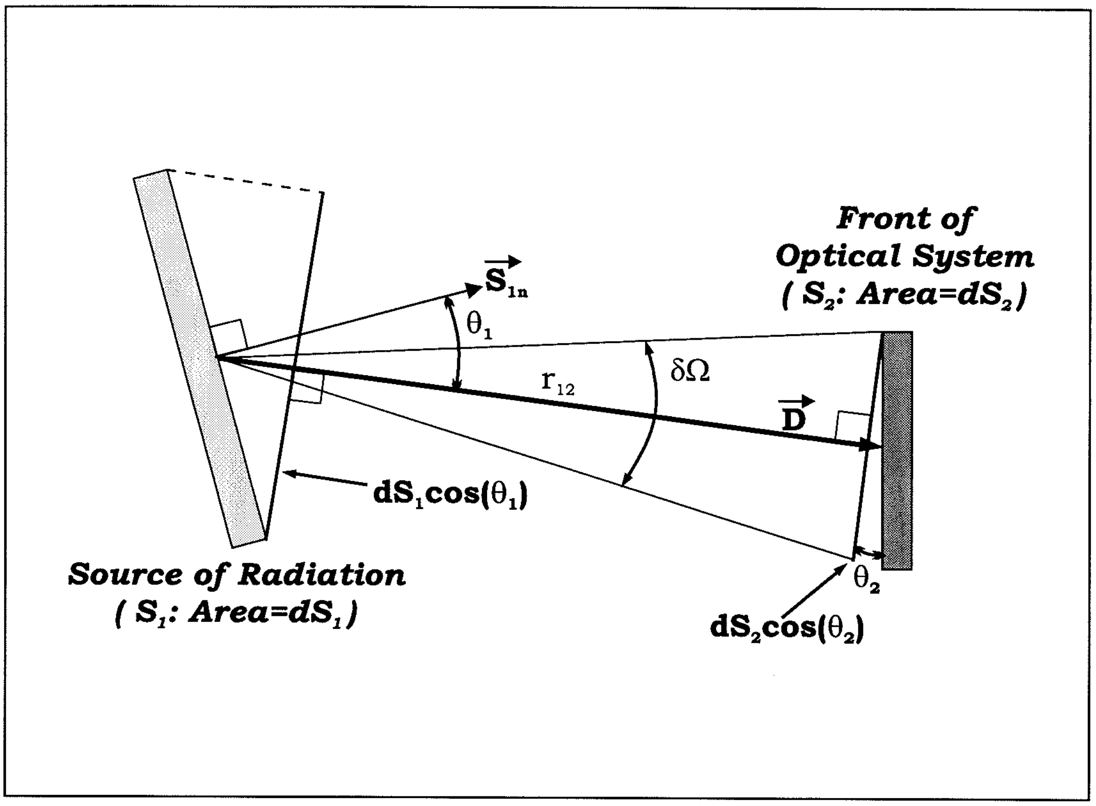


Figure 2.2: Generalized case of a radiating surface and receiving surface.

assume the radiating surface is Lambertian, i.e., radiates equally in all directions, then only the effective area of the radiating surface (dA_1) is needed. This effective area is just the area projected on a plane perpendicular to the ray. Since the radiance relies on the solid angle swept out by the receiving surface, then the solid angle must also be determined. Using the effective area of the receiving surface (dA_2) and the distance between the two surfaces (r_{12}), one can compute the solid angle. Using r_{12} requires that the distances from S_1 and S_2 to their respective projection planes is small compared to r_{12} and can be neglected.

$$\text{Effective Area of } S_1 : dA_1 = dS_1 \cos\theta_1 \tag{2.2.11}$$

$$\text{Effective Area of } S_2 : dA_2 = dS_2 \cos\theta_2 \tag{2.2.12}$$

$$\text{Solid Angle } (S_1 \text{ to } S_2) : d\Omega = \frac{dA_2}{r_{12}^2} = \frac{dS_2\cos\theta_2}{r_{12}^2} \tag{2.2.13}$$

Using the above quantities, we can then define the collected spectral power at S_2 to be:

$$\begin{aligned} P_{S_2}(\nu) &= L_{GB}(\nu) dA_1 d\Omega \\ &= L_{GB}(\nu)(dS_1 cos\theta_1)(\frac{dS_2 cos\theta_2}{r_{12}^2}) \end{aligned} \tag{2.2.14}$$

The one drawback to the backward ray tracing approach is that the detection ray only determines the interception point on the radiating surface. The interception point can provide the parameters for computing the radiance of the surface, but the effective area seen by the detector is necessary to compute the power incident on each detector. Figure 2.3a shows the single ray approach using a detector at a location (x, y) on the focal plane. We project rays from the four corners of the detector through the nodal points of the optical system onto the radiating surface. The interception points define the "active" area, dS, on the radiating surface for the detector. This area is assumed to have the same parameters as the point intercepted by the detection ray, $\overrightarrow{D}$. Using equation (2.2.11) and the angle between normal vector of the surface S_1, $\overrightarrow{S_{1n}}$, and $\overrightarrow{D}$, we can compute the effective area of the radiating surface and consequently the incident power on the optical system. This approach is independent of the ray tracing method selected even though the detector area may be sub-divided into smaller areas with a ray per sub-area.

Despite the ease of this approach, it adds several more steps to calculating the collected power. However, there is a way to approximate the effective area of the background surface seen by a detector using the detector's own area. By doing so, we can reduce the number of steps and computations significantly. The approximation relies on the principle of similar triangles. In figure 2.3a, the rays projected from the detector vertices onto the background sweep out a solid angle on either side of the optical system. If the rays all pass through the nodal points, then by definition the two solid angles will be equal:

$$d\Omega_{detector} = d\Omega_{surface} \tag{2.2.15}$$

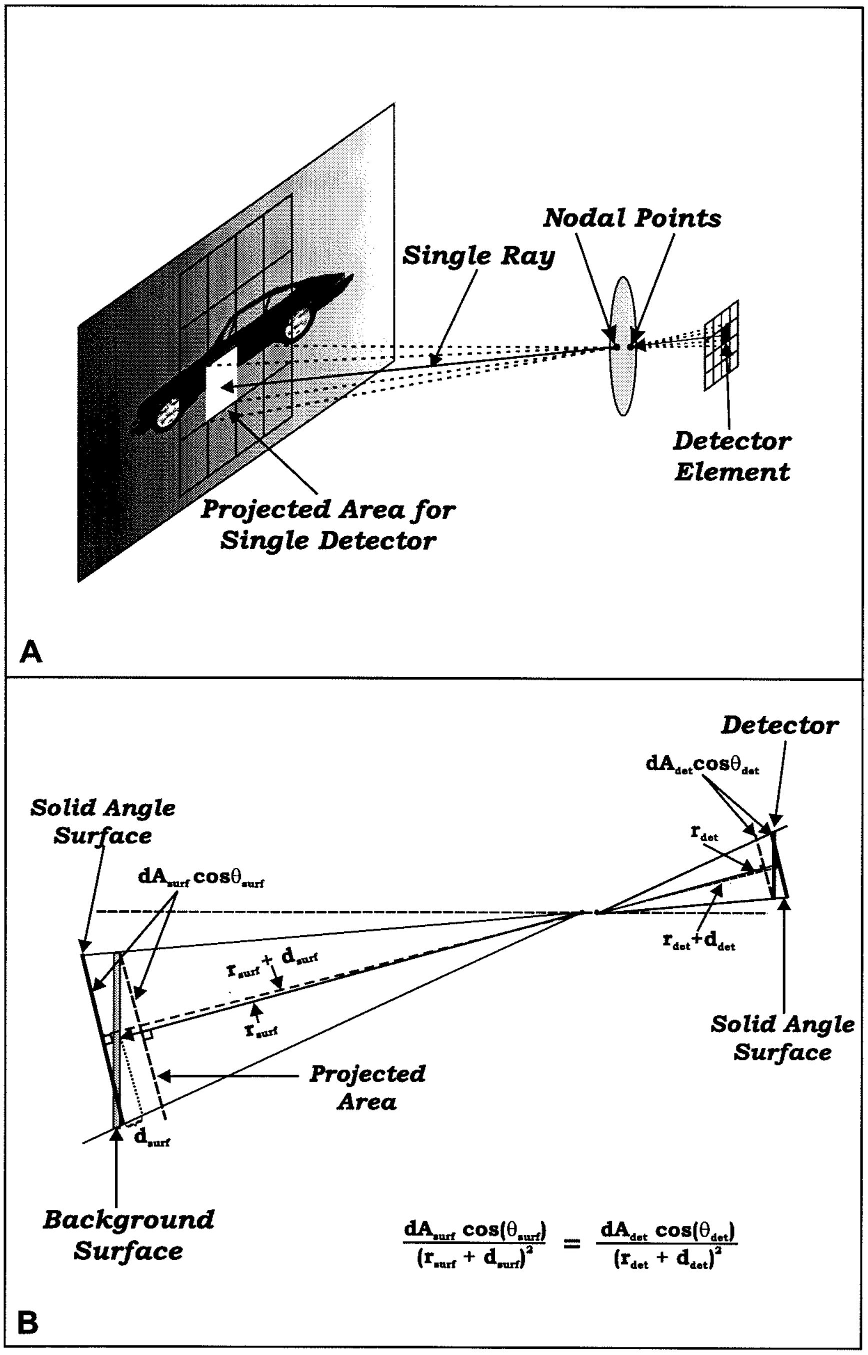

Nodal Points
Single Ray
Detector Element
Projected Area for Single Detector
A
Detector
$dA_{det}\cos\theta_{det}$
r_{det}
$r_{det}+d_{det}$
Solid Angle Surface
Solid Angle Surface
$dA_{surf}\cos\theta_{surf}$
$r_{surf}+d_{surf}$
r_{surf}
Projected Area
d_{surf}
Background Surface
$\frac{dA_{surf}\cos(\theta_{surf})}{(r_{surf}+d_{surf})^2} = \frac{dA_{det}\cos(\theta_{det})}{(r_{det}+d_{det})^2}$
B

Figure 2.3: (a) Relationship between detector element area and project surface area. (b) Using properties of rays passing through the nodal points, the solid angles swept out on either side of the optical system should be equivalent. This fact allows us to relate the visible surface area of the source and surface area of the detector using this solid angle equivalence.

In order to determine the solid angle, we need the area swept out at a particular radius value. On the source side, we can assume that the area swept out is equal to the visible surface area determined in (2.2.11), i.e., $dA_s = dA_{surf} \cos\theta_{surf}$. As can be seen in figure 2.3b, the radius vector to the center of the *solid angle area*, r_{surf}, is slightly longer than the radius vector to the center of the *actual surface area* which is expressed as $r_{surf} + d_{surf}$. Similarly on the detector side, we use the visible surface area of the detector, $dA_d = dA_{det} \cos\theta_{surf}$, and the radius vector $r_{det} + d_{det}$. Using these values in (2.2.15), we get the following:

$$\frac{dA_{det}\cos\theta_{det}}{(r_{det}+d_{det})^2} = \frac{dA_{surf}\cos\theta_{surf}}{(r_{surf}+d_{surf})^2} \tag{2.2.16}$$

Assuming that in the paraxial limit, $d_{surf} \ll r_{surf}$ and $d_{det} \ll r_{det}$, then

$$r_{surf} + d_{surf} \approx r_{surf}$$
$$r_{det} + d_{det} \approx r_{det}$$

By using this approximation and manipulating the terms, equation (2.2.16) becomes

$$dA_{surf}\cos\theta_{surf} = dA_{det}\cos\theta_{det}\left(\frac{r_{surf}}{r_{det}}\right)^2 \tag{2.2.17}$$

Finally, let S_1 be the radiating surface and S_2 be the aperture of the optical system. If we take (2.2.17) and substitute it in (2.2.14), the resultant power collected by the aperture, P_{apt}, is:

$$P_{apt}(\nu) = L(\nu)\frac{(dA_{det}\cos\theta_{det})(dA_{apt}\cos\theta_{apt})}{r_{det}^2} \tag{2.2.18}$$

Equation (2.2.18) describes the power collected using the single ray approach. Since the sub-area ray tracing method is an extension of the single ray approach, a more general form of (2.2.18) incorporates both methods. The sub-area method divides each detector surface into a mesh of smaller rectangles. Each rectangle acts like a smaller detector with its own detection ray. By summing up the power collected for each rectangle, one can get the total power collected for a detector using the sub-area scheme. If a detector uses an $N \times M$ mesh, the general form of (2.2.18) becomes:

$$P_{apt}(\nu) = \sum_{i,j=0}^{N,M} (P_{apt}(\nu))_{ij} \tag{2.2.19}$$

The single ray case is encompassed by this general equation where $N = M = 1$.

Once the spectral power is collected by the aperture, it can be adjusted by the optical system. As stated in section 2.2.3, the optical system is capable of attenuating the incident spectral power through the use of a spectral transmission curve, $T_{opt}(\nu)$. The transmission curve provides the transmission coefficient for any spectral frequency. Thus, the attenuated spectral power at a frequency ν is the product of the transmission coefficient and spectral power at that frequency:

$$P_{adj}(\nu) = T_{opt}(\nu) P_{apt}(\nu) \tag{2.2.20}$$

In IRIMAGE, the transmission curve is defined by a series of data points joined together to form a piece-wise continuous curve. The simulation derives the value at any point on the transmission curve by linearly interpolating between the two known data points that straddle the desired frequency.

Since we assume that all of the power collected by the optical system is either absorbed by the optical system or incident on the detector, then (2.2.20) in conjunction with (2.2.18) and (2.2.19) describe the spectral power incident on a given detector. Once we know the incident power, it is possible to compute the detector signal and incorporate the background fluctuation noise.

2.2.5 Modeling Detector Response

Once the incident spectral power is determined, it is converted to an output signal by the detector response function, i.e., *the Responsivity*. The response of any detector may or may not be a function of the frequency. For most photon detectors, the responsivity is usually independent of the photon frequency except near the cutoff frequency. However, since there is some dependency, equation (2.2.21) demonstrates how the output signal of a photon detector is simply the convolution of the spectral power function, $P(\nu)$, and the responsivity, $R(\nu)$, of the detector. Thermal detectors rarely depend on the spectral nature of the incident radiation. Instead they simply respond to the change in temperature of the active material which is a result of the total incident power. Therefore, equation (2.2.22) is just the product of the responsivity and the total power.

$$\text{Photon Detector: } V_{sig} = \int R(\nu)P(\nu)\delta\nu \tag{2.2.21}$$

$$\text{Thermal Detector: } V_{sig} = R\int P(\nu)\delta\nu \tag{2.2.22}$$

MODTRAN does not output the spectral power in terms of a function, but returns the spectral power values in terms of a set of discreet spectral lines. The user defines the resolution of the computed power spectrum by setting an integer number of wavenumbers between each computed spectral line. The user must also define the minimum wave number, ν_{min}, and the number of lines to compute, N. Using these factors, MODTRAN computes the average power per cm^{-1} for each spectral line. The discreet nature of the returned computed power reduces equations (2.2.21) and (2.2.22) to the form of a Riemann sum:

$$\text{Photon Detector: } V_{sig} = \sum_{\nu=\nu_{min}}^{\nu_{min}+N\Delta\nu} R(\nu)P(\nu)\Delta\nu \tag{2.2.23}$$

$$\text{Thermal Detector: } V_{sig} = R\sum_{\nu=\nu_{min}}^{\nu_{min}+N\Delta\nu} P(\nu)\Delta\nu \tag{2.2.24}$$

The responsivity of a detector is determined either by using a sophisticated device simulation or experimentally. In either case, the output should be a set of data points which may be frequency dependent. If this data could be fit to a specific function, then one would simply need to plug that function into (2.2.23) or (2.2.24) and sum over the various frequencies. However, since various detectors may have their own unique fitting function, it would be impractical to recode the simulation for each individul case. Another method simply treats the data as a piece-wise continuous curve where each point is joined to the adjacent points by lines. Figure 2.4a demonstrates how a response curve can be made up of points connected by lines.

From this type of representation, the responsivity can be interpolated between any two points on the curve. Numerically, the responsivity is determined by either finding two known data points whose frequencies either straddle the given frequency or by finding a data point that shares the same frequency. In figure 2.4b, a given frequency straddles two frequencies. Using (2.2.25), the desired responsivity is found by linearly interpolating the given frequency between the two known responsivities.

$$\begin{aligned} R(\nu_0) &= m(\nu_0 - \nu_n) + R(\nu_n) \\ &= \frac{R(\nu_n) - R(\nu_{n+1})}{\nu_n - \nu_{n+1}}(\nu_0 - \nu_n) + R(\nu_n) \end{aligned} \tag{2.2.25}$$

Although this method is less elegant than a fitting method, it makes modeling detectors much easier. The algorithm to compute $R(\nu_0)$ consists of a search routine to find the two straddle points followed by using the above equation or finding an equivalence point. This method allows IRIMAGE to define a generic response algorithm instead of a series of detector specific, response algorithms. In addition, this algorithm is based on actual computed or measured data instead of relying on the user to write their own detector response routines. However, as we will discuss in the next section, the object-oriented nature of IRIMAGE still allows a user to create their own detector routines and incorporate them into IRIMAGE.

2.2.6 Modeling Background Fluctuation Noise

Although computer simulations are an excellent way of getting precise answers, precision does not always mean accuracy. In the case of detecting IR radiation, all of the theory presented up to this point ignores background fluctuations of the incident radiation. In order to account for this random behavior, the simulation must include a way of modeling these fluctuations.

Background fluctuation of IR radiation is due to statistically random variations of the radiant flux of photons, number of photons emitted per unit area per second,

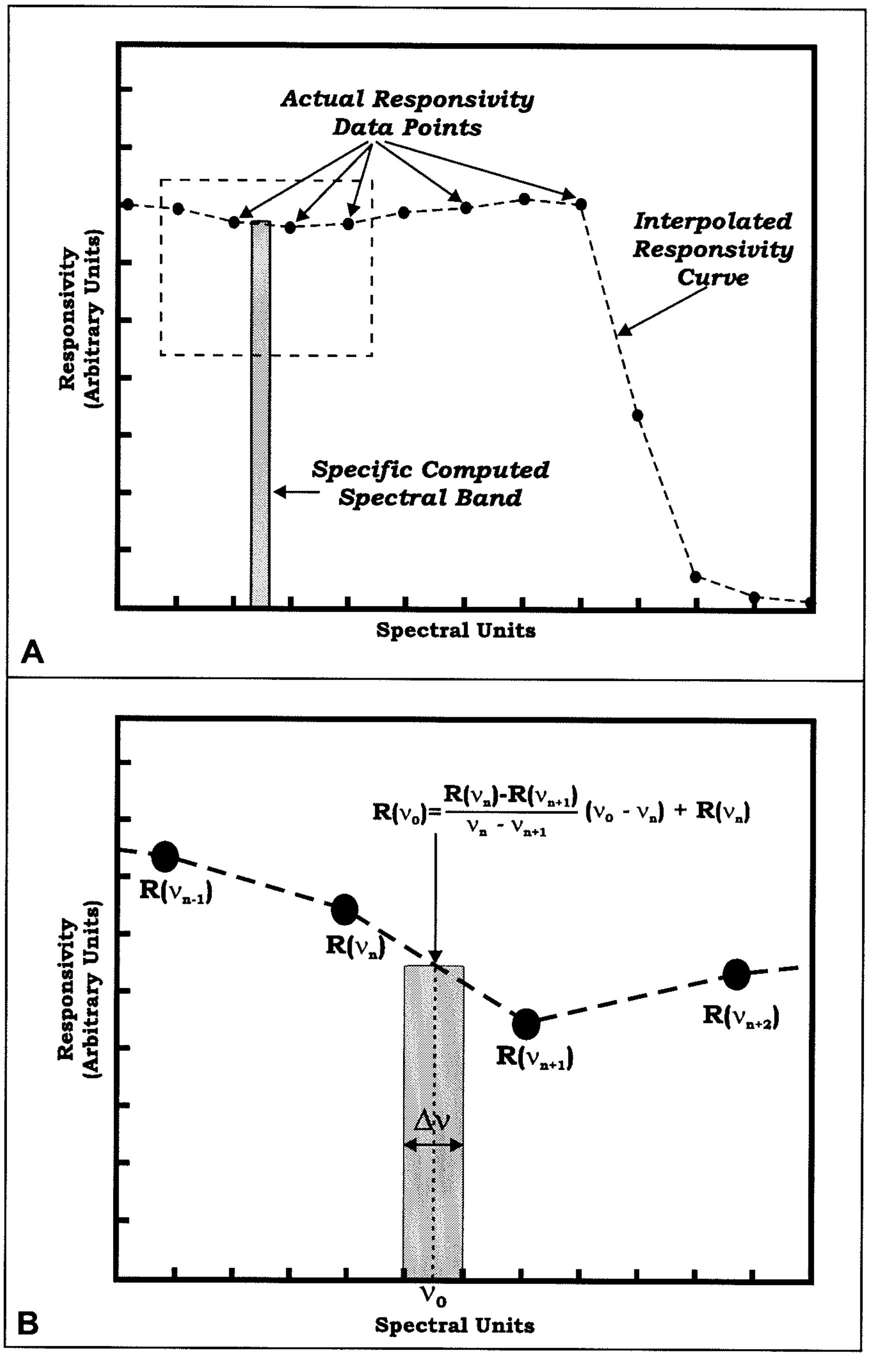

Actual Responsivity Data Points
Interpolated Responsivity Curve
Specific Computed Spectral Band
Responsivity (Arbitrary Units)
Spectral Units
A
$R(\nu_0) = \frac{R(\nu_n) - R(\nu_{n+1})}{\nu_n - \nu_{n+1}} (\nu_0 - \nu_n) + R(\nu_n)$
$R(\nu_{n-1})$
$R(\nu_n)$
$R(\nu_{n+1})$
$R(\nu_{n+2})$
$\Delta\nu$
ν_0
Responsivity (Arbitrary Units)
Spectral Units
B

Figure 2.4: (a) Responsivity curve composed of specific data points connected by straight lines. (b) Interpolating the responsivity of a detector for a spectral band with frequency, ν_0, and bandwidth, $\Delta\nu$, that lies between two points on the responsivity curve.

from the background. These variations may come from the background scene, the atmosphere and even the optical system. We define that the mean number of photons incident upon a detector with surface area A_D and integration time t_{int} is $\overline{N}$. Assuming that the fluctuations follow a random-walk statistical pattern, then equation (2.2.26) describes what the mean square fluctuations of incident photons, $\overline{(\Delta N)^2}$, upon A_D in time t_{int} should be:

$$\overline{(\Delta N)^2} = \overline{[N - \overline{N}]^2} \approx \overline{N} \tag{2.2.26}$$

Incorporating the discreet spectral nature of MODTRAN:

$$\begin{aligned} \overline{(\Delta N)^2} &= \sum_{\nu} \overline{[N(\nu)\Delta\nu - \overline{N(\nu)}\Delta\nu]^2} \\ &= \sum_{\nu} \overline{(\Delta N(\nu))^2}(\Delta\nu)^2 \\ &= \sum_{\nu} \overline{N(\nu)}\Delta\nu \end{aligned} \tag{2.2.27}$$

Since both $\overline{\Delta N^2}$ and $\overline{N}$ are just numbers, any problems with unit dependencies have been eliminated.

Our noise model utilizes a known probability function and a random number generator to generate realistic random noise voltages. In order to generate the noise probability function, the standard deviation of the noise voltage is necessary. Using the results of (2.2.27), we compute the standard deviation of the noise voltage.

Given the spectral power, $P(\nu) = N(\nu)h\nu$, and using the results of section 2.2.5, one can define $\Delta V_{noise}(\nu)$ to be:

$$\begin{aligned} \Delta V_{noise}(\nu)\Delta\nu &= (V(\nu) - \overline{V(\nu)})\Delta\nu \\ &= R(\nu)[P(\nu) - \overline{P(\nu)}]\Delta\nu \\ &= R(\nu)h\nu[N(\nu) - \overline{N(\nu)}]\Delta\nu \\ &= R(\nu)h\nu\Delta N\Delta\nu \end{aligned} \tag{2.2.28}$$

Summing over all available frequencies:

$$\begin{aligned}\Delta V_{noise} &= \sum_{\nu} \Delta V(\nu)\Delta\nu \\ &= \sum_{\nu} R(\nu)h\nu[N(\nu) - \overline{N(\nu)}]\Delta\nu\end{aligned} \tag{2.2.29}$$

Squaring and averaging equation (2.2.29) results in the following equation:

$$\begin{aligned}\overline{(\Delta V_{noise})^2} &= \overline{(\sum_{\nu} R(\nu)h\nu[N(\nu) - \overline{N(\nu)}]\Delta\nu)^2} \\ &= \overline{\sum_{\nu_1,\nu_2} R(\nu_1)R(\nu_2)h^2\nu_1\nu_2\Delta N(\nu_1)\Delta N(\nu_2)(\Delta\nu)^2} \\ &= \sum_{\nu_1,\nu_2} R(\nu_1)R(\nu_2)h^2\nu_1\nu_2\overline{\Delta N(\nu_1)\Delta N(\nu_2)}(\Delta\nu)^2\end{aligned} \tag{2.2.30}$$

We do not average the responsivity because we assume that it is either defined empirically or is already averaged. In addition, $\overline{\Delta N(\nu_1)\Delta N(\nu_2)}$ can be reduced to two cases.

For $\nu_1 = \nu_2$:

$$\overline{\Delta N(\nu_1)\Delta N(\nu_2)} = \overline{(\Delta N(\nu))^2}$$

For $\nu_1 \neq \nu_2$

$$\begin{aligned}\overline{\Delta N(\nu_1)\Delta N(\nu_2)} &= \overline{[N(\nu_1) - \overline{N(\nu_1)}][N(\nu_2) - \overline{N(\nu_2)}]} \\ &= \overline{N(\nu_1)N(\nu_2)} - \overline{\overline{N(\nu_1)}N(\nu_2)} - \overline{N(\nu_1)\overline{N(\nu_2)}} + \overline{\overline{N(\nu_1)}\,\overline{N(\nu_2)}}\end{aligned}$$

since $N(\nu_1)$ and $N(\nu_2)$ are independent variables,

$$\begin{aligned}&= \overline{N(\nu_1)N(\nu_2)} - \overline{N(\nu_1)N(\nu_2)} - \overline{N(\nu_1)N(\nu_2)} + \overline{N(\nu_1)N(\nu_2)} \\ &= 0\end{aligned}$$

Plugging these two cases into (2.2.30) and incorporating (2.2.27)

$$\begin{aligned}\overline{(\Delta V_{noise})^2} &= \sum_{\nu}(R(\nu)h\nu)^2\overline{(\Delta N(\nu))^2}(\Delta\nu)^2 \\ &= \sum_{\nu}(R(\nu)h\nu)^2\overline{N(\nu)}\Delta\nu\end{aligned} \tag{2.2.31}$$

Equation (2.2.31) defines the standard deviation of the noise voltage from the mean voltage. Using any normalized probability function and the noise voltage standard deviation, one can generate a random distribution of noise voltages that is weighted by the probability function through the use of the probability density function. In order to understand this technique, we must revisit the Probability Density function, $F(\alpha)$.

$$F(\alpha) = \int_{-\infty}^{\alpha} P(x)\delta x \tag{2.2.32}$$

Equation (2.2.32) defines $F(\alpha)$ to be the probability of finding a value that is less than or equal to α. For a normalized probability function, $P(x)$, the value of $F(\alpha)$ must be between 0 and 1. If we assume that α and x are voltages and $P(x)$ is the noise probability function defined by a mean noise voltage, $\overline{V_n}$, and the standard deviation, $\sigma_n = \sqrt{\overline{(\Delta V)^2}}$, then one can determine the probability density for any voltage.

If we were to reverse this idea and assume we know the probability density we want to achieve, then we can solve for the value of α that will generate this probability density. Most computer algorithms generate random numbers with a flat distribution between 0 and 1. However, if one defined the value of the probability density using a number generated by one of these alogithms, then the resultant alpha would be a random value that is weighted by $P(x)$. We can demonstrate this graphically. First, we reduce equation (2.2.32) to an equivalent sum:

$$F(\alpha) = \sum_{x=-\infty}^{\alpha} \Delta F(x) = \sum_{x=-\infty}^{\alpha} P(x)\Delta x \tag{2.2.33}$$

In figure 2.5a, we show the classic graphical representation of a Reimann sum for

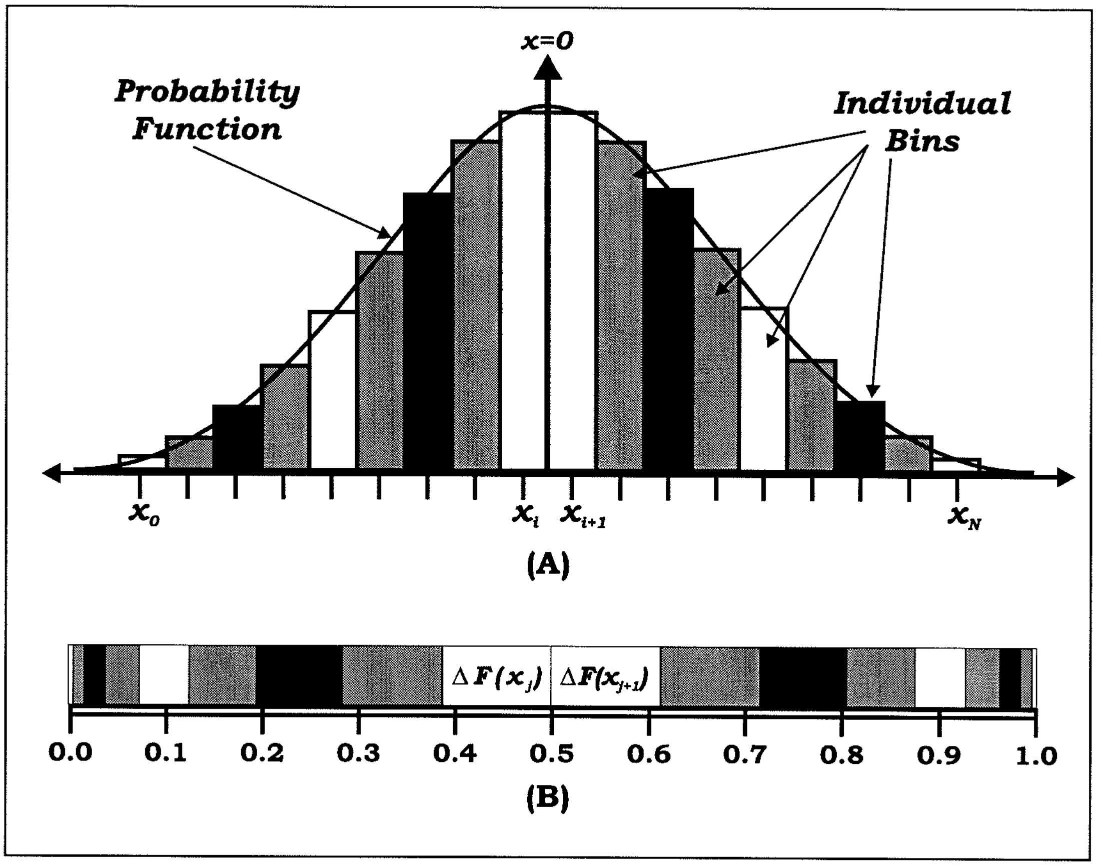


Figure 2.5: Number line representation of integrating the Probability Density function using a Reimann sum. (a) Normal representation of $P(x)$ with equally spaced divisions. (b) Number line with regions representing the area of each equally spaced division.

a normalized Gaussian probability function where the area of each rectangle defines $\Delta F(x_i)$. The centers of each rectangle are evenly spaced by the value Δx and the center of rectangle n is defined by $x_n = x_{min} + n\Delta x$. In figure 2.5b, each $\Delta F(x_i)$ is stacked sequentially along a number line from 0 to 1, effectively summing over all of the $\Delta F(x_i)$ values. By doing so, each $\Delta F(x_i)$ defines a small but unique region of the number line which corresponds to the value x_i. Therefore, given a specific $F(\alpha)$ which defines a point on the number line, we would simply determine which $DeltaF(x_i)$ region contains this point and set α equal to the corresponding x_i.

The real advantage of this methodology is that the size of each $\Delta F(x_i)$ region is proportional to the probability of having the value x_i. In the figure, a value of x near the peak of the Gaussian corresponds to large range of F values compared to an x value near one of the edges. In general, for regions of high probability, a large range of $F(x)$ will correspond to a small range of x values, while in regions of low probability, the exact opposite is the case. Therefore, a random value $F(\alpha)$ will have an α which is weighted by the probability function, $P(x)$.

Using a numerical algorithm based on the number line method above and the results of equation (2.2.31), IRIMAGE can generate a realistic random noise voltage from the following method:

1. Compute standard deviation of the noise voltage, σ_N, using (2.2.31).
2. Define or compute the following parameters for the probability density sum:
   - The mean voltage: $\mu = 0$
   - The voltage range: $-A\sigma_n <= x <= +A\sigma_n$ where $A > 0$ and a real number.
   - The number of $\Delta F(x)$ regions to sum over: N.
   - The voltage resolution: $\Delta x = 2A\sigma_n/N$
3. Plug σ_N and μ into the Gaussian probability function, $P_{Gauss}(x)$.
4. Compute each $\Delta F(x_i)$ and store each $\Delta F(x)$ in an array.
5. Sum over all of the $\Delta F(x_i)$ and normalize them so that sum equals 1.
6. Use a random number generator to get a number, F_0, between 0 and 1.
7. Find the noise voltage value, α, by summing over the array of $\Delta F(x_i)$ values until $F(x) > F_0$ making $\alpha = x$.

Although the mean value, μ, should be the signal voltage, there are several advantages to setting the mean of the probability function to zero. With a zero mean, $P_{Gauss}(x)$ is symetric around zero making the range symetric around zero as well which simplifies the calculation. In addition, the mean only effects the location of the central point of $P_{Gauss}(x)$ and not the shape. Thus, the only effect would be that α would have the mean value added to it. Since we are interested in outputing this signal, noise and total voltages, it is just as easy to sum the noise and signal voltages after the noise calculation. Therefore, a zero mean does not effect the total voltage and allows us to separate signal and noise voltages quickly.

The voltage range and value of N may effect the outcome. The voltage range determines how much of the probability function to sum over while N determines the resolution of the approximation. If the range is too small, then the result will be skewed toward a more even distribution and not a Gaussian. However, if the range is too large and N is too small, then the resolution of the calculation may be too rough which could also alter the distribution and could produce unpredictable results. Even for a reasonable range, if N is too small, then it limits the most probable noise voltages to only a few noise voltages since only a few are in the high probability portion of the distribution while the rest are in the low probability regions. On the other hand, the larger N becomes, the slower the calculation takes. Therefore, a value of $A = 2$ is sufficiently large enough to encompass most of the Gaussian and a value of $N = 1000$ provides a reasonable value for the resolution.

2.3 Computational Concepts

In the previous section, we discussed the physical principles that govern the IRIMAGE simulation. Due to the object-oriented approach used by IRIMAGE, it is beneficial to define certain computational constructs and programming objects which will simplify the description of the different simulation objects. Since any imaging simulation is heavily dependent upon the positions of various objects, we briefly introduce the *World Coordinate System (WCS)* used by IRIMAGE. Furthermore, the *Versatile Grid Object (VGO)* is an integral part of the Scene, Atmosphere and FPA objects which makes it necessary to discuss it as well as the methods the VGO uses to manipulate both two and three-dimensional data. Finally, we introduce the Ray program object which stores the information generated by the backwards ray tracing method. Although many of these concepts are introduced in appendix B, they are discussed here with respect to IRIMAGE.

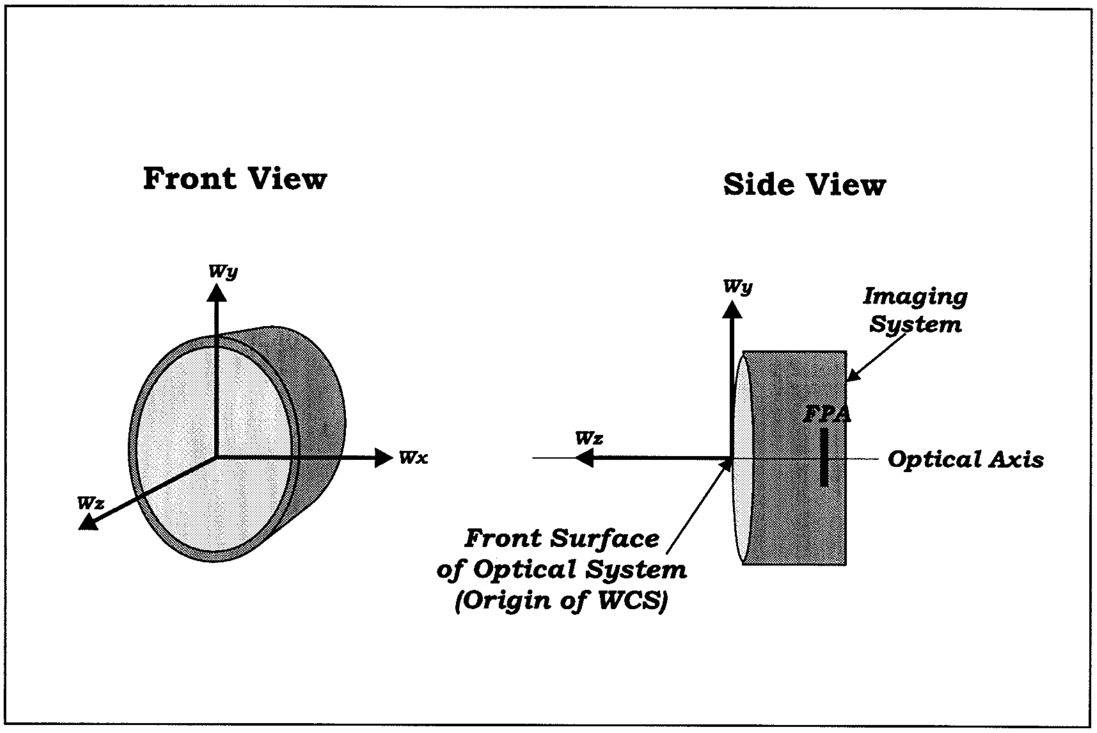


Figure 2.6: Orientation of WCS with respect to the imaging system. The WCS is a right-handed coordinate system whose origin lies along the optical axis exiting the imaging system and sits on the front surface of the optical system. The $+Z$ axis points away from the imaging system along optical axis. The $+X$ and $+Y$ axis are oriented to be horizontal and vertical respectively and follow the right-handed nature of the WCS.

2.3.1 The World Coordinate System (WCS)

As mentioned, any physical simulation of an imaging process depends on the placement of different simulation elements, both geometric models and simulation objects, in three-dimensional space. Some of these elements may even maintain an internal coordinate system. However, IRIMAGE defines a single primary coordinate system called the World Coordinate System (WCS). The WCS is a right-handed coordinate system with the positive Z axis pointing along the optical axis and the Y axis pointing up. Its origin is at the point where the optical axis and the external surface of the optical system intersect. Figure 2.6 shows the location and orientation of the WCS with respect to the imaging system.

Each simulation element must be defined with respect to the WCS or must be defined in a coordinate system that can transform points between itself and the WCS. Therefore, IRIMAGE restricts each element coordinate system (ECS) to be right-handed, orthogonal and share the same unit of length as the WCS. In addition, any element that uses an ECS must maintain a set of transformation matrices. These matrices allow a point defined in the ECS to be transformed between the ECS and the WCS. Since the transformation matrices may have to rotate and translate points from one coordinate system to another, the order of the rotations and translations are very important. We define the rotation and translation matrix for a point going from an ECS to the WCS using equation (2.3.1). Since transforming the point back should return it to the same position in the ECS, the other transformation matrix is just the inverse of equation (2.3.1).

$$\mathbf{M_{ECS\rightarrow WCS}} = \mathbf{R_z R_y R_x T} \tag{2.3.1}$$

$$\mathbf{M_{WCS\rightarrow ECS}} = \mathbf{T^{-1} R_x^{-1} R_y^{-1} R_z^{-1}} \tag{2.3.2}$$

Once a point in an ECS is transformed to the WCS, it can subsequently be transformed into a different ECS using the WCS as a bridge. Equations (2.3.3) and (2.3.4) demonstrates how a point, P, is transformed from ECS_i to the WCS and then could be transformed from the WCS to another element's coordinate system, ECS_j.

$$\vec{P}_{WCS} = \mathbf{M_{ECS_i\rightarrow WCS}} \vec{P}_{ECS_i} \tag{2.3.3}$$

$$\vec{P}_{ECS_j} = \mathbf{M_{WCS\rightarrow ECS_j}} \vec{P}_{WCS} = \mathbf{M_{WCS\rightarrow ECS_j}} \mathbf{M_{ECS_i\rightarrow WCS}} \vec{P}_{ECS_i} \tag{2.3.4}$$

These equations demonstrate the usefulness and the need for maintaining a single primary coordinate system like the WCS.

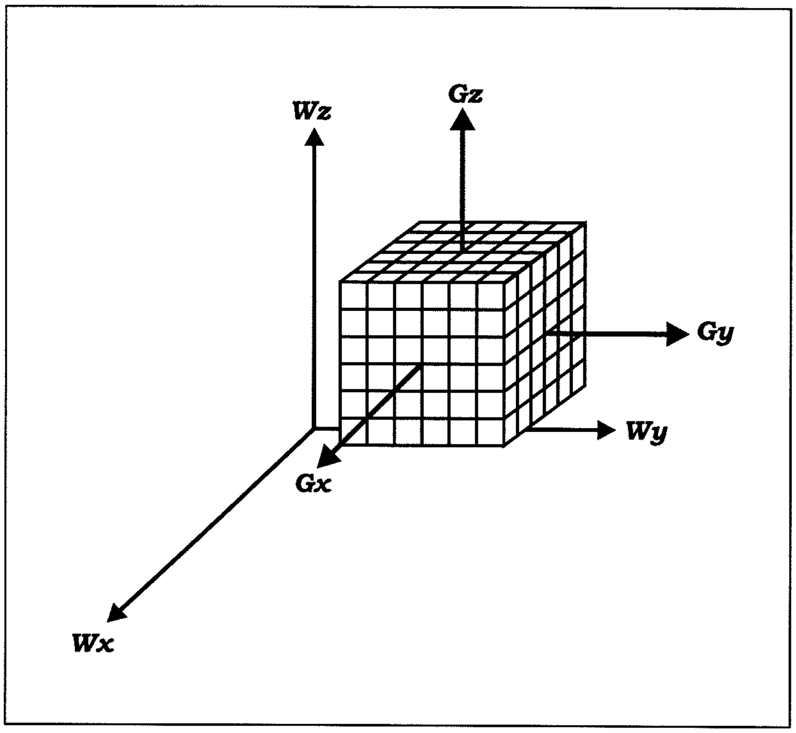


Figure 2.7: Relating the grid coordinate system to the world coordinate system.

2.3.2 The Versatile Grid Object (VGO)

The Versatile Grid Object defines a multi-dimensional, orthogonal grid structure for storing spatially oriented integer or floating point information. While the VGO is inherently three dimensional, its structure is capable of being one, two or three-dimensional. It does maintain its own ECS which is called the Grid Coordinate System (GCS). As shown in figure 2.7, the origin of the GCS lies at the physical center of the grid structure. Using the origin of the GCS and the physical spacing between adjacent grid points, the VGO can determine the spatial coordinates of any grid point in terms of the GCS.

Each VGO uses equations (2.3.3) and (2.3.4) to relate coordinates between the GCS and the WCS. In order to utilize these equations, the VGO requires that the GCS origin and the orientation of the GCS about this origin be defined with respect to the WCS. Using these values to generate the transformation matrices, the VGO can then relate any coordinate in the WCS to one in the GCS.

The VGO is capable of storing various types of data. Each grid point maintains either a floating point or integer data array. Since the VGO is designed to be flexible, the size and data type of the array isn't defined until the VGO is first initialized. Although flexible, the VGO requires every grid point array to be the same size and data type. This requirement avoids a lot of bookkeeping problems. In addition to the data array, each grid point stores a single "source of data" index. Using this index allows the VGO to not only store data from various sources onto the grid, but it allows the VGO to associate a unique index value with the data from a particular source. Using this index, the VGO, or an external routine, can limit data manipulation, retrieval or other operations to data from a particular source.

Three of the six major simulation objects of IRIMAGE utilize the VGO. The Scene simulation object represents the background image surface by mapping temperature and/or emissivity image data onto a 2D form of the VGO. In this case, each grid point simply acts like a pixel of an image except instead of storing a color value, it stores a temperature and emissivity value. Another two-dimensional form of the VGO is used by the FPA simulation object. Since a focal plane array already consists of a grid of detectors, the VGO acts as the natural data structure to store information about each detector in the array. Finally, the Atmosphere simulation object puts a three-dimensional form of the VGO to extensive use. The VGO provides a way of storing three-dimensional volumetric data in a form that the ray tracing technique can translate into an atmospheric profile for use with MODTRAN. These three simulation objects utilize the VGO as a major part of their operation which makes it an excellent example of the benefits of code reuse.

Since interpolating data onto a grid depends upon the dimension of the grid, the VGO does not perform data interpolation. It only loads data directly into the grid point arrays using the spatial coordinates for the data. Actual interpolation of any image or volumetric data onto the grid is left to external routines instead. In the next few sub sections, we discuss how specific two and three-dimensional data is interpolated and prepared for loading into a VGO.

2.3.3 Interpolating 2D Image Data onto the VGO

As mentioned above, the Scene simulation object constructs a background surface model by placing temperature and emissivity images onto a 2D VGO. In general, the method for interpolating image data onto a 2D grid involves mapping the coordinates of each data point onto the grid plane and then using a standard 2D interpolation algorithm to determine the value of each grid point. In IRIMAGE, a program object called a *Virtual Image Object (VIO)* maps the image coordinates onto the image plane and a near point or bilinear interpolation technique determines the value of each grid point. Using this method, any number of images can be loaded onto the VGO in a mosaic form. However, one can also combine objects using a "matte" process in conjunction with another program object called the *Layout Object.* With this added technique, the Scene object can create fairly sophisticated images to represent the background scene using just a few simple images.

Virtual Image Objects

The Scene object does not load images directly onto the imaging grid. Instead, each image used in the simulation is loaded into a database and is placed on the imaging grid using a set of Virtual Image Objects. Each VIO is assigned a particular image and a set of image processing operations which change the placement, scale, or rotation of the image before it is placed on the imaging grid. For simplicity, all of these operations use the Grid Coordinate System of the Scene object instead of the World Coordinate System of the simulation. Figure 2.8 shows an example of an image being scaled and rotated by a VIO. Since a single image may be associated with several VIO's, each VIO processes its image and then returns it to its original state. This eliminates the need for storing the same image multiple times.

Since the design of the Scene object includes animation capabilities, each VIO makes use of a *key frame database* to store the parameters used to process its associated image. Each key frame defines the active state and the image processing parameters of the VIO for a specific frame of the animation. The active state designates whether the VIO is active or not. If the VIO is active, then it uses the

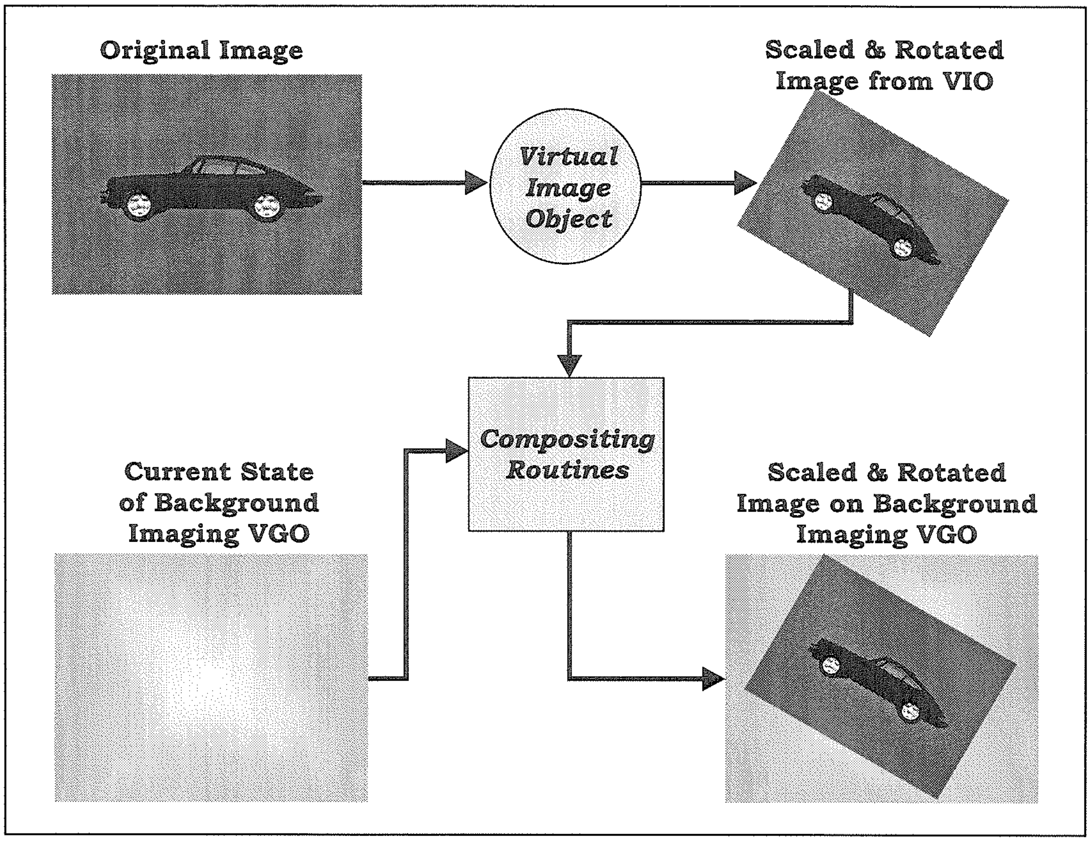

Original Image
Virtual Image Object
Scaled & Rotated Image from VIO
Compositing Routines
Current State of Background Imaging VGO
Scaled & Rotated Image on Background Imaging VGO

Figure 2.8: Porche image being processed by VIO (scaled, rotated and composited).

parameters to process and interpolate the associated image onto the Scene object's VGO. Otherwise, it skips the VIO entirely. Since the active state is just a flag, the active state only changes if another key frame changes it. For the case of a frame that lies between two key frames and the VIO is active, it will compute the image processing parameters by linearly interpolating between the two key frames. The VIO only adjusts its associated image while it interpolates onto the Scene VGO and its original state once this step is complete. By using a series of VIO structures with their associated key frame databases, the Scene simulation object can simulate a dynamically changing and interesting background surface.

Mattes and The Layout Object

In IRIMAGE, the background images are typically made up of several images that are *composited* together. Since most images are inherently rectangular, each image data set needs a *matte*, which is an image that defines what regions of an image to use in the composite and what regions to discard. An extension of this method uses a "soft" matte to combine two images using a simple weighted average. Such a method is useful for anti-aliasing the edges of the combination as well as making any part an image appear to be semi-transparent. The compositing method used by IRIMAGE treats a matte image much like any other image by associating it with a VIO. However, a matte VIO interpolates the matte image data onto the Scene's imaging VGO using a separate matte variable, or "channel." This separate channel makes it possible to combine an image already loaded on the imaging VGO with a new image.

An important factor in compositing is the order that images are loaded onto the imaging VGO. Figure 2.9 shows a car composited onto a background image. If the matte image were not loaded prior to the car image, then the entire car image would have been loaded, blocking out the background. In order to preserve the image loading order, the Scene simulation object employs a Layout program object which stores the order of VIO's loaded for a particular frame. Since the VIO's can be animated, the Layout Object must also be capable of being animated. The key frame database used by the VIO is also used by the Layout object. However, instead of storing processing parameters, it simply stores the VIO loading order. Like the active state in a VIO, the loading order is not interpolated but is set until the next key frame changes it. Through an effective use of matte's and layout order, the Scene simulation object can create a sophisticated background image from just a few interesting temperature/emissivity images.

Near Point and Bilinear Interpolation.

Each VIO that loads an image serves as the link between the GCS of the Scene's VGO and the GCS of the interpolated image's VGO. Although it would seem logical to project the image VGO onto the Scene VGO, it actually works better in reverse. By projecting the Scene VGO onto the image VGO, each grid point in the Scene VGO that lies within a rectangle formed by four points of the image VGO should be interpolated. All other points are considered to be outside of the image and are not interpolated. In addition, the rectangular nature of the four image points

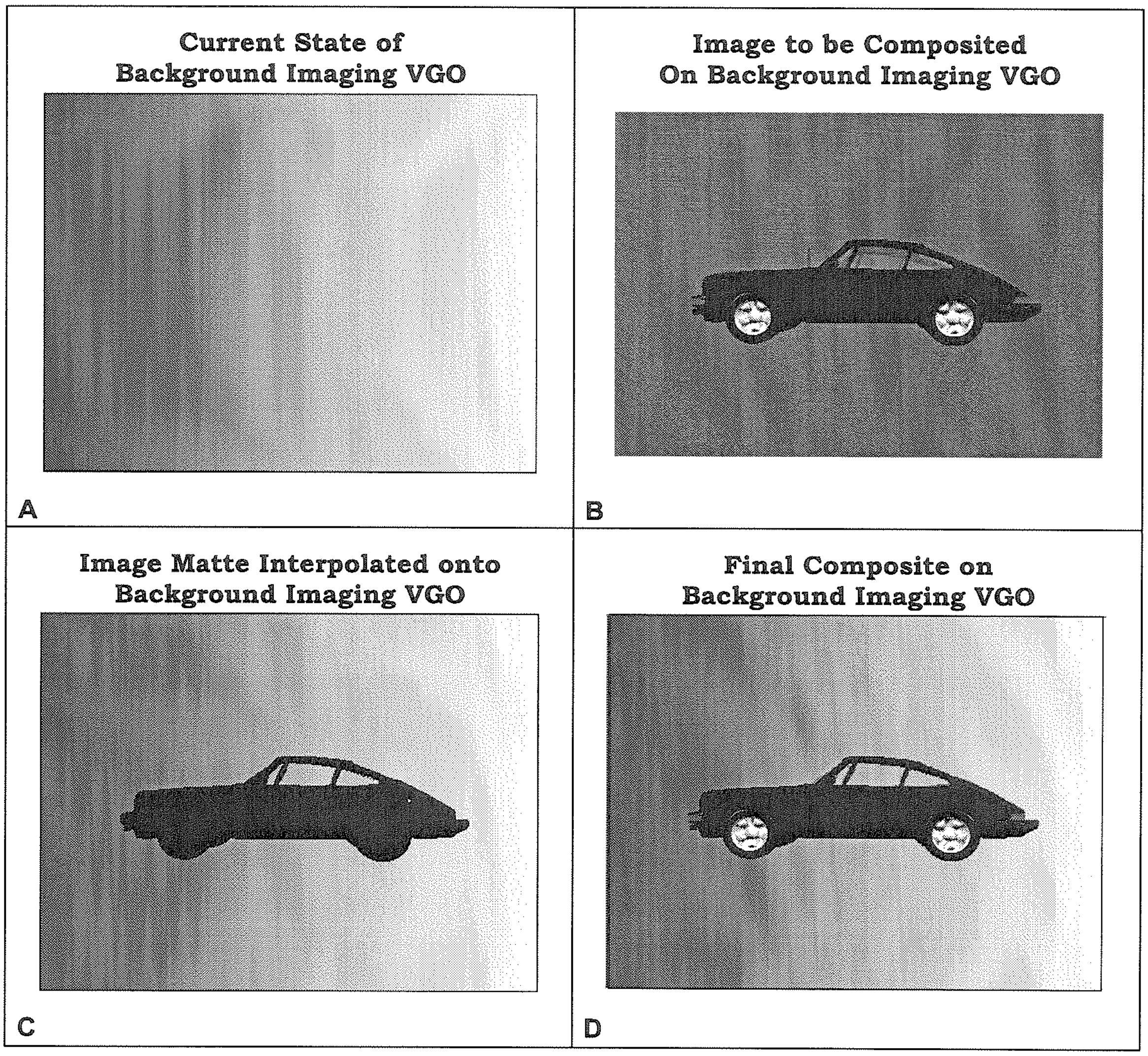


Figure 2.9: Example of the matte process. (a) The original Porche image. (b) The background image of a temperature gradiant. (c) The matte of Porche overlayed on the background image. (d) Final composite image of car on background.

surrounding each Scene grid point makes it easy to compute either the nearest point value or bilinear interpolated value.

In figure 2.10c, a Scene grid point,P, lies inside the rectangle formed by four image grid points. Using the Near Point approximation, this interpolated grid point assumes the value of the nearest image grid point ($P = .7$). This is a quick and dirty method of interpolating points. However, it relies on the fact that the values do not change rapidly between grid points. The Bilinear method [13] utilizes a weighted average between the four image grid points that surround each projected Scene grid point. One can see in figure 2.10d the method projects a line parallel to the X axis and one parallel to the Y axis which pass through the point to be interpolated. By computing the area of each small rectangle as a percentage of the original rectangle's area, the method multiplies the grid point value by its associated rectangle area and sums the four values up. The result is the bilinear interpolation of the four grid values. This method is more accurate because the actual location of P effects its value, but it reuires more time to compute. Both methods are available in IRIMAGE.

2.3.4 Interpolation of Volumetric Data onto the VGO

The Versatile Grid Object is an elegant method for modeling a 2D background surface using temperature or emissivity image data. The advantage of a VGO solution for modeling the 3D atmospheric data lies in the benefits of a Ray-Grid interception technique over a Ray-Geometric Model (GM) technique. For each frame of the Ray-GM approach, each ray checks to see if it intercepts every GM in a the scene. Since only a few GMs in a scene will be along a specific ray's path, a lot of CPU cycles may be wasted checking GMs that will not be hit. However, if one subdivides space into a regular, orthogonal, set of grid boxes and interpolates each GM onto the grid, then most of this wasted time can be averted. In this Ray-Grid case, each ray simply checks each grid box it passes through to determine if part of the GM lies within the grid box. When part of a GM does occupy a grid box, then the method checks the Ray-GM intersection. If the ray does intersect the GM, the search is over. Otherwise,

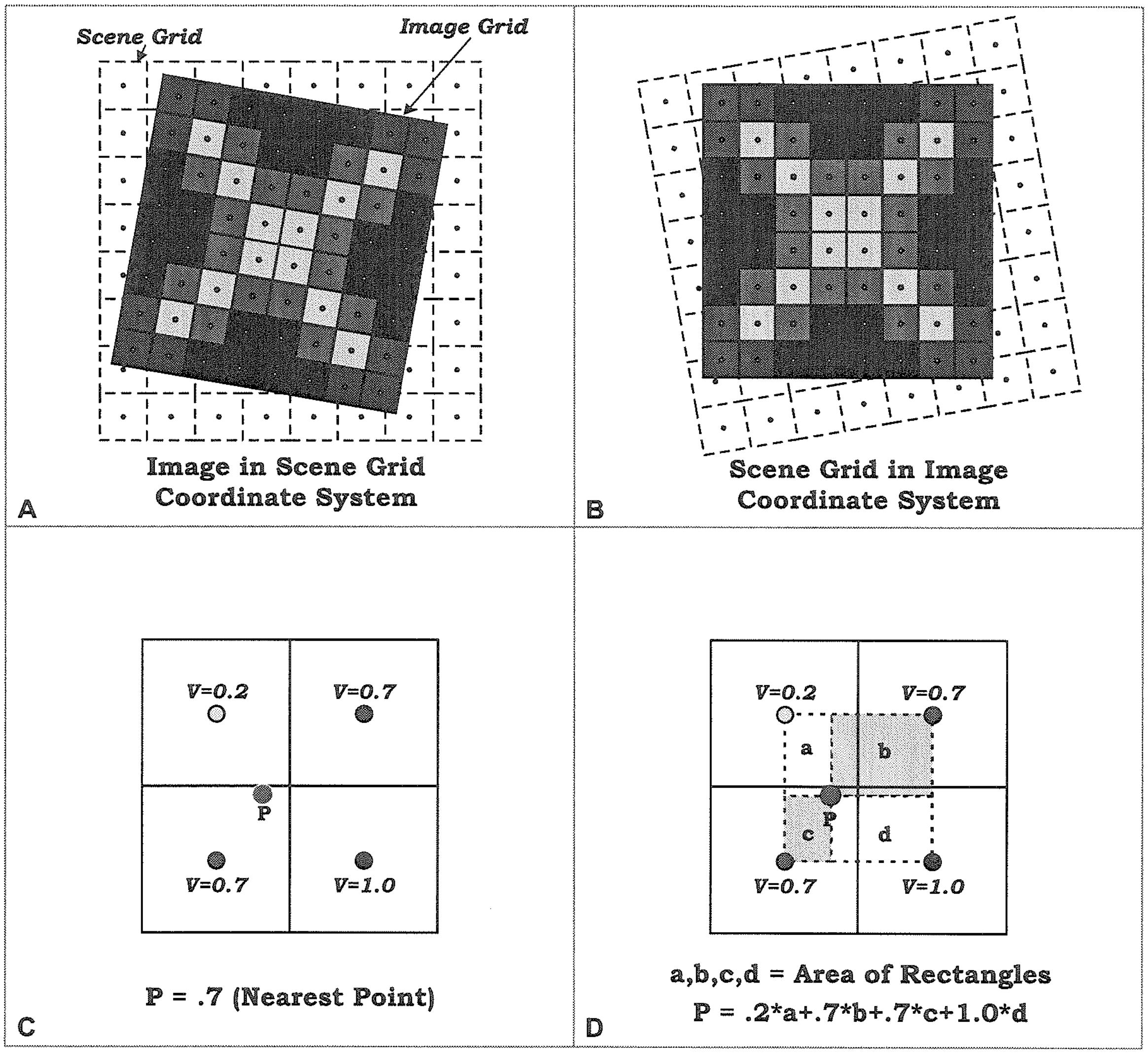


Figure 2.10: Interpolating images on the Scene object's imaging VGO. (a) Projecting the image grid on the Scene VGO. (b) Projecting the Scene VGO onto the image grid. (c) The Nearest Point interpolation method. (d) The Bilinear interpolation method.

the search continues. Various methods like this have been introduced in the past that clearly show a performance improvement by using a grid approach to ray tracing in computer graphics [14, 15].

For most cases, these Ray-Grid methods are only concerned with finding the Ray-GM intersection faster and still require intersecting the ray with the GM. In the case of IRIMAGE, the GMs we deal with are transmissive which make it necessary to continue checking for Ray-GM interactions even after one is found. The checking process goes on until the ray strikes the Scene object's imaging plane. Furthermore, since each GM stores volumetric data, any ray intersecting a GM needs to store the volumetric data in addition to the length of the ray that passes through the GM. The ray must also store data for the regions where a GM does not exist by storing a default set of values and the corresponding path lengths. However, by interpolating these GM's onto a three-dimensional VGO, we can significantly improve this process of checking and storage of data. The Atmosphere simulation object assigns each GM a unique index before it is interpolated on the VGO. As each GM is interpolated, any grid point inside the bounding surface of the GM is assigned the GM's index. As each ray intersects the VGO, it simply keeps track of which grid boxes it passes through and the path length through each box. Once it completely intersects the grid, it can then cycle through the stack of grid boxes and retrieve the indexes for each grid box. Using these indexes and corresponding path lengths, the ray can build an atmospheric profile from the associated GM's atmospheric conditions as well as any default conditions. This modified Ray-Grid intersection method eliminates the need for keeping track of whether the one GM is inside another as well as where a GM ends and the atmosphere begins.

One can see how important it is to accurately interpolate a GM onto a three dimensonal VGO. In IRIMAGE, each GM is defined as a GridObj program object. The GridObj uses a bounded surface approach based on the Universal Primitive Object (UPO) (see appendix B). The UPO is a hybrid, edge-based, surface representation well suited for defining non-intersecting, bounded surfaces. Its hybrid structure is specifically designed for interpolating bounded surfaces onto a three-dimensional orthogonal grid. However, the UPO has several limitations. It only defines the surface in its own coordinate system and it only stores a single unique object index. The GridObj adds the ability to animate the placement, orientation and scale of a UPO's bounded surface in the WCS. It uses the same Key Frame database object used by the VIO's in the Scene simulation object. Since the volumetric data associated with a GM may change over time, each key frame also stores the volumetric data associated with that frame. All of these attributes allow a GM to be completely defined using the GridObj as well as prepare it to be interpolated onto a VGO.

When a GM is ready to be interpolated onto a VGO, the GridObj utilizes a "slice and dice" algorithm defined by Springfield to interpolate its UPO-based bounded surface onto the VGO. This algorithm defines each grid point of the grid structure to be surrounded by a bounding box. If the algorithm finds more than 50% of the grid box's volume to be inside the GM, then GM's volumetric data is associated with the grid point through the grid point's unique index value. Using the GridObj interpolation algorithm, any GM based on the GridObj can build itself, transform its bounded surface to the WCS, be associated with a specific set of volumetric data and then use the bounded surface to interpolate its associated volumetric data onto a three-dimensional VGO.

2.3.5 The Ray Object

The Ray program object generated by IRIMAGE is designed to interact with the three-dimensional Atmosphere VGO and the two-dimensional Scene VGO. Each Ray stores the local origin, a local direction vector and its current endpoint in the WCS. The local origin establishes starting point of the Ray. However, the location of the starting point can be adjusted by another object, such as the Optics simulation object, so that the value stored by the local origin always defines the current starting point of the Ray. In IRIMAGE, this usually begins as the center of a detector on the FPA, but is replaced by the exit point of the optics after the Ray is processed by the Optics object. The direction vector defines the current direction that the Ray points. This vector may also be adjusted by the Optics object. It is used to compute points of interception between the Ray and any surface. The endpoint stores the coordinates of the point where the ray intercepts the background surface, i.e., Scene object's imaging VGO. Using the endpoint, the Scene object can retreive the surface temperature and emissivity from the imaging VGO.

In addition to the geometric properties, the Ray also stores a database of grid indexes and path lengths. The Atmosphere VGO can be defined as a set of grid boxes, or *voxels*, where each point in the grid lies at the center of one of these voxels. As the Ray goes from the exit point of the optics to its endpoint on the background surface, it will pass through the Atmosphere VGO and storing the grid point index to every voxel through which it passes. The Ray-VGO interception algorithm forces these indexes to be stored in the proper order to build an atmospheric profile. As the Ray passes through each voxel, it computes the length of path that it travels in that particular voxel and stores it as well. Using the path lengths in conjunction with the grid point indexes, the Atmosphere object can then use the Ray grid point database to generate the atmospheric profile.

2.4 Major Elements of IRIMAGE

Now that the physical and computational concepts have been introduced, we briefly describe each of the major objects that make up IRIMAGE and how they contribute to the overall simulation. In section 2.5, we will put these objects together to describe the entire simulation process and discuss two test cases.

2.4.1 The Scene Object

The Scene Object uses a two-dimensional imaging "plane" to define the background scene. As mentioned in section 2.3, three-dimensional IR scene generators have been in existence for over 10 years. The object-oriented nature of IRIMAGE and specifically of the Scene simulation object will allow us to incorporate those models into IRIMAGE. However, our initial research interests coupled with the general need for a basic scene generator, makes a two-dimensional background model more practical in this initial stage of IRIMAGE.

The Scene simulation object models the background imaging scene by compositing various temperature and emissivity "images" onto a two-dimensional form of the VGO. Since the imaging VGO is what defines the background scene, it must be placed properly so that it can be "seen" by the imaging system. Therefore, the user must specify its size, location and orientation in WCS as well as the number of grid points along the two orthogonal axis of the grid. The VGO also needs the default temperature and emissivity values so that no grid point has these values undefined. Once these parameters are set, the imaging VGO is ready to be loaded with data.

The Scene object maintains a database of image data, storing each image in a separate two dimensional VGO. Each image may be generated from the output of a real IR imager, from a 3D synthetic image generation, or using a standard graphics package. IRIMAGE supports 8 bit and 24 bit portable bitmap formats [16] as well as ASCII files with a single column of data. Because the bitmaps are in binary, the user must specify the gain and offset values required to convert each 8 bit number into a temperature, emissivity or matte value. In addition, IRIMAGE treats the red, green and blue channels of the 24 bit bitmap as three inidividual 8 bit images. By doing so, it is possible to store the temperature, emissivity and matte images in a single bitmap. Since ASCII is capable of storing floating point information, the data must already be the actual temperature, emissivity or matte values. Once these images have been loaded, the Scene object can associate them with the VIO's.

The processing, placement and layout order of the loaded images are controlled by the database of VIOs and the Layout program object. For each frame in the simulation, the Scene object accesses the Layout object to generate the ordered list of VIOs to load onto the imaging VGO. Each VIO determines the image processing parameters using the current frame. Once the parameters are defined, the image data linked to the VIO is manipulated, loaded onto the imaging VGO and then returned to normal. Matte images are loaded into the imaging VGO's matte channel. Temperature and emissivity image data is composited with the imaging VGO's corresponding image data using the matte channel. To prevent the matte from effecting more than one set of image data, the Scene object resets the matte channel after each composite. Once all of the images are loaded on the imaging VGO for the current frame, the Scene object is ready for the ray trace.

The Scene object intercepts each Ray object with the imaging VGO to determine the background temperature and emissivity for the corresponding ray. Since IRIMAGE does not allow any VGO to change its position, size or orientation between frames, the coordinates of the interception point between the ray and imaging VGO remain the same throughout the course of the simulation. By storing the interception coordinate in the Ray object instead of the temperature and emissivity values, each Ray only needs to compute this interception at the beginning of the simulation. Consequently, the Ray may retrieve the background temperature and emissivity for any frame by submitting the interception coordinate to the Scene object which will quickly retrieve the values from the imaging VGO. Although the calculations involved are simple, the shear number of rays generated per frame make any reduction in computation beneficial.

2.4.2 The Atmosphere Object

The Atmosphere simulation object defines the atmospheric conditions for IRIMAGE and uses the atmospheric modeling code, MODTRAN, to compute the spectral radiance of the background surface and intervening atmosphere. While MODTRAN only provides a one-dimensional layer solution, the Atmosphere object uses the VGO combined with a Geometric Model Database (GMD) object to define the atmospheric conditions three-dimensionally. The one-dimensional MODTRAN profile is then generated using the simple Ray-Grid interception technique. When combined with the surface data from the Scene object, the radiance incident on the optical system can be calculated by MODTRAN.

In the Atmosphere object, the VGO plays an important role in defining the atmospheric conditions. Like the Scene object, the user supplies the origin, orientation, size and grid point spacing of the three-dimensional VGO. Each grid point stores a geometric model index (GMI) and two integer indexes which relate the point to a gas model and an aerosol model. The GMI links an interpolated volumetric data set with each grid point that lies within its bounding surface. The gas model index points to an element of the Gas Model Database which contains an overall temperature and pressure as well as a set of specific concentrations for each of the gaseous constituents modeled by MODTRAN. Similarly, the aerosol model index points to an element of the Aerosol Model Database which stores the aerosol modeling parameters used by MODTRAN to incorporate aerosols into the atmospheric model. Since both of these databases are linked to MODTRAN, they are stored in the MODTRAN program object, a program object contained inside the Atmosphere simulation object. The VGO does not need to cover the entire region between the imaging system and the Scene object. Instead, it is only required in the region which needs to model the atmosphere using volumetric data. The rest of the space is defined using the default gas and aerosol model indexes.

The Geometric Model Database program object stores and manipulates several databases of geometric models which are based on the GridObj program object. Each database in the GMD stores all of the geometric models associated with one of four standard bounding surface types: a box, a cylinder, a cone and an ellipse. The user defines each geometric model and the gas and aerosol indexes associated with it. Once the individual databases are loaded, the GMD can load the geometric models onto the Atmosphere VGO. Like the Scene object loaded the VIO's, the order of how the GMs are placed on the Atmosphere VGO is very important. Therefore, the GMD uses the same Layout program object used by the Scene object to define the layout order of the geometric models. The Layout Object stores each geomtric model's primitive surface type as well as its index in the primitive surface database. This avoids confusing two different primitive types which have the same index in their respective databases. In addition, more complicated geometric models that represents a specific set of atmospheric conditions can be generated through the effective use of the Layout object. For each frame of the simulation, the GMD uses the Layout object to select which GMs to interpolate onto the Atmosphere VGO. Each GM is created in its own coordinate system and then transformed to the WCS using its own key frame database. Once the GM is defined in the WCS, it is transformed into the Atmosphere VGO's GCS and interpolated on the VGO using the GridObj interpolation routine.

After the geometric models are interpolated, the Ray-Grid intersection algorithm uses the grid point indexes in conjunction with the gas and aerosol databases to compute the atmospheric profile. This profile is passed onto the MODTRAN object to compute the spectral radiance. Since the Atmosphere VGO does not need to occupy the entire space between the optics and the background surface, the Ray-Grid interpolation requires three major steps. First, it determines the distance the Ray travels in the atmosphere between the imaging system and the Grid. If this distance is greater than zero, it stores the distance and the grid point index $(-1, -1, -1)$ which tells the ray to use the default atmospheric model indexes. Second, the Ray intersects each grid box along its path storing the grid point index of the grid box (not the atmospheric model indexes) as well as the distance the ray travels through the box. As we show two-dimensionally in figure 2.11, the intersection technique represents each grid box using a series of parallel planes that lie along the three major axis of the GCS. Each plane is separated by the grid spacing along each axis. Once the entry point on the first plane, i.e., entry point of the grid, is found, the method checks the three orthogonal planes nearest this point along the direction vector of the ray. It selects the next point by finding which of the next set of planes intersect the ray closest to the first point. The path length through the grid box is simply the distance between these two points and the grid point index is determined by the VGO using the mid point between the two intersection points. In the third step, if the ray has not struck the Scene object's imaging plane before it exits the Atmosphere VGO, the ray again stores the $(-1, -1, -1)$ grid point index and computes the distance between the Atmosphere VGO exit point and the interception point with the Scene's imaging VGO. Figure 2.12 demonstrates these three steps.

Since the Ray object stores the grid point indexes and path lengths in the order that each grid box was struct, the Atmosphere object can build the atmospheric

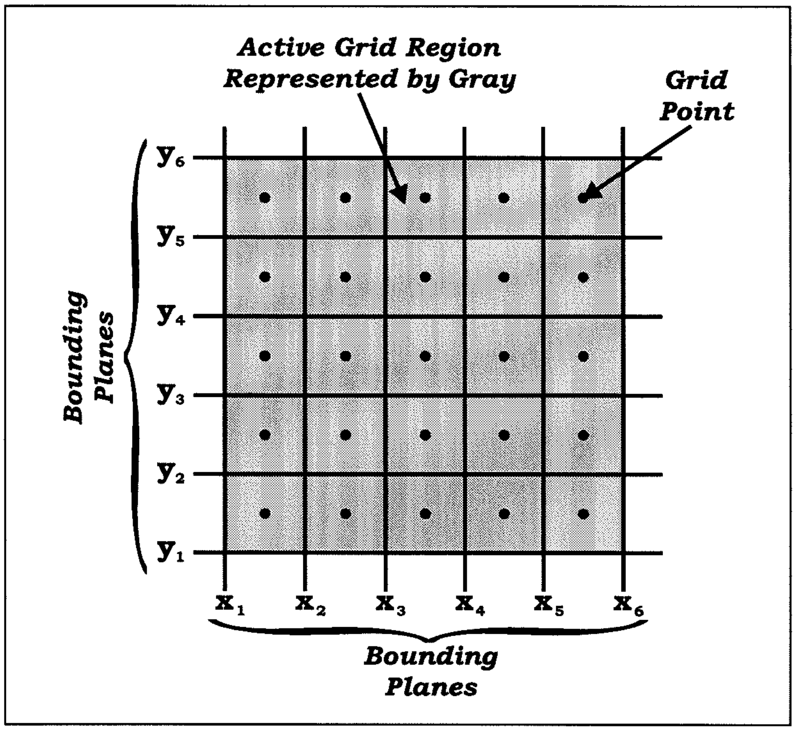


Figure 2.11: 2D representation of VGO construction using planes.

profile using this stack of parameters. The process of building a profile consists of going through the stack of grid point indexes and accessing the Atmospheric VGO to get the atmospheric model indexes. When consecutive grid boxes have the same gas and aerosol indexes, they are combined into a single layer and their path lengths are added together. Using this method, the atmospheric profile along any ray is built and passed on to MODTRAN.

The Ray-Grid interception technique does not store any atmospheric model indexes in the Ray object. Instead, these are retreived from the Atmosphere VGO when it builds the atmospheric profile. If a geometric model changed in anyway between two frames, it is reinterpolated on the VGO. However, this only changes atmospheric model index values that are stored in the Atmosphere VGO. Consequently, the atmospheric profile would change, but the grid point indexes and path lengths that build the profile would remain unchanged. Therefore, as long as the Atmosphere VGO remains stationary in space, IRIMAGE only needs to compute the Ray-Grid interceptions once which significantly cuts down on the number of computations between

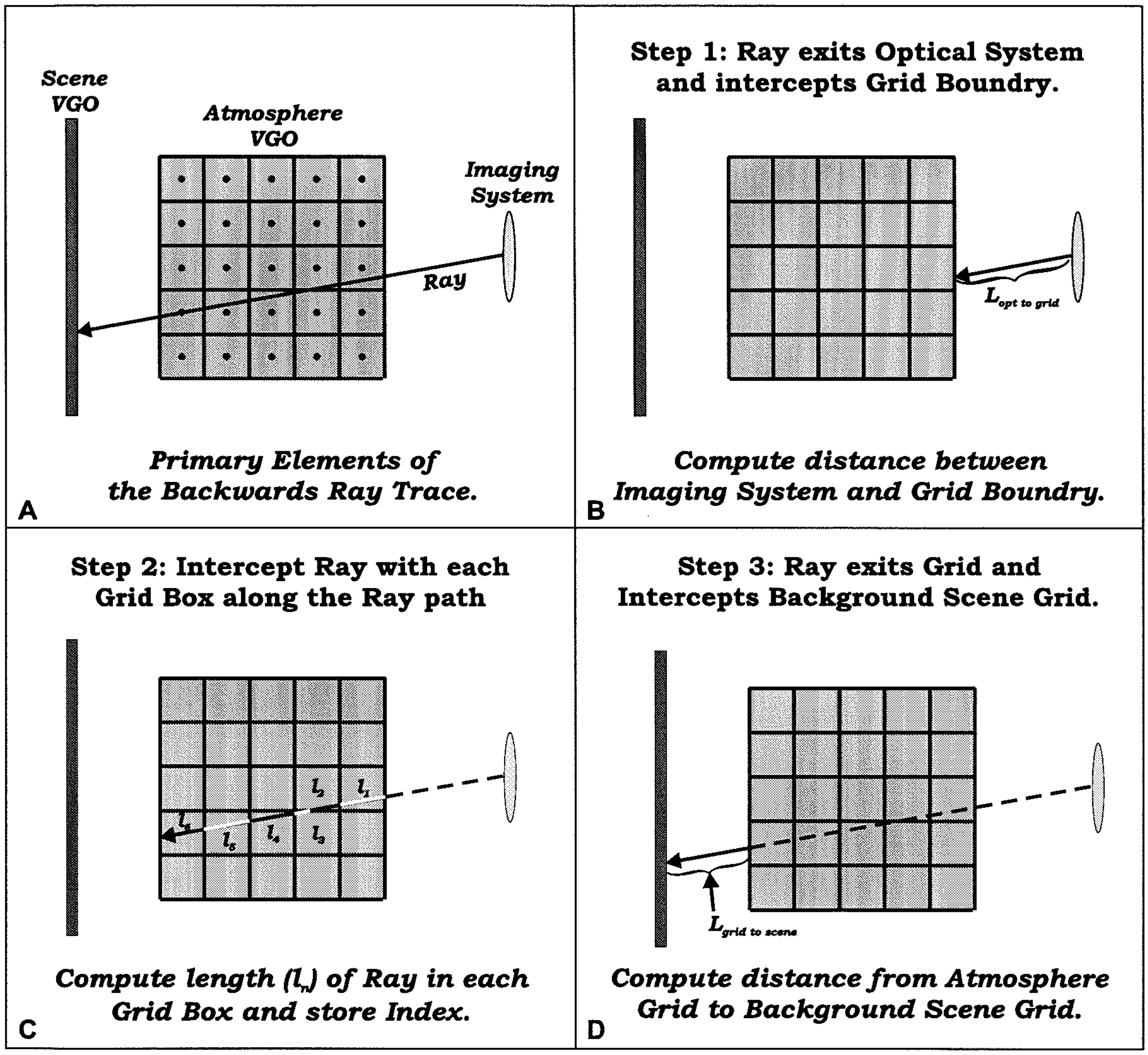

Scene VGO
Atmosphere VGO
Imaging System
Ray
Primary Elements of the Backwards Ray Trace.
A
Step 1: Ray exits Optical System and intercepts Grid Boundry.
$L_{opt\ to\ grid}$
Compute distance between Imaging System and Grid Boundry.
B
Step 2: Intercept Ray with each Grid Box along the Ray path
l_1 l_2 l_3 l_4 l_5 l_6
Compute length (l_n) of Ray in each Grid Box and store Index.
C
Step 3: Ray exits Grid and Intercepts Background Scene Grid.
$L_{grid\ to\ scene}$
Compute distance from Atmosphere Grid to Background Scene Grid.
D

Figure 2.12: Two-dimensional representation of a ray intercepting a three-dimensional atmospheric VGO.

frames.

The MODTRAN program object defines the data structures and routines that set up the atmospheric profile for each ray and then passes the profile to the MODTRAN model which computes and returns the spectral radiance for the ray. This object loads the general parameters that define the MODTRAN model as well as the individual gas and aerosol models used by the geometric models stored in the GMD. It also processes the atmospheric profile generated by the Atmosphere object and returns a MODTRAN compliant profile. The MODTRAN object acts as the connection point between the C++ code of the Atmosphere object and the FORTRAN code that defines MODTRAN. Using this link, the Atmosphere object passes a complete profile to the MODTRAN object which passes it through a set of linking routines to the MODTRAN code. After performing the computation, the MODTRAN code returns the spectral radiance back through the linking routines to a spectral radiance array which is passed onto the imaging simulation objects in IRIMAGE.

Like the Scene Object, the Atmosphere Object is designed to easily incorporate other atmospheric modeling codes. For this reason, all of the data structures and routines related directly to MODTRAN are stored in the MODTRAN program object. This separates MODTRAN from the rest of the Atmospheric simulation object. However, many of the routines in the Atmospheric object are still based heavily on the MODTRAN methodology. Therefore, new atmospheric codes must conform to the MODTRAN methodology or the Atmospheric object will need to be adjusted to conform to the new methodology without changing how data is passed into and out of the Atmosphere simulation object.

2.4.3 The Focal Plane Array (FPA) Object

The Focal Plane Array simulation object drives the entire IRIMAGE simulation. Once all of the major simulation objects load their parameters and the Scene and Atmosphere objects are initialized, the FPA object lays out a two-dimensional VGO where each grid point represents a stack of one or more detectors. Once the VGO is setup, the FPA generates a single Ray object or set of Ray objects for each detector stack. These rays are passed on to the Optics, Scene and Atmosphere simulation objects which return the spectral radiance of the background scene and intervening atmosphere. Once the radiance incident on the detector stack is returned, the FPA passes it to the Detector simulation object which computes the output value for a given detector. The output value from the Detector object is finally passed to the Output simulation object which places each detector's output into a data file. All of this interaction with the other simulation objects makes the FPA the only simulation object which is heavily dependent on the structure of these other objects.

The VGO, combined with a unit cell approach, places detectors in a regular fashion onto the focal plane. Like the other two simulation objects that use the VGO object, the user defines the location and orientation of the FPA grid with respect to the WCS. In this case, the origin and orientation are usually centered on the optical axis with the two-dimensional grid perpendicular to the axis. The number of grid points is specified by the number of detectors along the X,Y directions and the spacing is defined by the detector pitch in both directions. The UnitCell program object "tiles" a specific pattern of *detection elements* onto the FPA's VGO. The user defines a detection element by linking it to a specific set of detectors stored in the Detector object's database. Each detection element represents a detector stack by storing the type and index value for each detector in the stack. In addition, it stores the spectral range and bandwidth of the stack. Once a detection element is defined, it is assigned a unique index and stored in the UnitCell object. The pattern defined by the UnitCell object can either be a set of completely different detection elements or a few detection elements in a specific repeated order. Using the pattern of indexes as a guide, the UnitCell object associates each grid point on the VGO with a specific detection element. In figure 2.13, we show an example of a four detector unit cell as it is tiled across the surface of the focal plane grid to form an 8x8 FPA. When the FPA object needs the detectors or information for a particular grid point, it uses the detection element's index to access the information from the UnitCell object. Therefore, the VGO only stores a single integer index value, which significantly reduces the

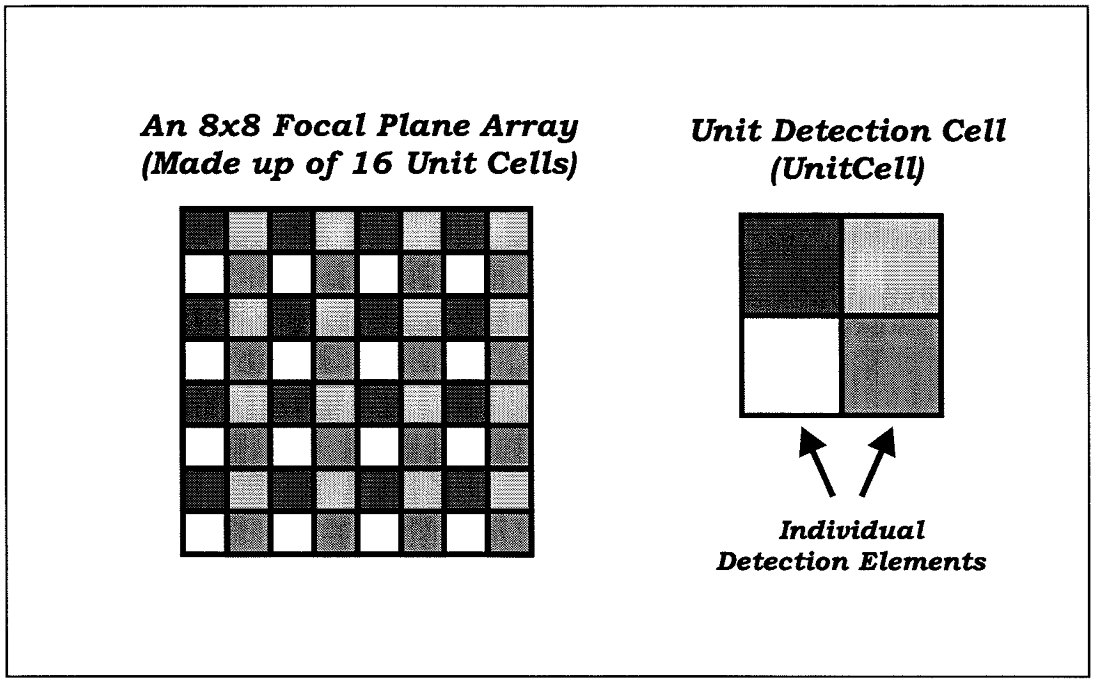


Figure 2.13: Example of an 8x8 FPA using a 2x2 Unit Cell.

amount of space required to store a pattern of detectors.

In addition to detector layout, the FPA object also initializes and maintains the Ray objects associated with each grid point of the VGO. The user defines the *detection area* as a specific percentage of the product of the rectangular area that surrounds each grid point. In addition, the user may select either the single ray or sub-mesh approach to assign Ray objects to each detection area. For the single ray method, the FPA object places each Ray object's origin at the center of the *detection area* which is just the associated grid point location in the WCS. The mesh method divides the detection area into an $n \times m$ array of rectangles where n and m are the user defined number of rectangles along the X and Y directions. The origin of each Ray object is placed at the center point of each smaller rectangle. Once the Ray objects are created, the FPA object uses the back nodal point of the optics with each Ray object's origin to compute the initial direction vector of the ray. The two approaches converge after the Ray objects are defined.

The FPA object shepherds the Ray objects through the Optics, Atmosphere and Scene simulation objects to generate the spectral radiance. Prior to computing any frames, the FPA passes the rays to the Optics object where the the Ray object origin is replaced by the exit point from the external surface of the optics while the direction vector remains the same. Each Ray is returned by the Optics to the FPA where it is handed off to the Scene object to determine the interception point with background VGO and to the Atmosphere object to build its Ray-VGO interception database. After all of the rays are processed, IRIMAGE proceeds to compute a single frame or an entire animation. During each frame, the FPA object passes the rays to the Atmosphere object to generate the atmospheric profile along the ray and uses the Scene interception point to set the background surface temperature and emissivity for the profile. Before passing the final atmospheric profile back to the Atmosphere object to be processed by MODTRAN, the FPA accesses the Ray object's associated detection element in the UnitCell program object to get the spectral range and bandwidth for the profile. This final set of parameters define the active spectral region of the detectors in the stack. Once the profile is completely generated, it is passed to MODTRAN, via the Atmosphere object, which returns the spectral radiance of that profile. Finally, the FPA object passes the spectral radiance of the Ray object back to the Optics object where the transmission curves are used to adjust the spectral radiance. The attenuated spectral radiance is returned to the FPA to be converted to the spectral power incident upon the detection element.

At this point, the two approaches briefly diverge again to compute the incident spectral power. The single ray method calculates the incident spectral power using the spectral radiance of the ray and the area of the entire detection element. The mesh method also uses the spectral radiance of each ray, but the area is reduced to the area of the rectangle. The mesh method then sums the spectral power incident on each of the rectangles to generate the total incident spectral power on the detection element. The actual computation is the same in both cases, but the mesh method accounts for the multiple rays involved due to the subdivision of the detection element.

Once the incident spectral power has been computed for a detection element, the output signal for the detectors of that element is calculated and stored in the output file. Using the detection element index, the FPA object accesses the detector types and ID indexes and gives them, along with the element's incident spectral power, to the Detector simulation object. It uses the type and index values to access the correct routines and parameters for each detector. Once the right detector is found, Detector object converts the incident spectral power into an output signal as well as computing the background fluctuation noise. The final output values are returned to the FPA object which submits it to the Output object for storing in an ASCII data file.

From the discussion above, one can see how the FPA object drives the simulation. Although very little computation is done by the FPA, it acts as the main conduit for information between the other five simulation objects. Since multiple FPA's might be incorporated into a single imaging system, the Optics and Output simulation objects are contained within the FPA object and not as separate objects. By directly linking them to the FPA, IRIMAGE can model multiple FPA's in a single imaging system without needing to use the same Optics and Output object for all of the FPA's. Such strong links between the FPA and the other objects makes it difficult to alter another object without affecting the FPA. However, as long as the changes to any of the other objects do not effect how data is passed or stored, the FPA object should not require any changes.

2.4.4 The Optics Object

Like the Scene and Atmosphere simulation objects, the design of the Optics simulation facilitates adding a full featured ray tracer to IRIMAGE. However, for many cases the simple pin-hole approximation is adequate. Therefore, the Optics object currently only deals with the pin-hole approximation. Under this approximation, the Optics object performs three functions. It stores the basic optical parameters that define the pin-hole approximation. Using these parameters it manipulates Ray objects so that each ray correctly exits the optical system so that it may be interpolated by the Scene and Atmosphere VGO's. Finally, it stores a set of transmission curves that approximate the spectral filtering of a real optical system. These curves are used to attenuate the spectral radiance returned by each ray after the MODTRAN calculation. These three functions define a reasonably simple and accurate optical system.

The Optics simulation object stores several user-defined parameters that define a basic optical system. The cardinal points of the optical system define the focal length and location of the nodal points along the optical axis. Since the WCS should be defined at the front surface of the optical system, all of these coordinates are defined with respect to the WCS origin and orientation. Figure 2.14a shows an optical system with the locations of its cardinal points defined. According to figure 2.14a, the coordinate of the 2^{nd} focal point must coincide with the origin of the FPA object's VGO. The focal length used by the Optics object is the *effective focal length* which is the distance from the principle point to the corresponding focal point. The Optics object requires the user to define each cardinal point in terms of its WCS coordinate along the optical axis. This ensures that the Optics object can then correctly compute the focal length. Since the optical system is assumed to be optically thick, the user must also provide the location of the front and back surfaces along the optical axis. Note that the front surface should be the origin of the WCS, i.e., $(0, 0, 0)$. Another parameter to be specified by the user is the optical F-number (F/#). The F/# relates the diameter of the optical aperture to the focal length ($F/\# = D/f$). Although most of these parameters are only used by the Optics object, the FPA object extracts the focal length, F/# and back nodal point. These parameters are used to setup the Ray object and convert the spectral radiance to incident spectral power.

Under the pin-hole approximation, ray tracing through the optical system becomes a simple change of origin calculation. As mentioned in section 2.2.4, the nodal points define the points of the optical system where an incident ray striking one nodal point will appear to exit the other with the same direction vector. Since the FPA simulation object sets the direction vector of each Ray object to point from the origin of the ray on the detection element toward the 2^{nd} nodal point of the optical system, only the Ray object's origin needs to be adjusted by the Optics object. As one can see from figure 2.14b, a ray in an optically thick system will not exit the optical system at the

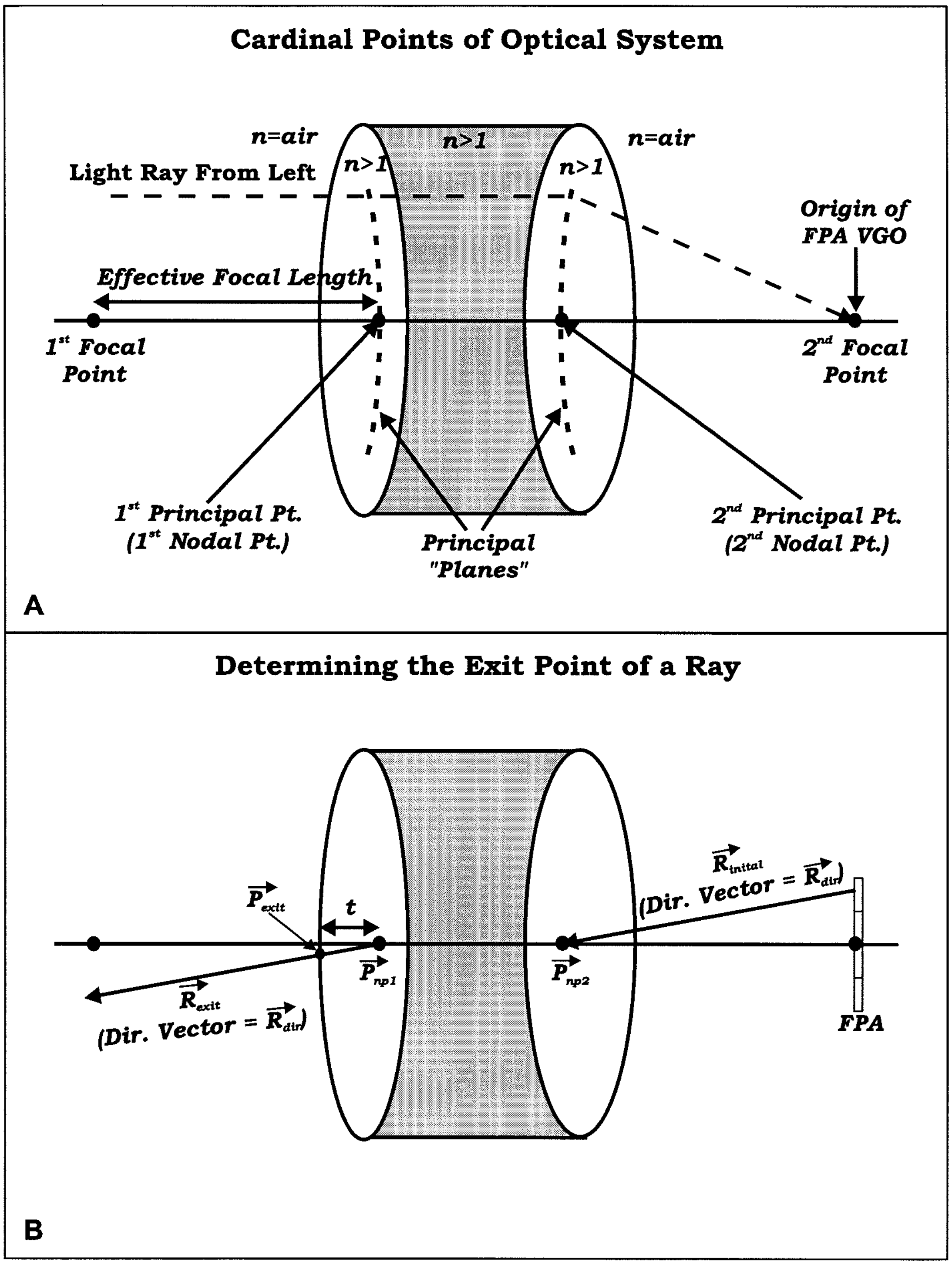


Figure 2.14: (a)The cardinal points of an optical system: (i) focal points, (ii) principle points, (iii) nodal points. (b)Example of a ray passing through the nodal points of an optically thick system.

1st nodal point, but at the point it passes through the front surface of the optical system. Technically, the ray should be refracted at this interface, but the definition of the nodal point already takes this into account. Therefore, the exit point, $\vec{P}_{exit}$, can be approximated using the equation (2.4.1) in conjunction with the distance from the 1st nodal point to the front surface, t, the direction vector of the ray, $\vec{R}_{dir}$, and the coordinates of the 1st nodal point, $\vec{P}_{np1}$:

$$\vec{R}_{org} = \vec{P}_{exit} = \vec{P}_{np1} + t\vec{R}_{dir} \tag{2.4.1}$$

This equation assumes that within the paraxial limit, t is approximately the same between the front nodal point and any point on the front surface of the optical system. Since IRIMAGE is only interested in the portion of the Ray object that travels between the optical system and the background scene, the Optics object replaces the Ray's origin with the exit point, $\vec{P}_{exit}$.

Once MODTRAN calculates the spectral radiance for a ray, the Optics object attenuates the incident radiance using its optical transmission curves. Although most optical systems will only need one transmission curve to simulate the filtering of the system, IRIMAGE does provide for cases of more than one filter through the use of multiple transmission curves. As mentioned in section 2.2.4, each curve consists of an array of spectral values and associated transmission coefficients which form a piecewise linear curve. The Optics object linearly interpolates between any two points on this curve to get the transmission coefficient for a specific wave number. Once all of the transmission coefficients for a particular wave number are determined, they are multiplied together to form a single transmission coefficient. Finally, the spectral radiance is multiplied by this transmission coefficient to generate the attenuated radiance at the given wave number. This process continues for every member of the spectral radiance array. Once the entire spectral radiance array has been adjusted, it is passed back to the FPA object.

2.4.5 The Detector Object

Four types of detectors are modeled inside the Detector simulation object: photoconductor, photovoltaic, pyrometer, bolometer. The object-oriented nature of the base objects used by the Detector object allows all four of the modeled detector types to use the same core routines for loading detector parameters and computing the output signal using a responsivity curve. The Detector object maintains a database for each detector type. Using the detector type and indentifying index value, the database passes information back and forth between individual detectors and the FPA object. This type of organization makes it easy to add detectors to IRIMAGE without altering any of the other simulation objects.

Before loading the parameters for a specific detector, Detector object must first create an entry for it in one of the four detector databases. Once these databases are defined, the Detector object cycles through each entry in its parameter file and loads the proper detector parameters. When the FPA object passes the spectral power along with a detector's type and index number, the Detector object uses the type and index to direct the spectral power array to the proper detector. After it passes the spectral power to a specific detector, the detector generates the OutVal data structure which stores all of the data to be output by the detector. The detector loads the incident spectral power in several forms as well as the signal, noise and total output from the detector. Once this data has been computed and stored, it is returned to the FPA object.

All four of the current detector models are based on the DetType program object. DetType is a generic detector object which defines a set of routines to load a set of standard detector parameters and compute the detector's output signal using a piecewise linear responsivity curve. DetType loads all of the common parameters shared by most major detectors such as the quantum efficiency, fill factor, and integration time as well as either the detector's responsivity curve or its D* curve and the data required to convert it to a responsivity curve. Besides these detector specific parameters, the DetType object needs the effective area, the spectral range and bandwidth. In

IRIMAGE, these parameters are transfered by the FPA object along with spectral power array.

Since the background fluctuation model defined by equation (2.2.31) must use the average number of photons incident upon the detector over each spectral bandwidth, the spectral power must be put in these terms. For the case of MODTRAN, the spectral power provided to the Detector object is in the form of Watts/cm^{-1}. Equation (2.4.2) demonstrates how DetType converts this spectral power at a specific wave number and bandwidth, $P(\nu)\Delta\nu$, to the number of incident photons at the same wave number and bandwidth, $N(\nu)\Delta\nu$, for a specific integration time, t_{int}:

$$N(\nu)\Delta\nu = P(\nu)\frac{10000}{h\nu}t_{int}\Delta\nu \tag{2.4.2}$$

In addition to the spectral power, the responsivity must also be converted to the form of *output/photon*. Fortunately, the DetType is flexible enough to provide several acceptable units for the responsivity curve. As long as the user specifies the unit type, via a simple integer flag, DetType will convert the responsivity to the necessary form. Once the conversions have been made, the DetType program computes the output signal using equation (2.4.3):

$$V_s = \sum_{\nu=\nu_{min}}^{\nu_{max}} R(\nu)N(\nu)\Delta\nu \tag{2.4.3}$$

DetType generates the output noise, V_n, using the Background Fluctuation Noise model defined in section 2.2.6. When these two calculations are complete, DetType returns the OutVal data structure with these two detector outputs as well as the total output and the total detectable power.

Flexibility is the most important thing to understand about the DetType program object. Although most detector types can use the standard DetType routines to load data and compute the detector output, the object oriented design allows any detector type to use its own routines. These routines may load more specific data, process the computed output, or even compute the detector output using another algorithm entirely. Two of the four detector types in IRIMAGE utilize this property. For instance, the photoconductor detector incorporates the generation-recombination (g-r) rate by processing the computed output. A separate routine multiplies this g-r rate times the signal, noise and total output. Since the g-r rate is not loaded by the standard DetType routine, the photoconductor also uses a separate routine to load the g-r rate. Such modifications do not effect any other simulation objects outside of the Detector object which simplifies adding new detector types, or improving on existing ones.

2.4.6 The Output Object

Once the detector output is computed, the FPA simulation object passes the resultant OutVal structure to the Output simulation object for storing in an ASCII file. Like the Optics object, the Output object is contained inside the FPA object so that each FPA in a simulation outputs its data to a separate set of files. Since many simulations may be animated, each frame of the animation is stored as an individual file. For this reason, each file's name is made up of a user defined header name and the frame number. The output data from each detector of the FPA is stored as a single record in this file. For instance, a 60 frame simulation using a single detector in an 8x8 FPA will have 64 records per file in 60 files. The user has the choice to output any combination of the following data values stored by a single detector in the OutVal structure:

- Detector type.
- X,Y coordinates for the center of the detection element in the GCS.
- Incident power in watts and/or photons/sec.
- Signal, noise and/or total output voltage/current.

In addition, the user can specify whether the incident power and detector output are stored as their actual value or the Log_{10} of their value. This approach allows the user to store those data values which are important to the current simulation and discard the rest (saving disk space). Although the Output object currently outputs only an ASCII file, the object-oriented design allows the detector data to be output in any form. A future version of IRIMAGE could output images in a standard graphics format (GIF, TGA, etc.) or even directly to a graphics window.

2.5 The IRIMAGE Simulation

With the introduction of the six simulation objects of IRIMAGE, we can address how the simulator works to produce images. We begin with a brief discussion of the Graphical User Interface (GUI). The GUI allows the user to define the parameter files for each of the six simulation objects using a set of intuitive windows and dialog boxes. Following the GUI discussion, we describe how the IRIMAGE generates an image from these parameter files. Finally, we present a couple of test results from the simulator.

2.5.1 The Graphical User Interface (GUI)

In IRIMAGE, all of the settings for each simulation object are loaded from the object's parameter file. In addition to the six parameter files loaded by the simulation objects of IRIMAGE, there is a seventh general parameter file. This file stores the file names and locations of the six other parameter files as well as the general animation parameters including the starting frame and how many frames to compute. Since many of the simulation object's may load multiple incarnations of the same simulation object or load information for other supporting program objects, the parameter files are divided up into sub-sections which define the parameters for a specific part of a simulation object. As an example, the GMD has its own sub-section of the Atmosphere object's parameter file while each geometric model stored in the GMD also has a sub-section of its own. Such organization makes it possible to add new geometric models or new detector types without invalidating old parameter files. Furthermore, the parameter files utilize English descriptions making it possible to print out each file to see how a particular simulation works. Despite these advantages, the parameter files can be very long and involve hundreds of parameters.

Prior to the development of the GUI, the only way to set or change the values in a parameter file was to use a standard editor like emacs or vi. The GUI reduces the problems encountered when trying to write or edit scripts by hand. We maintain the flexibility of the parameter file method by using the GUI to generate the files instead of as a full front-end for IRIMAGE. Since the time to run a simulation may vary from a few minutes to several days, keeping this methodology makes it possible to run a simulation in the background without the GUI having to remain open. The GUI provides a more intuitive interface for the user to load information while making sure that the parameters are also formatted correctly and placed in the parameter file in the proper order. In addition, many of the parameters have default values and the GUI automatically sets them if they are not set by the user. Finally, the GUI can use old parameter files allowing the user to make small changes in a simulation or define a series of simulations that share many of the same parameters. Overall, the GUI provides the user with an effective way of loading parameters into IRIMAGE.

2.5.2 How IRIMAGE Computes An Image

Once the user defines the parameters for a simulation, IRIMAGE proceeds to compute the sequence of images desired. To begin, IRIMAGE loads the various parameter files and initializes all of the simulation objects. Once the parameters are loaded for the Scene and Atmosphere simulation objects, IRIMAGE initializes each of their VGOs. Subsequently, it loads the Image, VIO and Layout databases into the Scene object followed by the GMD, the standard MODTRAN profile and the various gas and aerosol models into the Atmosphere object. Prior to initializing the imaging system objects (FPA, Optics, and Output), IRIMAGE sets up the Detector database. After the FPA simulation object's parameters are loaded, the corresponding Optics and Output simulation objects are read in by IRIMAGE. Once the entire imaging system parameters have been set, the FPA object creates the layout by assigning a detection element to each grid point of the FPA's VGO using the UnitCell program object as a guide. As soon as the simulator is initialized, the simulator is prepared to generate the rays and compute each image.

The first step in creating an image, IRIMAGE must generate the Ray object(s) associated with each detection element of the FPA. Once the rays are all created, the FPA object transfers them to the Optics object to generate the exit coordinate and direction vector. Using each modified Ray object, the Scene object intercepts the ray with the imaging VGO and the Atmosphere object generates the database of grid point indexes by intercepting the ray with the Atmosphere VGO. As discussed, these calculations only need to be computed at the beginning of a simulation. For an animation, this property significantly reduces the work done between successive frames and improves the speed of the overall simulation. While the amount of memory required to store the ray's vector data may be small, the database of grid point indexes and path lengths requires much more memory. When one considers an FPA with a large number of detection elements or a large number of rays per detection element, the memory requirements grow quite large. Therefore, IRIMAGE has two different memory models to suit various machines running IRIMAGE. If one has sufficient memory to store all of the rays including each grid point database, the user can specify that IRIMAGE use the *Large* memory model. Otherwise, IRIMAGE must use the *Small* memory model which preserves the vector data for the entire simulation but rebuilds each atmospheric grid database just prior to determining the atmospheric profile and destroys the database as soon as the detector output has been returned by the Detector object. Both models are built into the simulator so that the user specifies the model at run time. Once the memory model is chosen and the rays are constructed and associated with the detection elements of the FPA object, IRIMAGE can compute each frame.

Even though each detection element may contain a different detector type or set of detectors, computing the output for each individual detector remains much the same. The procedure for generating a frame is outlined below as a series of simple steps:

1. Load VIO's for current frame onto the Scene's imaging VGO.

2. Interpolate Geometric Models for current frame onto the Atmosphere VGO.

3. Open output file for current frame.

4. Initialize the FPA object's variables.

5. Compute the output of each detector using FPA Object.

   (a) Locate current detection element.

   (b) *For the small memory model, intercept ray or rays with Atmosphere VGO.*

   (c) Atmosphere object computes atmospheric profile using Ray object.

   (d) Scene object determines background temperature/emissivity for Ray object.

   (e) Use detection element to assign spectral range to profile.

   (f) Pass profile to Atmospheric Object which passes it on to MODTRAN.

   (g) Using MODTRAN, compute the spectral radiance and return it to Atmosphere object.

   (h) Optics object adjusts spectral radiance using the spectral transmission curve.

   (i) FPA object uses detection element information to convert spectral radiance to the spectral power collected by the imaging system for the current Ray object.

   (j) Store or add the spectral power to the total incident power of the detection element.

   (k) For mesh sizes greater than 1, steps 5d through 5j are repeated until all Ray objects associated with the current detection element are computed.

   (l) *For the small memory model, delete Ray object's grid point database.*

   (m) Pass detection element's total incident power to it's associated detector(s).

(n) Use incident power with the detector responsivity curve and the noise model to generate the output signal and noise.

(o) Detector object returns the OutVal data structure which includes the signal, noise and total output.

(p) Pass OutVal structure to Output object which stores the output values, incident power and coordinates in the current frame's output file.

6. Repeat step 5 for each element of the FPA.

7. Close the output file.

8. Cycle to next frame and repeat the entire process until all frames are generated.

As one can see, the process of computing each frame is fairly straightforward. It is also clear that the FPA really drives the entire simulator. It acts as an information server which passes information to be processed between each of the various objects. The object oriented nature of the entire calculation clearly shows how other steps can be easily incorporated in this procedure.

2.5.3 Two Test Cases

In order to judge the effectiveness of the methods employed by IRIMAGE, we conducted a series of test cases. These cases serve to test both the inner workings of the simulation (such as testing the capabilities of the VIO's and the geometric models) and provide some insight into some of the applications of infrared imaging we intend to address with IRIMAGE in the future. The two cases addressed here are:

1. Simulating passive imaging of a rotating ellipsoid of high concentration, carbon monoxide (CO) at high temperature.

2. Simulating passive imaging of a car moving across a varying temperature background with a cone of carbon monoxide exiting the tail pipe.

Parameter	***Value***
FPA Size	256 by 256
Detector Pitch	38 μm
Detector Type	InSb Photovoltaic
Detectable Range	FWHM=4.610 to 4.776 μm
QEff	65.0%
Integration Time	0.005208 sec
Mean Responsivity	$1.90x10^{-7}$ volts/photon
Focal Length	100 mm
Optical F/#	2.545

Table 2.1: Amber IR camera system parameters for the two test cases.

Although a quantitative evaluation of IRIMAGE is necessary and is currently taking place, these examples rely on the proven validity of MODTRAN as an atmospheric modeling package. Furthermore, we have conducted qualitative experiments similar to the one in case 2 which provide results that are comparable to the simulation.

IRIMAGE Parameters

Before addressing either case, there are a few general similarities between the two. First of all, they both use the same imaging system. Table 2.1 points out the major parameters used by IRIMAGE for the focal plane, detector and optical system. The imaging system parameters are based on the Amber AE-256 camera system which we acquired to perform validation experiments.

In addition to the imaging system, both simulations utilize the similar atmospheric models to simulate an atmosphere with a polluted region. Both simulations are based on the 1976 US Standard atmospheric profile which is part of MODTRAN. The clean atmosphere relies on the gas concentrations of the US standard model and is at 293K and 1 atm. However, the polluted model increases the concentration of carbon monoxide (CO) from the standard concentration of approximately 10 ppm to a much higher concentrations of 1000 ppmv for the ellipsoid and 10,000 ppmv for the car simulation. The polluted model also is at 1 atm, but its temperature is raised above the 293K. Therefore, the two simulations we examine utilize the GMD with the atmospheric VGO to model a hot polluted region surrounded by a clean atmosphere at room temperature.

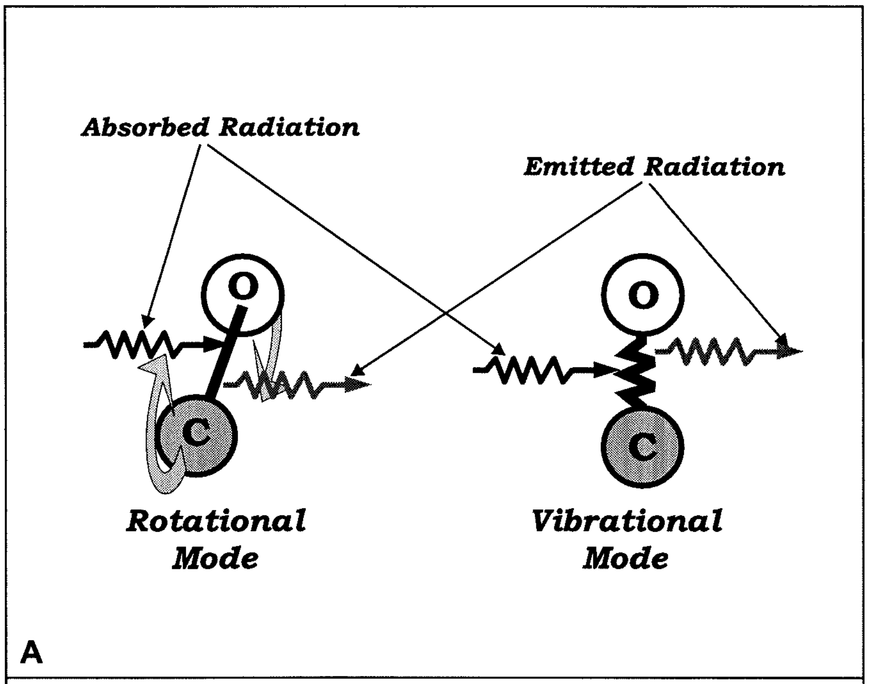
Scientific Motivation for Simulations

Like most gases, CO absorbs and radiates infrared radiation. This property is due to the vibrational and rotational quantum modes in the Carbon Monoxide (CO) gas molecule. The vibrational states place the energy states in the mid-wavelength IR region (MWIR) while the rotational states produce perturbations around these vibrational states. The CO molecule absorbs and radiates energy in the range from 4.4 to 4.9 microns (See figure 2.15). Current methods of detecting CO optically utilize *active* detection of its absorption signature. Using either a blackbody or laser source, a spectrometer sweeps across the active region of CO and measures the reduction in the transmission of the source. However, most sources of CO pollution are hotter than their surroundings. Therefore, we should be able to measure the CO radiation instead of its absorption. For this reason, we conducted the following tests with a heated region of high concentrations of CO.

Case 1: Rotating Ellipsoid of Carbon Monoxide.

Our first test case examines the ability of IRIMAGE to resolve a localized region of high concentrations of CO against a standard background atmosphere and scene. Using a cigar shaped ellipsoid geometric model that is 50 cm long and 25 cm wide along the major and minor axis, we associate a region of space with an elevated temperature of 313K and a concentration of 1000 ppm per volume of CO in addition to the standard concentrations of the other normal atmospheric constituents. We place this ellipsoid rotating about its center point at a point 7m from the imaging system. Outside of this ellipsoidal region, we set the temperature to 293K and utilize the 1976 Standard Model stored in MODTRAN. At a distance of 10m, we simulate the background scene using a wall at 293K and emissivity of .9. The simple background scene prevents mistaking the background for an atmospheric effect and also allows us to evaluate the capabilities of the atmospheric modeling as well as the detector and noise model.

In figure 2.16, we present four images from a single frame of this animation. The first two images span a voltage range from 0.5 to 1 volts, while the second two images span a much smaller range from 0.73 to 0.755 volts. In addition, the images



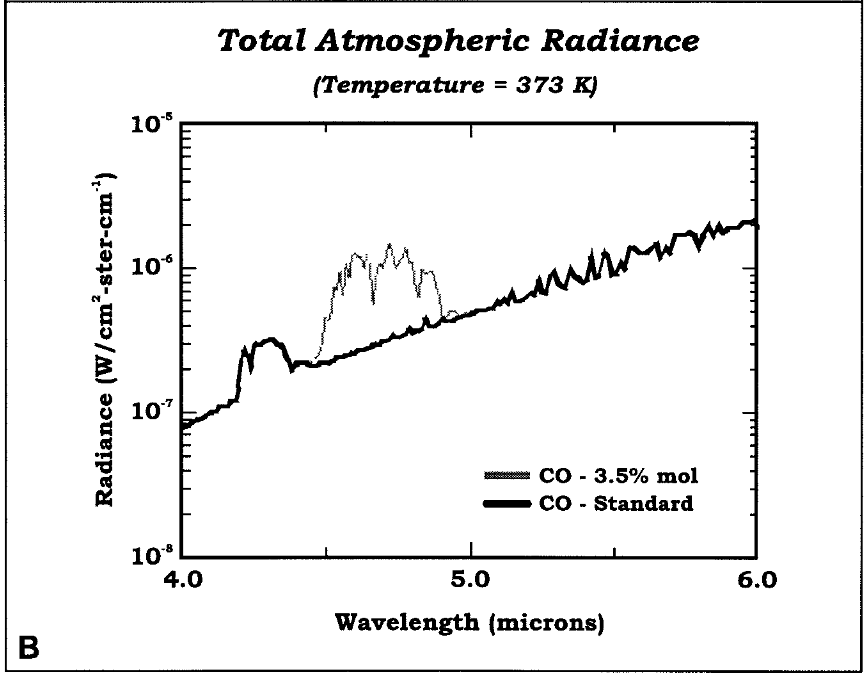


Figure 2.15: Detection of carbon monoxide (CO): (a) Vibrational and rotational states of a non-diatomic molecule can either absorb or emit radiation. (b) Radiance of CO in the 4.0 to 6.0 micron region of the infrared.

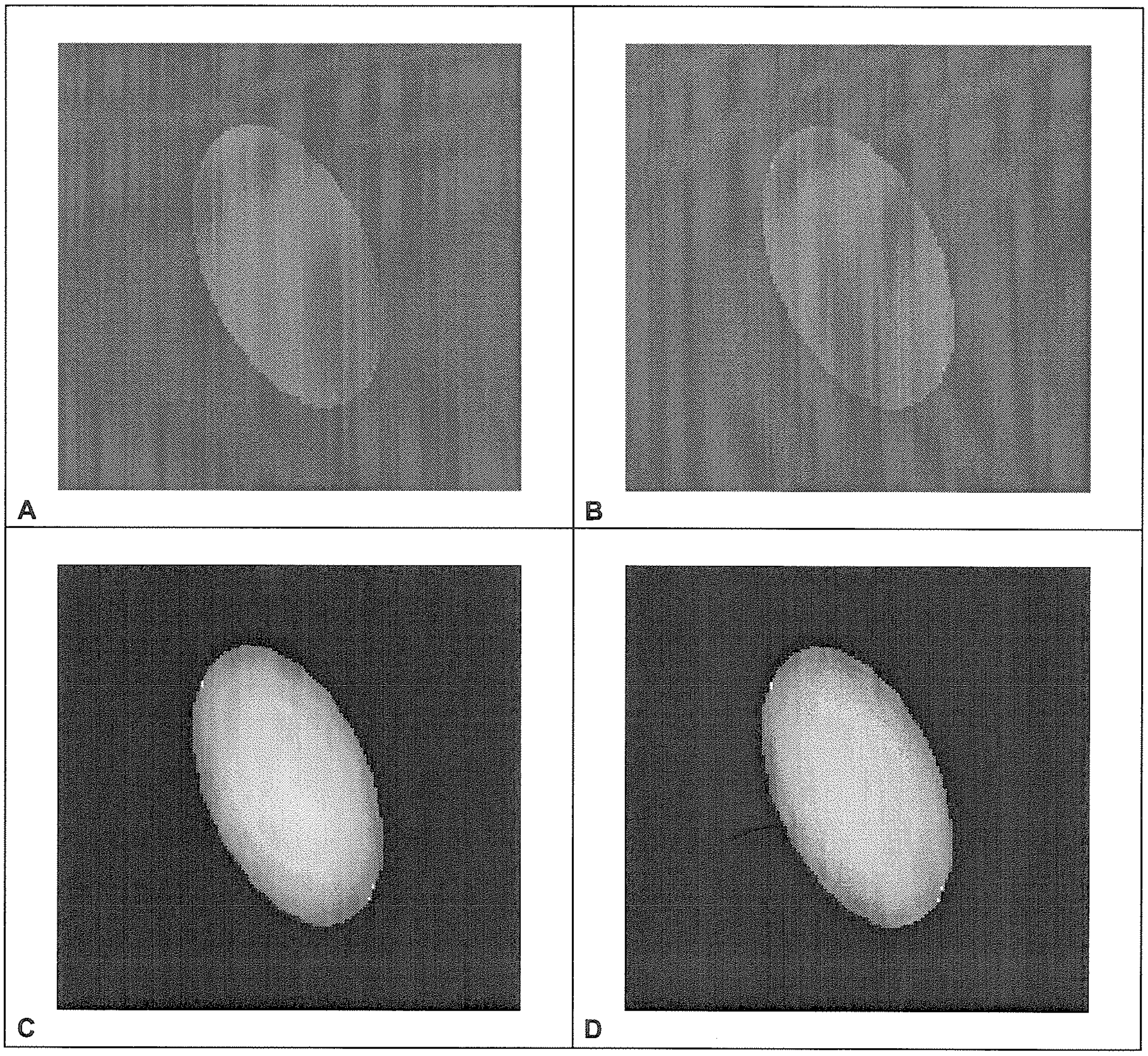


Figure 2.16: Frame 64 of Rotating Ellipsoid of CO in a clean atmosphere. (a) No noise, Voltage range = 0.5 to 1 volt. (b) With noise, Voltage range = 0.5 to 1 volt. (c) No noise, Voltage range = 0.73 to 0.755 volts. (d) With noise, Voltage range = 0.73 to 0.755 volts. Bright spots in (c) and (d) are caused by a path that is too short for MODTRAN to properly compute. They are artifacts of the calculation.

in (b) and (d) include the noise model contribution, while the other two images only represent the signal voltage. All four of these images were generated using the output from a single file and the 2D data imaging capabilities of the AVS imaging software. Comparing the four images we can see that it easier to resolve a change from the standard atmosphere to the ellipsoid region than it is to resolve the varying thickness of the ellipsoidal region. Adjusting the "contrast" by limiting the voltage range improves the ability to resolve the varying thickness (edges are darker than the center). However, when we incorporate the noise, the subtle changes in the thickest part of the ellipsoid become washed out by the noise.

This simulation demonstrates the effectiveness of the atmospheric VGO-GMD approach as well as the detector modeling capabilities of IRIMAGE. Unlike the simulations presented in chapter 2 (see figure B.13), we interpolated the ellipsoid onto a high resolution VGO. This nearly eliminates the "layered" look associated with a coarsely spaced VGO. Instead, the ellipsoid in figure 2.16c gets darker near the edges and brighter toward the center, giving it a true three-dimensional appearance. This is what we expect since the thicker a hot polluted region is, the more it will radiate. Therefore, this simulation clearly shows that the three-dimensional VGO approach provides a reasonable method for representing a gaseous region using a primitive geometry which is connected to a specific atmospheric model. The small voltage range is also consistant with what we expect from a real imaging system. Furthermore, the noise model adds significantly to the realism necessary to make adequate judgments regarding the usefulness of a particular imaging system or application.

Case 2: Car Moving Across a Background.

This test demonstrates the ability of IRIMAGE to simulate a composited background scene in combination with an atmospheric model. Using the same atmospheric models as in case 1, this case incorporates an image of a moving car whose surface temperature ranges from 293K to 305K at a distance of 5m from the imaging system. Although fairly unrealistic, we test this car moving across a background image whose top half consists of a temperature gradient running from 275K to 325K and bottom half is fixed at 273K. We model the car exhaust with a cone of gas at 323K

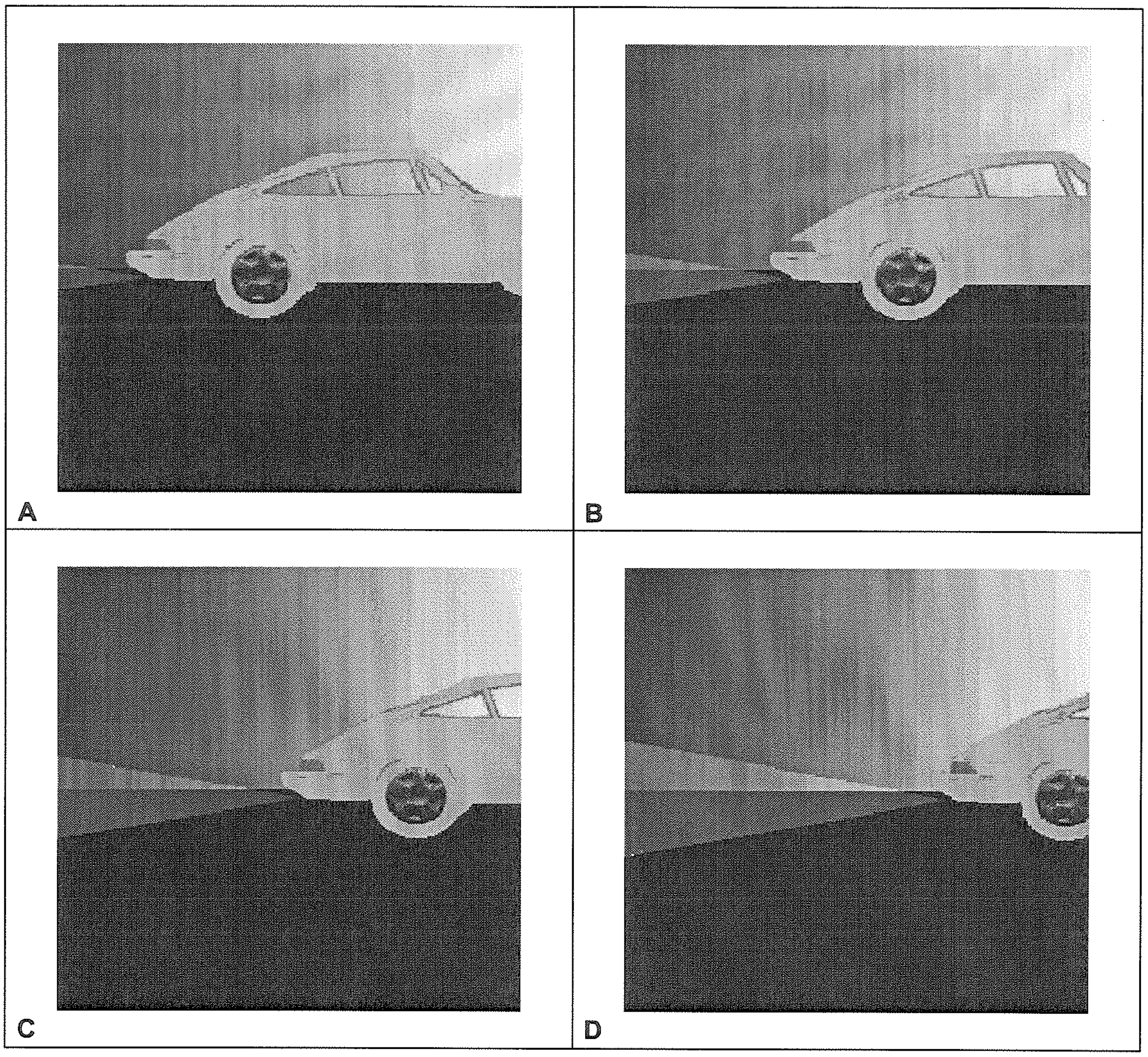


Figure 2.17: Car moving across a temperature gradiant background with a cone of CO at 325 K. (a) frame 1, (b) frame 30, (c) frame 60, (d) frame 90.

and containing 10,000 ppmv of CO (1% CO). This cone translates across the field of view of the camera, approximately 2.5m from camera. This simulation is designed to simulate a car moving across the scene at a distance of 2.5m. More importantly, this simulation tests the two-dimensional capabilities of the Scene object in conjunction with an animated atmosphere.

Figure 2.17 shows four different frames of the animation. Because the temperature range is higher in this simulation, the voltage range runs from 0.25 to 1.5 volts. Although it is nearly invisible, these frames do incorporate the noise model. In this case, we can see that the car does clearly stand out against the background image and the polluted cone also stands out as well. The varying temperature background affects the contrast between the cone and the clean atmosphere which could be a problem when trying to quantitatively detect the concentration of CO in a real plume. By the final frame, the varying thickness of the cone is slightly discernible although the temperature gradient of the background image does mask it. The only problem we encountered was that the atmosphere object must move at a slightly slower rate so that it does not encroach upon the image of the car. One would expect this result because we are approximating a three-dimensional scene using a two-dimensional representation. What actually is happening is that since the car is further away than the plume, it must travel a greater distance in order to cross the frame. If one keeps this in mind, it is possible to model many physical situations using this approach.

In case 2, we were able to show that it is possible to combine a 2D background scene with a 3D atmospheric model to simulate a 3D situation. Like case 1, the slicing planes are very subtle because we used a much finer grid spacing. This was possible because the atmosphere VGO only needed to be in the region where the plume would travel. Futhermore, the concept of the VIO is proven by the overlaying of the car image on top of the background scene using a simple matte. Finally, the animation abilities of both the Scene and Atmosphere objects appear to work well.

2.6 Conclusions

The previous examples show just a few of the uses of IRIMAGE. IRIMAGE has the potential to be used as a tool for both application research as well as system design. At Caltech, we intend to use it to explore the possibilities of passive pollution detection using multi-spectral methods. IRIMAGE will allow us to figure out what gases can be detected and problems might be encountered. Furthermore, it will be possible to try different designs to maximize the detectability of various gases. IRIMAGE will not only determine what pollution detection methods are possible, but also what systems will be best suited for the methodolgy.

In general, one can see how the object-oriented approach makes IRIMAGE an effective tool for research and design for now and the future. Despite its many short comings, IRIMAGE can be easily improved. Throughout its development, many suggested changes have been incorporated with relative ease. In addition, the objects were designed with certain improvements in mind such as different noise functions, new detector models, and more complex geometries. The object-oriented design also allows future developers to incorporate all or part of IRIMAGE in their own software which embodies the concept of code reuse.

We feel that IRIMAGE represents the new philosophy in applied research where costly experiments are becoming more difficult to undertake. Tools like IRIMAGE make it possible to address these experiments from a new direction. Although sometimes more limiting in scope, a simulator like IRIMAGE provides a first cut at a research topic before any expensive investment in equipment is decided upon.

Bibliography

[1] J. R. Schott, R. Raqueño, and C. Salvaggio, "Incorporation of a time-dependent thermodynamic model and radiation propagation model into infrared three-dimensional synthetic image generation," *Optical Engineering*, 31(7), July 1992, 1505-1516.

[2] A.D. Sheffer and J.M. Cathcart, "Computer generated IR imagery: a first principles approach," *Proc. SPIE*, 933, 1988, 199-206.

[3] F.X. Kneizys, E.P. Shettle, L.W. Abreu, J.H. Chetwynd, G.P. Anderson, W.O. Gallery, J.E.A. Selby, and S.A. Clough, "Users Guide to LOWTRAN 7, AFGL-TR-88-0177," Environmental Research Paper No. 1010, Air Force Geophysics Laboratory, Optical/Infrared Technology Division, Hanscom AFB, Maryland (1988).

[4] J.L. Meléndez and C.R. Helms, "Process Modeling and Simulation for $Hg_{1-x}Cd_xTe$. Part I: Status of Stanford University Mercury Cadmium Telluride Process Simulator," *Journal of Electronic Materials*, 24(5), May 1995, 565-572.

[5] For more information contact: Dawn Technologies Inc., 491 Macara Avenue, Sunnyvale, CA 94086.

[6] A. Berk, L.S. Bernstein, and D.C. Robertson, "MODTRAN: A Moderate Resolution Model for LOWTRAN7, GL-TR-89-0122," 1989.

[7] R.A. McClatchey, W.S. Benedict, S.A. Clough, D.E. Burch, R.F. Calfee, K. Fox L.S. Rothman, and J.S. Garing, "AFCRL Atmospheric Absorption Line Parameters Compilation, AFCRL-TR-0096," 1973.

[8] F.X. Kneizys, L.W. Abreu, G.P. Anderson, J.H. Chetwynd, E.P. Shettle, A. Berk, L.S. Bernstein, D.C. Robertson, P.K. Acharya, L.S. Rothman, J.E.A. Selby,

W.O. Gallery, and S.A. Clough, "The MODTRAN 2/3 Report and LOWTRAN 7 Model," *prepared for PL/GPOS*, 1996.

[9] F.X. Kneizyz, et al., "The MODTRAN 2/3 Report and LOWTRAN 7 Model," pp. 110.

[10] W.L. Godson, "The Computation of Infrared Transmission By Atmospheric Water Vapor," Journal of Meteorology, 12, 1955, pp. 272-284.

[11] F.X. Kneizyz, et al., "The MODTRAN 2/3 Report and LOWTRAN 7 Model," pp. 146.

[12] An excellent review of Ray Tracing can be found in: A.S. Glassner ed., *An Introduction To Ray Tracing* (Academic Press, San Diego, 1989).

[13] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C, 2nd Ed.* (Cambridge University Press, New York, 1992), pp.123-128.

[14] A.S. Glassner, "Space Subdivision for Fast Ray Tracing," *IEEE CG&A*, 4(10), Oct. 1984, 15-22.

[15] A. Fujimoto, T. Tanaka, and K. Iwata, "ARTS: Accelerated Ray-Tracing System," *IEEE CG&A*, 5(4), April 1986, 16-26.

[16] J. Levine, *Programming for Graphics Files in C and C++* (John Wiley & Sons Inc., 1994) pp. 35-112.

Chapter 3 Multi-Spectral Experiments and the Verification of IRIMAGE

3.1 Introduction

Any good simulation requires a firm footing in the physics that governs the system that is simulated and a set of experiments that can verify that the simulation's output is logical and realistic. In chapter 2, we discussed the physical background and design of the IRIMAGE simulator. This chapter discusses the results of the experimental verification of IRIMAGE. The original motivation behind the IRIMAGE project was to explore multi-spectral applications of infrared imaging. Since IRIMAGE was written to explore this topic, our verification experiments not only test the various imaging capabilities of the simulator, but also serve as the initial investigation into using multi-spectral imaging in pollution detection. Using a commercial IR imaging system, we demonstrate that IRIMAGE can reasonably and accurately emulate what an actual imager might see through a series of narrow band filters as well as over the wider 3 to 5 micron atmospheric window. By comparing the experimental results with simulations based on the experimental setup, we will show that IRIMAGE generates imaging data consistent with these experiments. In addition, we will demonstrate that multi-spectral imaging, either passive or active, does have promise for the detection of common gaseous pollutants.

In an effort to codify the verification process, we laid out three criteria that IRIMAGE should satisfy in order to be considered a valid simulator. These criteria are:

1. The Scene object accurately emulates a simple background scene that has regions with varying temperatures.

2. The Atmosphere object reasonably approximates the absorptive and radiative effects of a three-dimensional heterogeneous atmosphere.

3. The Imaging system (made up of the FPA, Optics, Detector and Output objects) generates a reasonable and realistic output based on the actual physical parameters of the experimental setup.

Using these criteria in conjunction with the desire to investigate pollution detection using multi-spectral imaging, we designed and conducted three experiments and their accompanying simulations. The first of these experiments is the Relative Temperature Experiment (RTE). The RTE images a background scene that contains two thermally isolated surfaces which are at two different temperatures. The two-dimensional nature of this experiment allows us to verify the Scene object's capabilities. The second and third verification experiments both explore using multiple narrow bandwidth filters to image different atmospheric pollutants. Using a series of narrow band filters, the second experiment analyzes plumes of warm and cold natural gas (CH_4 or methane) exhausted from the standard gas lines entering a typical laboratory environment. The simulation of this experiment not only verifies the Atmosphere object, but also helps determine whether the gas line does contain an excess of CO_2 in addition to the methane. The third experiment utilizes a controlled *gas cell* to explore the multi-spectral imaging possibilities of looking at the exhaust of an internal combustion engine. The gas cell allows us to control the atmospheric content of a section of the optical path so that we may accurately model the same experiment using IRIMAGE. More importantly, the gas cell allows us to use hazardous pollutants like carbon monoxide safely. Since it is impossible to separate the third criterion from the first two, we use empirical comparisons between the experimental and simulation images in combination with results quoted by the imaging system manufacturer to verify the third criterion. After a brief discussion of the imaging system in section 3.2 and some physical background for the multi-spectral pollution detection in section 3.3, each of these experiments is discussed in detail in sections 3.4 through 3.6 of this chapter.

3.2 The Imaging System

Since the third criterion requires that the simulated imaging system generates images that are approximately the same as the experimental images, IRIMAGE must model as much of the actual imaging system as possible. In order to do this, we need to acquire all of the IRIMAGE imaging system parameters for the camera used in the experiment. From this information, we can develop a model that reasonably approximates this camera. However, another set of approximations cause certain problems with reconciling the images produced by the camera and those produced by IRIMAGE. These problems can be minimized using a straightforward calibration process which involves a set of calibration backgrounds. Once this image calibration is done, the images produced by the camera should be comparable to those generated by IRIMAGE.

3.2.1 The Amber AE-256 Imaging System

Throughout the experimental verification phase we used a single infrared imaging system built by Amber Engineering. This system consists of an FPA housed in a liquid nitrogen dewar, a filter wheel, the external controlling hardware for both the FPA and the filter wheel. In addition, the actual imaging data can be captured and stored using a sophisticated 12-bit data acquisition system provided by Amber as well as a standard Hi-8 VCR. A schematic of this imaging system is provided in figure 3.1.

The Amber camera consists of several key components: the FPA, a filter wheel, a dewar and the optical system (See figure 3.2). The focal plane array is made up of 256x256 individual InSb-based PN-junctions which serve as photovoltaic pixels. In order to facilitate a rapid set of multi-spectral experiments, the camera utilizes a cooled filter wheel that can hold up to five filters. Each filter can be rotated into position either manually or automatically through a computer controlled stepper motor. Due to the cooling requirements of the FPA and the need to minimize any thermal effects from the filters, both the filter wheel and the FPA are housed inside a liquid nitrogen cooled dewar. A ferro-fluidic coupler minimizes the thermal effects of

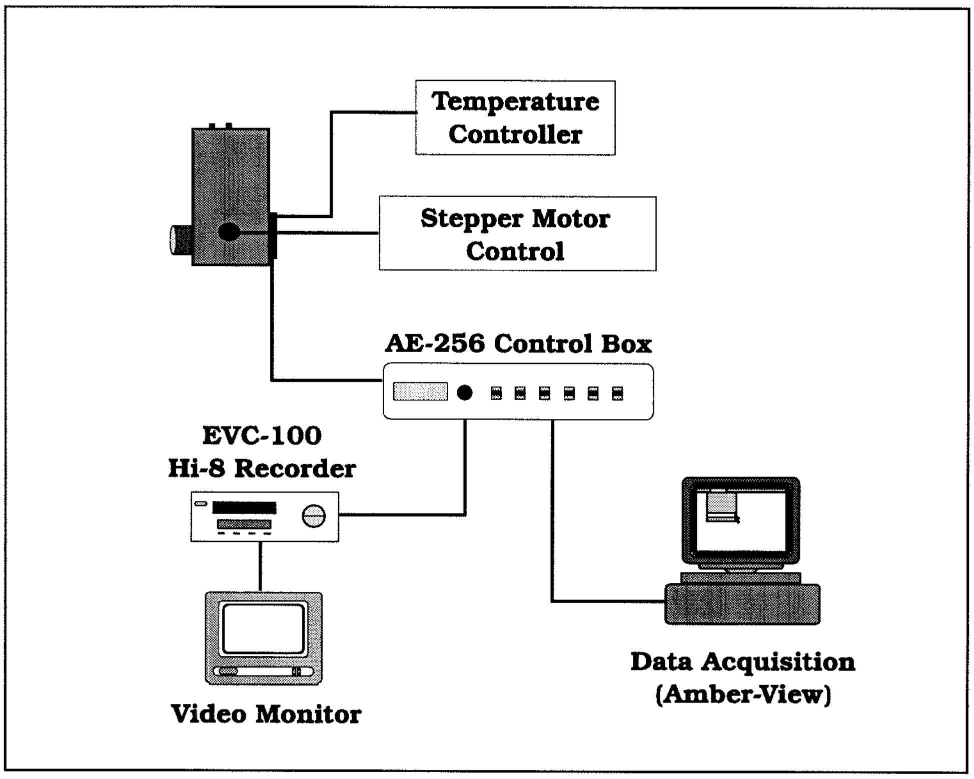


Figure 3.1: Amber AE-256 infrared imaging system with 5 position filter wheel plus support hardware: computer controllable stepper motor for the filter wheel, FPA electronics control box, temperature controller (for measuring temperature of FPA), Hi-8 recorder, video monitor and data acquisition computer.

the external filter wheel drive shaft and the internal drive system. A calcium fluoride (CaF_2) window provides both an aperture for the cold shield while maintaining the vacuum that maximizes the cooling efficiency of the dewar. Besides the CaF_2 window, the camera has an uncooled Germanium (Ge) lens to capture and focus the infrared radiation. This lens has a focal length of 100 mm, an F/# of 2.6 and a FOV of 5.5 °. Table 4.1-1 provides some of the common parameters that describe the imaging system. These components make it possible to image across the entire 3-5 μm infrared range as well as several narrower bands within this range.

The support hardware allows the user to calibrate and retrieve image data and provides the flexibility to control several key parameters used by the FPA to optimize data collection. Prior to calibrating the camera for a specific filter and experiment, the user has the option of setting the integration time, frame rate and initial offset and

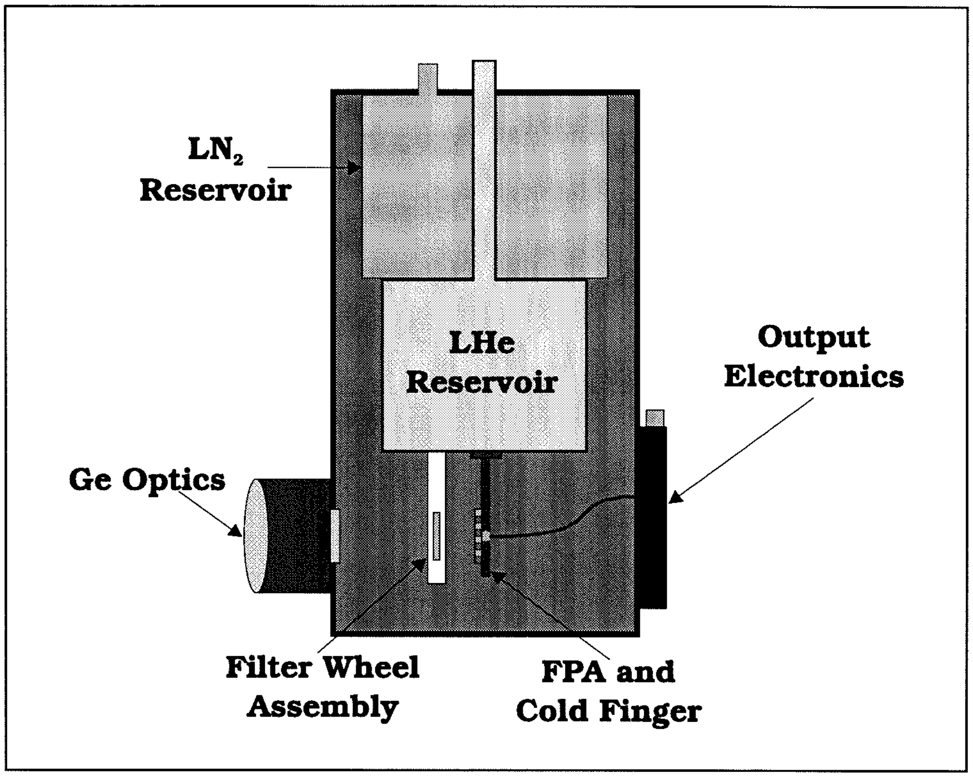


Figure 3.2: Drawing of major components of Amber AE-256 infrared camera.

gain values for the entire FPA. This is particularly useful in multi-spectral experiments where the integration time and the frame rate may need to be varied significantly in order to improve the image. Once these parameters are set, the FPA can be calibrated using a two point procedure in which the imager is exposed to homogenous sources which are at the extremes of the desired temperature range. Using these two images, the control hardware not only sets the individual pixel gains and offsets to maintain uniformity across the FPA surface, but it also identifies bad pixels and compensates for them by substituting neighboring pixel values. After the camera calibration procedure is complete, the global gain and offset can be adjusted to match the conditions of a particular experiment. Most of these parameters can be set either from the Amber electronics control box or via a PC.

The electronics control box provides the image data in both standard video form as well as in 12-bit digital form. The video may be viewed on a monitor in either 8-bit black and white or in 12-bit false color. The video signal may also be recorded on any

Parameter	***Value***
Number of Det.	256 by 256
Detector Pitch	38 μm
Detector Type	InSb Photovoltaic
Detectable Range	1 μm to 5.5 μm
QE	.65
Cint	.45 pF
Storage Capacity	$1.2x10^7$ charges
Operating Temp.	77 K
Mean Responsivity	$1.90x10^{-7}$ volts/photon
Eff. Focal Length	100 mm
Optical F/#	2.545
Optical FOV	5.48 °
CaF Transmission	.91
Cold Shield FOV	22.2 °

Table 3.1: The fixed parameters for the Amber AE-256 camera.

VCR directly in black and white and in color if an RGB color encoder is available. In our system, we record the black and white signal using a Sony Hi-8 VCR with the green channel from the control box providing the signal. Using the Amber supplied data acquisition board and Amber-View software package [1], the 12-bit image data can be captured and stored. This package also can control the camera via an RS-232 connection so that it serves as both a data acquisition station and a camera control system. The Amber-View software controls the acquisition of individual image frames as well as entire sequences of frames. Once these images are captured, they may be processed, analyzed and stored in several standard and proprietary formats. Initially, our 12-bit images were stored as 16-bit FITS images [2]. These images were then processed and stored as 8-bit greyscale GIF images.

For the verification experiments, most of the Amber imaging system parameters are fixed or set by the calibration procedure. Since these parameters are either fixed or not used by the simulation, there are only a few parameters which are allowed to vary in each experiment and are simulated by IRIMAGE. These parameters are the integration time, frame rate of the FPA and the optical transmission curves of the filter used by a particular experiment. These parameters are set prior to the calibration and remain fixed until another filter is selected and the camera is recalibrated.

3.2.2 Simulating the Imaging System using IRIMAGE

In addition to modeling the background scene and atmospheric conditions, IRIMAGE is designed to model all aspects of the imaging system from the optics through to the output voltage from the FPA. To do so, IRIMAGE uses a set of parameter files to define the Optics, FPA, Detector and Output objects. Since IRIMAGE is designed to simulate various imaging systems, we were able to set up these parameter files to emulate the Amber AE-256 system. Using the test data provided by Amber for our focal plane and the optical data for the lens, CaF_2 window and the various filters, we constructed a reasonable model of the imaging system. The parameters used to describe this model in IRIMAGE are provided in table 3.2. A comparison of the parameters used by IRIMAGE in table 3.2 with the data provided by Amber in table 3.1 shows which of the standard parameters are used by IRIMAGE to simulate the AE-256 camera. We note that most of the parameters in table 3.1 that are not used by IRIMAGE are either superfluous or redundant. The two parameters that are useful but not accounted for are the storage capacity and the effective range. The storage capacity is directly related to the maximum voltage which the user can limit when generating the images from the output data. The effective range can be treated as a bandpass filter in the Optics object or should just be accounted for by the user when setting the range to calculate. There are other parameters not specified in either table that need to be accounted for such as the integration time, frame rate and filter transmission. However, since each filter varies in bandwidth and peak wavelength, this may affect both the frame rate and the integration time. Therefore, these parameters are not fixed, but will vary from simulation to simulation. The combination of the fixed parameters and a few varying parameters allows IRIMAGE to model the Amber imaging system with reasonable accuracy.

Although the major parameters of the imaging system are accounted for, there are some parameters that are ignored. The simulation does not account for the uniformity

Parameter	***Value***
Number of Det.	256 by 256
Detector Pitch	38 μm
QE	.65
Mean Responsivity	$1.90x10^{-7}$ volts/photon
Eff. Focal Length	100 mm
Optical F/#	2.545
CaF Transmission	.91
Cold Shield FOV	22.2°

Table 3.2: The fixed parameters IRIMAGE uses to model the AE-256 camera.

and calibration procedures which may set individual pixel gain and offsets. It also does not account for bad pixel substitution or the global gain and offsets. Since these properties may be too difficult to obtain and set (there are over 64,000 individual pixel offsets and gains) or can be introduced once the image data is generated (post-processed), we chose not to incorporate these parameters into the current IRIMAGE modeling scheme. However, the object-oriented nature of IRIMAGE does allow for the implementation of these parameters at some future time.

IRIMAGE does not generate images directly. Instead, the Output simulation object generates an ASCII file with the detector output voltages and various other pixel specific output (incident power, coordinates, etc.). In order to convert this ASCII data into a two-dimensional image, we use the AVS Visualization package from Advanced Visual Systems [3]. Using AVS, we are able to generate images of the detector output (with or without the background fluctuation noise) or the incident power. These images are typically 8-bit gray scale images although AVS is capable of generating 24-bit color images. All of the images generated by IRIMAGE in this chapter are created using AVS.

3.2.3 Comparing Experimental Images with Simulated Images from IRIMAGE.

Several important issues regarding data analysis and image comparison need to be discussed briefly. As mentioned above, several parameters for each pixel are set automatically by the calibration procedure of the Amber system; these include the individual pixel offset and gain as well as the bad pixel substitution. In addition, the global offset and gain values also affect the data retrieved by Amber-View. The analog to digital conversion, the 12-bit nature of the retrieved data and the gain and offset issues make it extremely difficult to reproduce accurate voltages for each pixel of the images collected by the Amber FPA. Therefore, direct comparisons between the actual FPA voltages and voltages generated by IRIMAGE are virtually impossible. However, comparisons are possible between images generated from the FPA data and the IRIMAGE data, provided that an image calibration procedure exists to link the images from the FPA with those from IRIMAGE.

With this idea in mind, we created a method of generating standard images in the experimental setup which can be reproduced in IRIMAGE. These standard images utilize two calibration plates. One of the plates is cold (usually room temperature) and the other is significantly hotter (ΔT between the plates is 5K to 50K). Since the Amber camera must be calibrated for each filter using a hot and cold source, these plates serve as these sources. Prior to the camera calibration, the plates are used to set the integration time and frame rate. During the camera calibration, images of the two plates serve as standards to compute the individual pixel gain and offsets to maximize uniformity across the FPA. In addition, they are used to determine what bad pixels should be remapped. Once the camera calibration is complete, these two plates serve another purpose.

For each experiment, we need to define the extreme temperatures of that experiment. By doing so, we are able to adjust the global gain and offset to maximize the image (voltage) resolution. These two calibration plates define the extreme temperatures for the experiment. With the two plates fixed at their average temperatures, we use the video image to adjust the global gain and offset of the camera to maximize the analog to digital resolution between these two temperatures. Since the 12-bit data returned by the camera has ranges of values between 0 and 4095, we attempt to set the gain and offset so that the average pixel value of the cold plate is between 200 and 300 and the hot plate is between 3800 and 3900. Staying 5% above the minimum

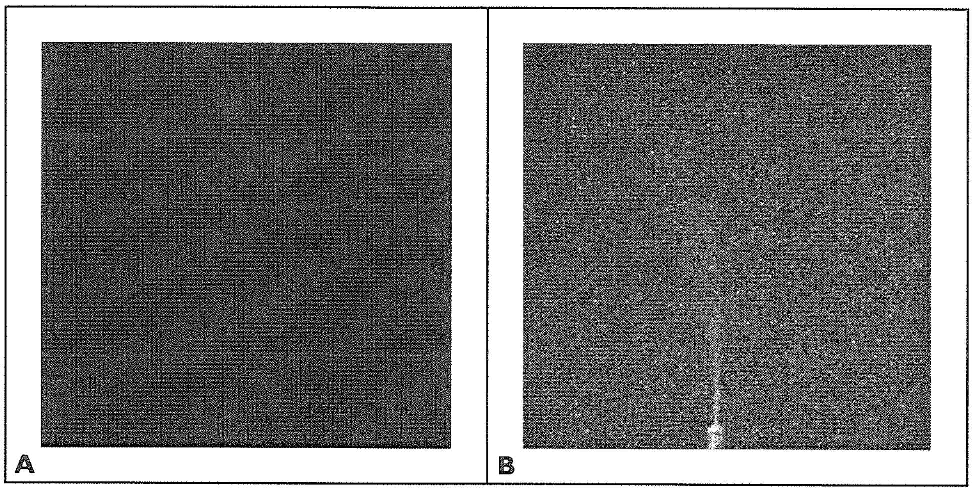


Figure 3.3: Varying the range of a captured image to enhance details. Example of a plume of 97% CH_4 at 310.9K against a 298.4K background. Image captured by looking through a narrow filter centered around 3.45μm and bandwidth of $.047\mu$m. (a) Range is 0 to 3500, (b) Range is 350 to 550.

and 5% below the maximum provides some tolerance for an experiment which has a background temperature near one of these two extreme temperatures. With the gain and offset set, we capture an image of each plate, measure the plate temperatures and store the FPA and temperature data for use with simulation later.

When the experiment is over, it is necessary to convert the collected 12-bit images into usable 8-bit images. This conversion is necessitated by the fact that most computer monitors and printing processes are unable to handle more than 8-bits of greyscale information. Therefore, the experimental images collected by Amber-View must be converted into a more portable form. Amber-View is capable of setting the 8-bit range of any input 12-bit image to any range with a maximum and minimum lying between 0 and 4095. It is possible to use this ranging function to increase the resolution of an 8-bit image by choosing the maxima and minima to be closer together. This is equivalent to changing the gain and offset of the image. This is particularly useful for cases like figure 3.3a where the image has subtle differences that are washed out over the full range. By reducing the range, one can enhance these differences as in figure 3.3b. We use this capability extensively in the images generated by the verification experiments.

The two standard images are very useful in setting the initial range for a set of experimental images. Using the hot and cold images, we can determine the 12-bit values that correspond to black and white respectively (black being 0 and white being 255 in the 8-bit representation). Again, we actually stay 5% to 10% above and below the extremes. This prevents any of the actual experiment images from being clipped over this range. Once these values are set, we save the hot and cold image data as 8-bit GIF files and also record the average 12-bit minimum and maximum that corresponds to that image. Once these two images are saved, the rest of the images are saved with the same range and subsequently any smaller ranges that might enhance specific image data. In all cases, one must record the 12-bit minimum and maximum used to generate each 8-bit image. By doing so, it is possible to compare the data with other 8-bit images that share the same initial range (between the hot and cold plate) and the subsequent smaller ranges.

Once all of this experimental data has been saved, IRIMAGE uses the experiment parameters to simulate the hot and cold plates as well as the actual experiment. Since the average temperature of the plates is known and the plates are painted with a black paint of known emissivity, it is possible to generate a simulated image of these two plates. In the case of IRIMAGE, the images are generated from the individual detector voltages which are floating point values and not 12-bit numbers. We use AVS to generate an 8-bit image from the voltage values in much the same way that Amber-View creates 8-bit images from the 12-bit data.

Once the IRIMAGE simulations are complete, including the images of the hot and cold plates, it is possible to calibrate AVS to generate 8-bit images that can be directly compared with the images captured and processed by Amber-View. Like Amber-View, AVS allows the user to set the minimum and maximum values that correspond to black (0) and white (255). By adjusting these values, it is possible to generate images of the simulated hot and cold plates that have the same 8-bit values as the images of the experimental plates.

Provided that the maximum and minimum for an IRIMAGE simulation are set according to the hot and cold plate image, any images generated using this range

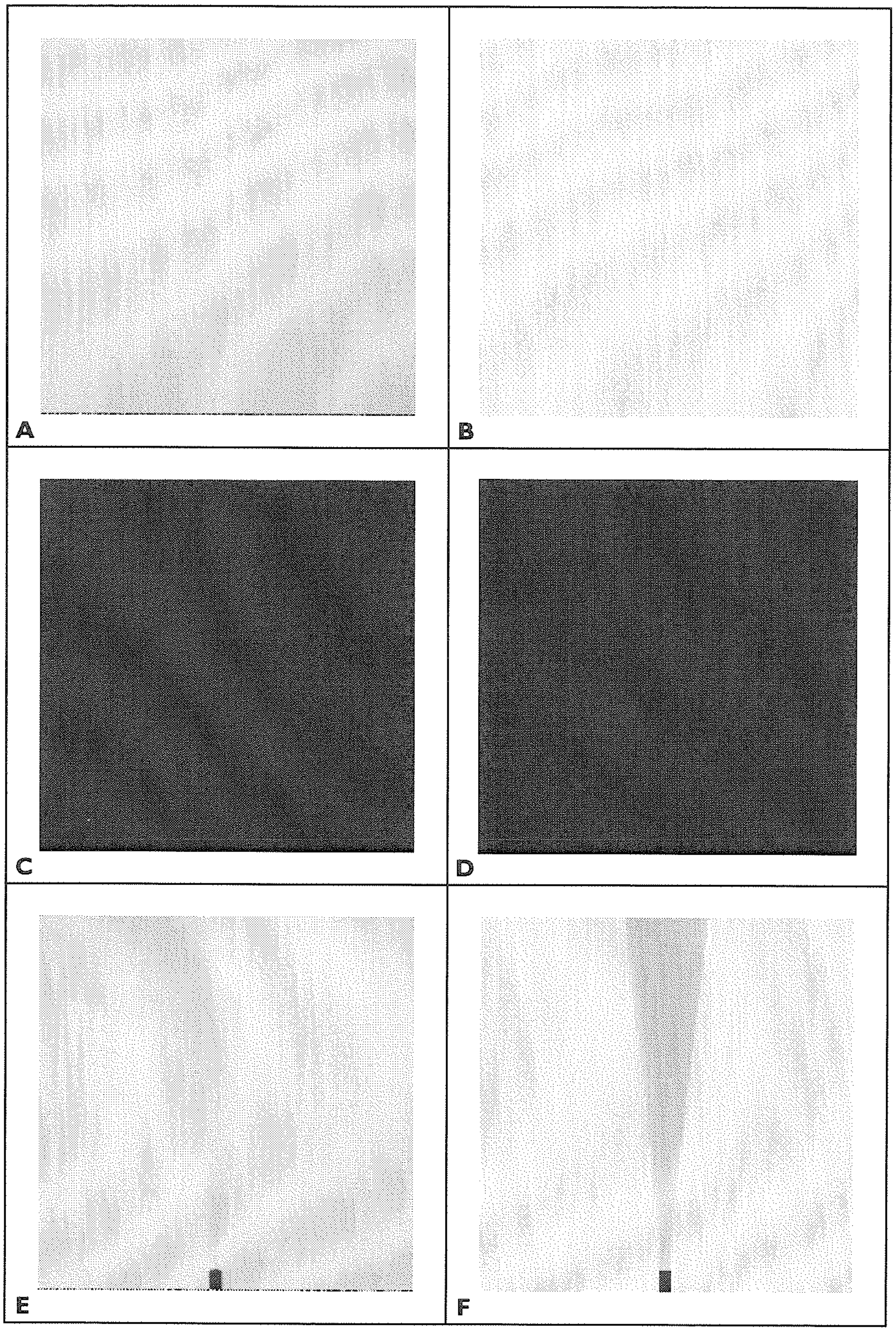


Figure 3.4: Calibrating the experimental and simulation images of the methane experiment using the matched hot and cold plate images from the experiment and simulation. (a) Hot plate image from experiment. (b) Hot plate image from simulation. (c) Cold plate image from experiment. (d) Cold plate image from simulation. (e) Image of plume from experiment. (f) Image of cone from simulation.

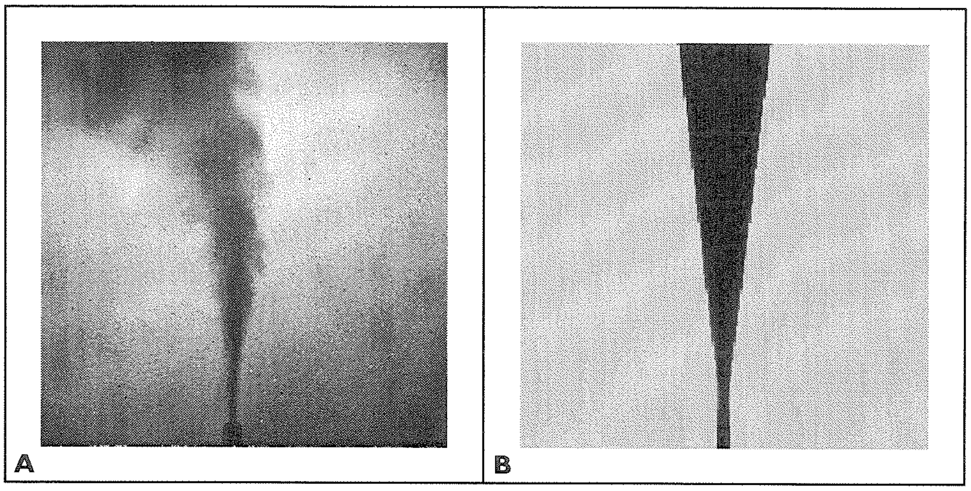


Figure 3.5: Extending figure 3.4 to the case of a range enhanced version of the experimental image and the corresponding simulation image. Methane plume at 310.9K against a background at 329.2K. (a) Enhanced image of CH_4 plume from experiment (2800 to 3400). (b) Enhanced image of CH_4 cone representing the plume from simulation. (467 mV to 555 mV).

can be directly compared with each other. Figure 3.4 demonstrates how this works for the case of the CH_4 experiment. We show the hot and cold images for both the simulated and experimental case of the CH_4 filter. Since they are nearly the same except for minor spatial differences, we can then compare an experiment image directly with the IRIMAGE simulated image and see that they are quite close in fact. Furthermore, an enhanced experiment image with a smaller range can also be compared to a similar simulated image provided that the ranges are also comparable. For smaller ranges, a linear interpolation using the maximum and minimum values for the hot and cold plates can be conducted. Equation (3.2.1) demonstrates how a 12-bit pixel value (P_{exp}) is converted to an IRIMAGE detector voltage ($P_{IRIMAGE}$) using the experiment image's 12-bit maximum (E_{max}) and minimum (E_{min}) and the IRIMAGE maximum (V_{max}) and minimum (V_{min}) detector voltages.

$$P_{IRIMAGE} = \frac{V_{max} - V_{min}}{E_{max} - E_{min}} P_{exp} + V_{min} \tag{3.2.1}$$

Figure 3.5 shows the methane case from figure 3.4 with the images enhanced by this method. All of the image data presented in this chapter follows this procedure for generating and comparing experimental and simulated images with a few noted exceptions.

3.3 Pollution Detection Using Multi-Spectral Imaging Methods

The optical detection of any element or molecule in a gaseous state relies on some type of emission and/or absorption of electro-magnetic radiation. For the single atom case, this type of interaction relies entirely on atomic transitions of electrons. In addition to atomic transitions, many molecules may interact through inter-atomic vibration and molecular rotation, which both rely on dipole absorption and radiation. Most atomic transitions require enough energy that the photons must be in the near-IR to UV range to be generated or absorbed. However, in most cases, the vibrational and rotational modes of molecules emit and absorb photons of much lower energy and lie almost exclusively in the infrared region. Since these modes rely on dipole interactions, any molecules that do not form dipoles (diatomic molecules like N_2 and O_2) will not absorb or emit photons in the infrared range. Fortunately, most of the gaseous pollutants do have a significant IR signature while the three primary constituents of our atmosphere (N_2, O_2 and Argon) have relatively insignificant signatures over most of the infrared range (1 to 100 microns).

Work related to active multi-spectral infrared imaging of the earth's surface and atmospheric conditions has been done for more than twenty-five years [4]. However, most of this work is in the area of terrestrial monitoring from airborne or space borne imaging spectrometers. An example is the AVIRIS (Airbourne Visible/Infrared Imaging Spectrometer) project at JPL. AVIRIS uses the reflected blackbody spectrum from the sun as its source in combination with a "whisk-broom" imager to acquire 224 individual spectral bands between .4 and 2.45 microns [5]. Images are generated by successive scans as the aircraft moves along the surface of the earth. With the development of reliable 2D FPA's, similar systems were designed that use more sophisticated optics to split the incident IR radiation in to many spectral lines along one axis of the FPA, while the spatial information would be stored along the perpendicular axis [6]. Although the same as earlier systems in terms of generating the image, the spectral data and spatial data is collected simultaneously and images can be generated much more quickly and reliably. For most earth-bound experiments and monitoring, this type of system is impractical. Instead, most earth-bound systems rely on active sources and either full IR spectrometers with a single detector or an imaging array with a narrow bandwidth filter. Because a majority of these systems rely on transmission properties of the pollutants, passive detection has been relatively ignored.

Our experiments and subsequent simulations demonstrate the effectiveness and problems associated with trying to image three common gaseous pollutants using either passive or active multi-spectral imaging. The three pollutants are carbon dioxide (CO_2), carbon monoxide (CO) and methane (CH_4). These experiments use an imaging system that contains a filter wheel of narrow bandwidth filters with an imaging FPA. Each filter was chosen to span a region of the infrared spectrum that corresponds to the active spectral region of one of the three gases or a non-active region of all three gases. The active detection experiments image the transmission of each of the three pollutants against a constant high temperature background. The passive detection experiments image the emitted radiation of each of three gases at an elevated temperature against a room temperature background. By collecting and comparing images of the same background and atmospheric conditions but using different filters, it is possible to determine whether passive detection is a viable method of imaging pollution and whether a narrow bandwidth FPA is effective in either active or passive detection.

3.3.1 The Vibration and Rotation Modes of a Molecule

In quantum mechanics, the concepts of absorption and radiation of electromagnetic radiation not only apply to the bound states of an electron, but also apply to the vibration and rotation states of any molecules that exhibit a natural dipole moment. In general, a molecule with N atoms has $3N$ degrees of freedom. If we eliminate the translational motion of the center of mass, the degrees of freedom are reduced to $3N - 3$. These other degrees of freedom are made of the vibrational and rotational states of the molecules. A dipole moment, $\vec{P}$, will form if the atoms of the molecule do not share the electrons equally. Although a molecule may have a dipole moment, only certain vibrations and rotations will generate the oscillating dipole moment which makes it possible to absorb or emit photons. For example, in figure 3.6, the carbon monoxide molecule (CO) has an oscillating dipole for the vibrational state along the bond between the atoms and the rotational mode around the axis perpendicular to the bond. However, the other rotational mode around the axis that is along the CO bond does not cause a change in the dipole moment and thus does not contribute to the absorption or emission of photons. Experimental evidence and quantum theory both demonstrate that the effective vibration and rotation modes of a molecule are quantized. Therefore, only photons with certain energies will be radiated or absorbed by a molecule with an oscillating dipole moment. In the following paragraphs, we describe how the vibrational and rotational modes of a diatomic molecule interact to absorb and radiate photons [7].

Vibrational Modes of a Diatomic Molecule

Since individual molecules are not rigid bodies, the atoms of the molecule are capable of vibrational motion that is constrained by the bonds between the atoms. For example, a diatomic molecule only vibrates along the bond between the two atoms. Figure 3.7 shows a typical energy potential for a vibrating diatomic molecule. In addition, this figure shows a few of the quantized energy levels for this energy potential. For the lower energy states, this potential can be approximated by a quantum mechanical, simple harmonic oscillator (SHO). The ground state energy is not zero, but is $h\nu_0/2$ where ν_0 is the fundamental frequency for the vibrational mode and is a characteristic of the molecule. Using ν_0, the general equation for the energy

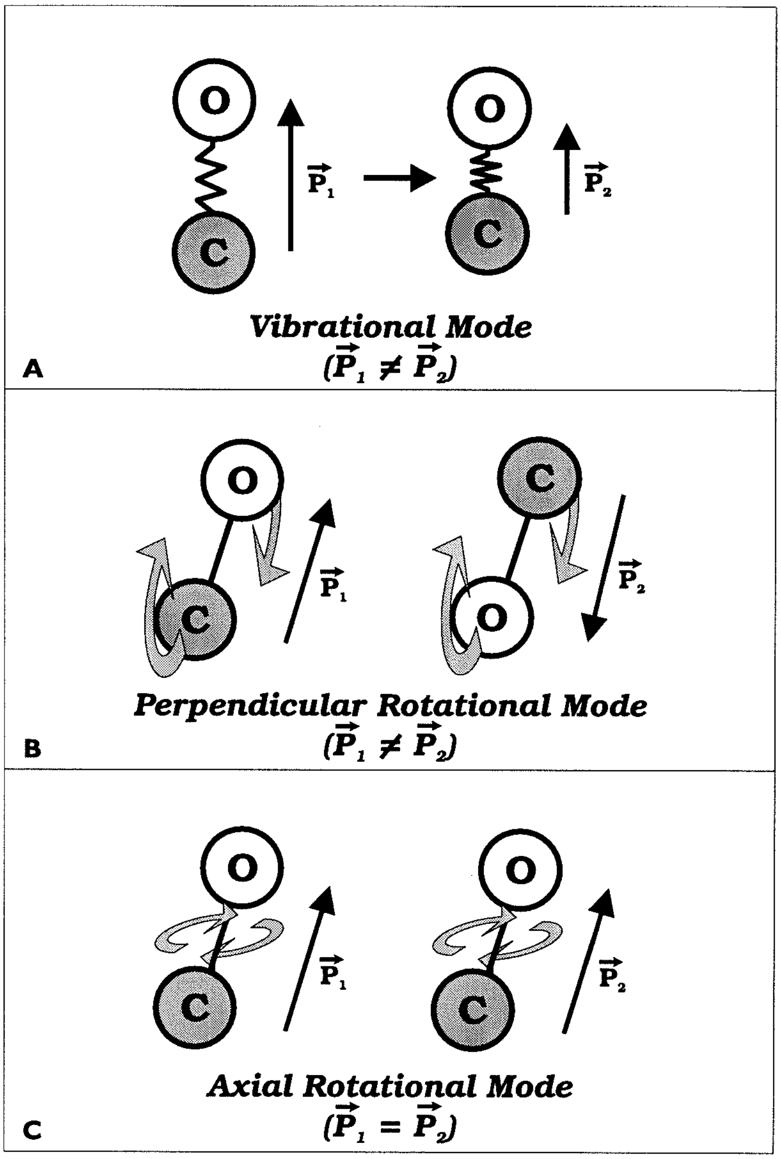

O
C
$\vec{P}_1$
$\vec{P}_2$
Vibrational Mode
$(\vec{P}_1 \neq \vec{P}_2)$
A
Perpendicular Rotational Mode
$(\vec{P}_1 \neq \vec{P}_2)$
B
Axial Rotational Mode
$(\vec{P}_1 = \vec{P}_2)$
C

Figure 3.6: The vibrational and rotational modes of CO: (a) Vibration along the bond forms an oscillating dipole. (b) Rotation around perpendicular axis to bond forms another oscillating dipole. (c) Rotation around bond does not form an oscillating bond.

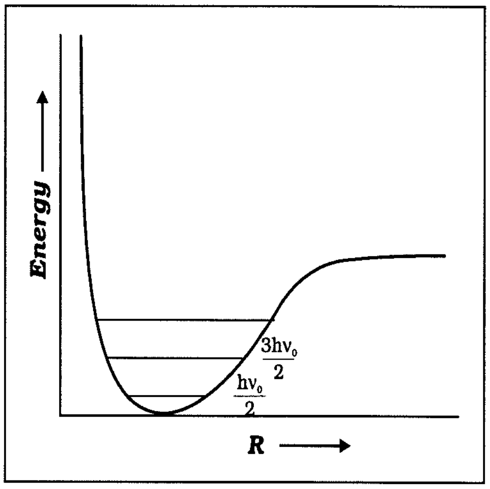


Figure 3.7: The energy potential for a vibrating diatomic molecule. The valley of this curve closely approximates the potential for a simple harmonic oscillator.

states in the SHO region of the potential is:

$$E_n = (n + \frac{1}{2})h\nu_0 \text{ where n is an integer and } n >= 0 \tag{3.3.1}$$

Since the vibration of the molecule results in the oscillation of the dipole moment, transitions between the different energy levels of the molecule result from the absorption or radiation of photons of a specific energy. The selection rule for radiative transitions between vibrational modes (those involving the absorbtion or emission of a photon) is $\Delta n = \pm 1$. From this selection rule and equation (3.3.1), we deduce equation (3.3.2) for the energy of a photon generated or absorbed, $E_{pht} = |\Delta E_n|$:

$$\Delta E_n = \pm h\nu_0 \tag{3.3.2}$$

An important observation is that the selection rule eliminates any energy level dependency in the transition energy. As long as the molecule is not excited to an energy level outside of the QM-SHO approximation, any transition obeying the selection rule will generate a photon with frequency, ν_0. Since the fundamental frequency for most diatomic molecules lies in the 1 μm to 20 μm region of the infrared spectrum, it should be possible to detect these transitions using an infrared detector.

Rotational Modes of a Diatomic Molecule

Rotational motion of individual molecules is also described quantum mechanically. Specifically, the angular momentum of the molecule about its center of mass is quantized. This quantum behavior limits the angular momentum to certain allowed frequencies or "modes." For a diatomic molecule, the only rotation that contributes to the dipole oscillation is the perpendicular rotation (See figure 3.6). With this in mind, the angular momentum can be described by the following equation:

$$L^2 = r(r+1)\hbar^2 \text{ where r is an integer and } r >= 0 \tag{3.3.3}$$

Using (3.3.3) and the moment of inertia, I, the energy of a rotational state is:

$$E_r = \frac{L^2}{2I} = \frac{r(r+1)\hbar^2}{2I} \tag{3.3.4}$$

Radiative transitions between rotation states that involve the emission or absorption of a photon obey the same selection rule as orbital angular momentum, i.e., $\Delta r = \pm 1$. Using this selection rule, we can specify the equations for the energy of an emitted or absorbed photon, $E_{pht} = |\Delta E_r|$, and the characteristic frequency of the photon, ν_r:

$$\Delta E_r = \pm \frac{\hbar^2 r}{I} \tag{3.3.5}$$

$$\nu_r = \frac{E_{pht}}{h} = \frac{\hbar r}{2\pi I} \tag{3.3.6}$$

For most molecules, the value of ν_r is in the 100 μm to 1 mm range which is deep into the far infrared region. Since most IR detectors do not detect radiation at those ranges, it would be very difficult to image molecules that only radiate through rotational methods. However, such a case is not physically allowed according to the selection rules for radiative transitions of a vibrating and rotating molecule. Since transitions preclude a transition resulting in $\Delta n = 0$ or $\Delta r = 0$, all transitions must include both a change in rotational state and a change in vibrational state. Therefore, the energy required for a transition to take place is just a sum of the transition energies between a vibration AND a rotation state. This phenomena has been demonstrated experimentally by the fact that all diatomic infrared signatures do not have a central peak, but have two peaks on either side which represent the perturbation from the central peak because of transitions between the ground state ($r = 0$) and the first excited state ($r = 1$) of the rotational mode.

The Vibrational-Rotational Transition Bands of a Diatomic Molecule

From the results above, one can see that the transition energy spectrum for a molecule with a dipole moment is simply a sum of the transition energies between the initial vibrational and rotational modes and the final ones. Using equations (3.3.2) and (3.3.5), we derive the equation for the allowed transition energies of a diatomic molecule.

$$\Delta E_{n,r} = \Delta E_n + \Delta E_r \tag{3.3.7}$$

$$= \pm h\nu_0 \pm \frac{\hbar^2 r}{I} \tag{3.3.8}$$

Since the vibrational transition energy is the dominant energy value, the direction of this transition determines whether a photon is generated or absorbed. The transitions between adjacent rotational modes splits the vibrational transition energy into a series of energy bands that are centered around this vibrational energy. As mentioned earlier, the selection rules disallow the $r = 0$ energy, preventing a central peak in the band. Under most conditions, a majority of the rotational transitions occur between the ground state and the first excited state. Therefore, instead of a single central peak, there is a peak on either side of the vibrational frequency. Doppler broadening due to the translational motion of the molecules and the conditions related to the Heisenberg uncertainty principle cause the individual energy peaks to spread out and form a series of overlapping energy bands. All of these factors define the characteristic transmission/emission spectrum for any molecule that has a natural dipole moment.

3.3.2 Active and Passive Detection of Gaseous Pollutants

Since we are interested in both passive and active imaging of gaseous pollutants, it is important to understand the physical difference between active detection and passive detection. Thermodynamically, if a gas is at a lower temperature than its background, it will tend toward thermal equilibrium by absorbing the emitted radiation from the background. On the other hand, if the background is at lower temperature than the gas, the gas will move toward thermal equilibrium by radiating photons to its surroundings. Active detection measures the amount of radiation absorbed by gas from a hot background while passive detection measures the amount of radiation emitted by the gas to its surroundings.

Active multi-spectral imaging requires a hot source with a known temperature and emissivity. Using this source as a background, the various pollutants must be introduced into the atmosphere in front of the background. For a filter encompassing the active spectral region of a pollutant, the imaging system will detect a measurable decrease in the incident radiation for regions of the atmosphere with elevated concentrations of the pollutant that lie between the background source and the imaging system. The other filters should have no decrease or a very minor decrease in the transmission for those same regions of the atmosphere. In other words, a set of images utilizing active detection will show the transmission characteristics of the atmosphere.

Passive multi-spectral imaging acts almost in reverse of the active case. The background does act like a source, but only as basis of comparison for the emitted radiation from the hotter pollutant. In this case, the pollutant is hotter than the background source, so it will emit radiation which can be detected by the imaging system. An image, generated using a filter that encompasses the active spectral region of a pollutant, will show a measurable increase in the incident radiation for regions with an elevated concentration of the pollutant. The other filters should demonstrate the same behavior as the active case by showing a minor increase or no increase at all in the incident radiation.

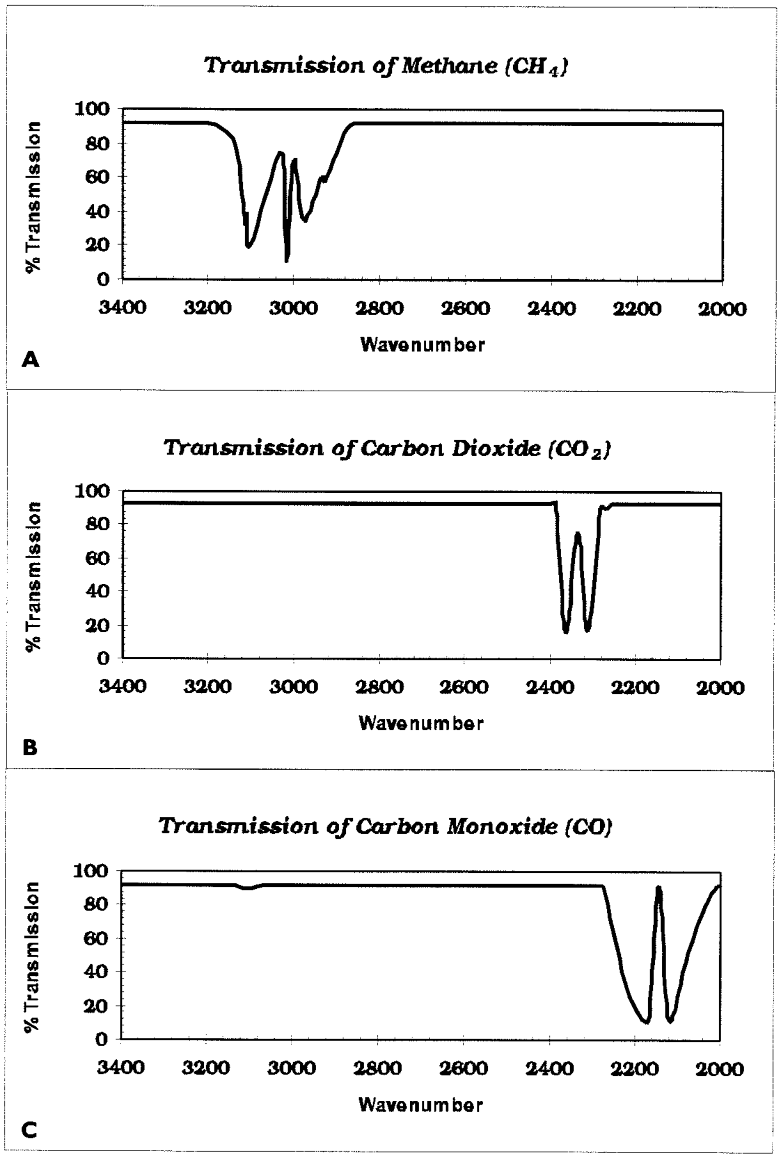


Figure 3.8: The transmission curves over the 3μm to 5μm range (2000 cm^{-1} to 3333 cm^{-1}) for (a) CO, (b) CO_2, (c) CH_4 [8].

3.3.3 Detection of CH_4, CO and CO_2

In our verification experiments, we investigated both the passive and active methods of detecting several common gaseous pollutants using multi-spectral imaging. The three pollutants we examined were carbon dioxide (CO_2), carbon monoxide (CO) and methane (CH_4). Figure 3.8 shows the transmission spectra for these three gases. Since the vibrational energy is independent of the level transition and the rotational modes are only perturbations around this energy, the location of the radiative bands and absorption bands should be approximately the same. Therefore, a filter that encompasses these bands should allow the FPA to image either the transmission or emission properties for one of these pollutants. For this reason, we selected a series of filters that isolate the incident radiation on the FPA to either an active region of a specific pollutant or a region where all three pollutants should not affect the incident radiation at all.

For all three pollutants, we imaged several spectral regions both actively and passively. In addition, we simulated the experiments using IRIMAGE. We discuss the results of these experiments in the sections that follow.

3.4 The Relative Temperature Experiments (RTE)

In multi-spectral imaging, the desired spectral information must be separated from the rest of the incident radiation. In most cases this is either done through a grating or narrow band-pass filter. In either case, this reduces the number of incident photons significantly. Such a reduction will effect an imaging system's ability to resolve temperature differences in the background. It is important that IRIMAGE is able to accurately simulate the imaging effects from different band-pass filters. For this reason, we conducted the Relative Temperature Experiment which looks at a two temperature sources through three different bandwidths of the 3-5 μm range.

The RTE uses a background source that consists of a large area at room temperature and a smaller isolated region at a higher temperature. The smaller region's temperature is controllable and is measured using a thermocouple which measures the temperature difference between the standard plate and the isolated region. This setup can be modeled by IRIMAGE and it verifies the ability of the Scene object to model simple background sources. In addition, the temperature resolution of the camera's FPA for various spectral bandwidths can be measured and compared with IRIMAGE's predictions. This also verifies the noise modeling and detector modeling of IRIMAGE. Finally, the RTE avoids a complicated atmospheric model and uses a standard model instead. By doing so, we are able to separately test the Scene and Atmosphere objects.

We report the results for three different temperature differences in three different spectral bandwidths. Using the temperature measurements and filter transmission curve from these experiments, we simulated each spectral bandwidth and temperature in IRIMAGE. Using the calibration images, we generated the simulated images and then compared them with the actual experimental images. By doing so, we were able to demonstrate the effectiveness of IRIMAGE and validate its output.

3.4.1 Experimental Setup

The RTE consists of a temperature controlled two-dimensional background source and the Amber infrared imaging system. The background source consists of an 8"x8"x.25" sheet of aluminum with a 2.25" diameter hole cut out of the center and an aluminum disk, that is 2" in diameter, placed inside the hole and surrounded by styrofoam to thermally isolate it from the larger plate. The small disk has a small contact heater connected to the back which allows the disk to be heated. The surfaces that face the imaging system are coated with an aluminum primer and then a standard emissivity black paint ($\varepsilon \approx 0.88$). This minimizes any surface reflections from the aluminum. A thermocouple is attached to the surface of the small disk and the surface of the larger plate. This thermocouple allows us to accurately measure the temperature difference between the surface of the small disk and larger area surrounding it. With the heater attached to an adjustable current source and the thermocouple connected to a volt

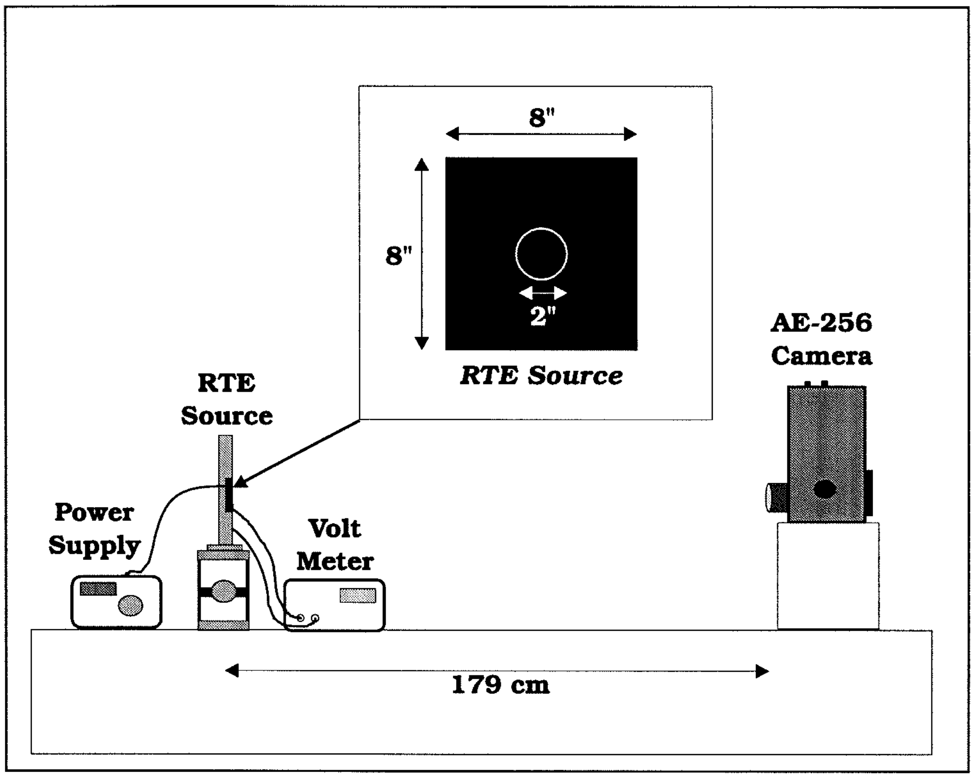


Figure 3.9: A side view of the experimental setup of the RTE with a front view of the RTE source. To generate the calibration images, the RTE source is replaced by the calibration plates.

meter, we are able to heat the small disk and measure the temperature difference between the two plates. Once the RTE source is ready, it is placed 179 cm away from the front surface of the optical system and the small disk is centered in the camera's field of view. After we focus the optical system on the disk (usually by placing a finger right in front of the disk), the camera can capture images.

In addition to the RTE background, we also built two aluminum calibration plates. Both plates are 12"x12"x.25" and are painted with the same black paint as the RTE. The Hot plate has a 12"x12" contact heater attached to its back surface which allows us to set the plate temperature to values in excess of 325K. The Cold plate has an attached reservoir which can be filled with ice water to maintain a relatively stable low temperature plate. In these experiments, the cold plate is kept at room temperature. These plates serve as the calibration plates for calibrating the Amber camera at the beginning of each experiment and also serve as the standard plates used to calibrate the experimental image data with the simulated image data.

3.4.2 Experimental Procedure

Camera Calibration

Since the integration time may be different for different filters, it is necessary to recalibrate the Amber camera each time a new filter is rotated into the view of the FPA. Any time the integration time is changed, it invalidates the current camera calibration. Therefore, after each filter is changed, the following camera calibration must be performed:

1. Rotate the correct filter into place and reinitialize the camera so that a previous calibration does not effect the new calibration.

2. Place the Hot calibration plate in front of the lens about 30 mm from the lens surface. Set the integration time and TIA offset so that the image generated is bright but not white, i.e., does not saturate the FPA. Repeat with the Cold plate to make sure that the offset does not cut off the low temperature region. Write down the values for the integration time, offset and temperatures of the Hot and Cold plates.

3. Begin the automatic calibration procedure and place the Cold and Hot plates in front of the lens when prompted by the Amber camera's software (on the control box or in Amber-View). This procedure defines the temperature range of the calibration using the two plates as the two temperature extremes. It also sets the individual pixel gain and offsets as well as correcting for bad pixels.

4. After the camera is calibrated, use the Hot and Cold plates to generate another bad pixel map in Amber-View.

Using this procedure, we calibrate the camera for a filter and then conduct all of the experiments for that filter before moving on to the next filter.

Generating the Calibration Images

Once a filter is in position and the camera is calibrated, we use the following procedure to capture the calibration images:

1. Set the temperature of the Hot plate several degrees above the expected temperature of the RTE and keep the cold plate at room temperature.

2. Place the Hot plate 179 cm from the front of the lens, focus the lens and set the global gain of the FPA until the image captured by Amber-View has a maximum pixel value between 3800 and 3900.

3. Replace the Hot plate with the Cold plate and set the global offset of the FPA until the image captured by Amber-View has a minimum pixel value between 100 and 200.

4. Repeat steps 2 and 3 until both the Hot plate image has its maximum between 3800 and 3900 and the Cold plate image has its minimum between 100 and 200. For certain filters this is unattainable. In those cases, maximize the difference between the two images.

5. Capture images of the Hot plate and Cold plate and record the temperatures and positions of the two plates as well as the image names and the global offset and gain values for the camera.

Setting up RTE and Capturing Images of RTE Source

After capturing the calibration images, the RTE backgound source can be setup and the initial and final images can be captured in a few simple steps:

1. Place the RTE background at 179 cm from the lens and attach the heater leads to the power supply and the thermocouple leads to the voltmeter.

2. Measure and record the temperature of the large plate and the small plate.

3. Capture an initial image of the RTE source ($\Delta T \approx 0$K).

4. Heat the small plate until the thermocouple reads the desired temperature difference, ΔT.

5. Record the temperature difference and capture a sequence of 30 frames.

Once the images of the RTE background are captured, the calibration image and RTE measurement procedures need to be repeated for each temperature difference. As the temperature difference of the RTE background decreases, the Hot plate temperature should also decrease. By doing so, the gain and offset are set to maximize the difference between pixel values for the heated disk and pixel values for the room temperature plate. The entire procedure must be repeated for every filter.

3.4.3 Simulation Setup

The measured temperature differences, calibration plate temperatures, filter specifics and the integration time for each experiment conducted are presented in tables 3.3 through 3.5. IRIMAGE uses these parameters as well as those specified in section 3.2 to define the imaging system and the background scene. In addition to these parameters, IRIMAGE specifies the location of the RTE source (represented by the Scene object's imaging VGO) to be (0m, 0m, 2.02m) in the WCS. Since the RTE source and calibration plates were perpendicular to and centered on the optical axis, we did not rotate or translate the imaging VGO except along Z.

Parameter	$\Delta T = 10\boldsymbol{K}$	$\Delta T = 5\boldsymbol{K}$	$\Delta T = 1\boldsymbol{K}$
Hot Plate T	337.75 K	311.75 K	302.80 K
Cold Plate T	297.95 K	297.90 K	297.75 K
Integration Time	16.40 ms	16.40 ms	16.40 ms
Frame Rate	60 Hz	60 Hz	60 Hz
RTE T_{plate}	297.85 K	297.65 K	297.75 K
RTE T_{disk}	307.85 K	302.65 K	298.75 K

Table 3.3: Parameters for RTE experiments using the filter N03322-8 ($\lambda_{central}$ = 3.319 μm, FWHM = .071 μm, τ_{peak} = .86).

In order to define the background scene, the Scene object loads a series of 24-bit images which contain 8-bit maps for the temperature, emissivity and matte values.

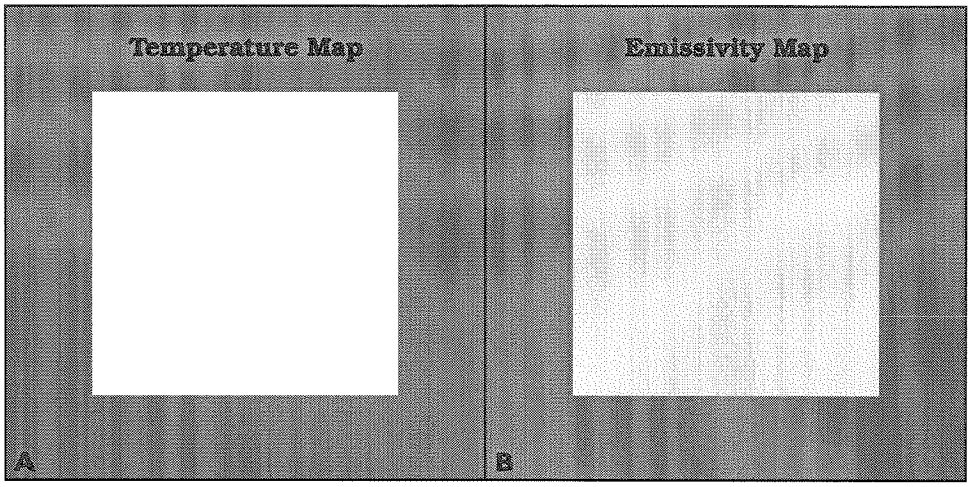


Figure 3.10: The 8-bit maps that represent the temperature and emissivity for the Hot and Cold calibration plates. The maps are also used for the background plate of the RTE source. (a) Temperature map, (b) Emissivity map.

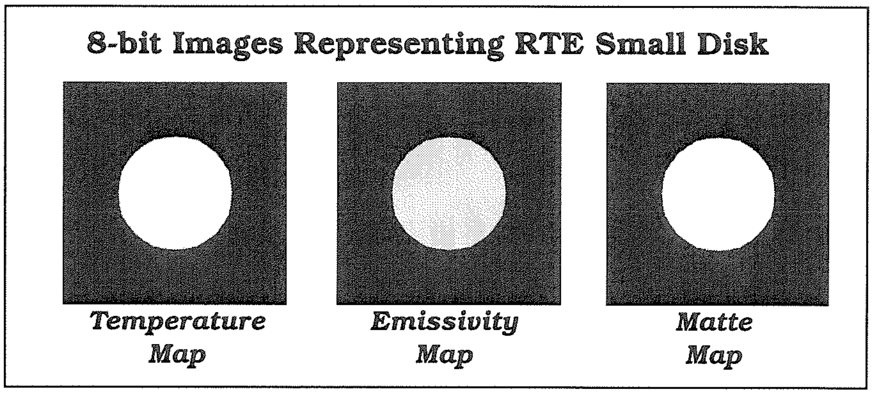


Figure 3.11: The 8-bit maps that represent the small heated disk.

Parameter	$\Delta T = 10\boldsymbol{K}$	$\Delta T = 5\boldsymbol{K}$	$\Delta T = 1\boldsymbol{K}$
Hot Plate T	336.65 K	312.05 K	304.55 K
Cold Plate T	297.95 K	297.65 K	297.75 K
Integration Time	5.208 ms	5.208 ms	5.208 ms
Frame Rate	60 Hz	60 Hz	60 Hz
RTE T_{plate}	297.75 K	297.55 K	297.55 K
RTE T_{disk}	307.6 K	302.55 K	298.55 K

Table 3.4: Parameters for RTE experiments using the filter N03990-4 ($\lambda_{central}$ = 3.930 μm, FWHM = .19 μm, τ_{peak} = .898).

Parameter	$\Delta T = 10\boldsymbol{K}$	$\Delta T = 5\boldsymbol{K}$	$\Delta T = 1\boldsymbol{K}$
Hot Plate T	334.75 K	327.65 K	302.85 K
Cold Plate T	297.55 K	297.55 K	297.95 K
Integration Time	0.391 ms	0.391 ms	0.391 ms
Frame Rate	60 Hz	60 Hz	60 Hz
RTE T_{plate}	297.15 K	297.35 K	297.85 K
RTE T_{disk}	307.15 K	302.35 K	298.85 K

Table 3.5: Parameters for RTE experiments using an open filter position (CaF_2 window only).

These maps are converted from the 8-bit values into the corresponding temperature, emissivity and matte values using a set of multiplicative and additive constants for each map. Using this method, the Scene object represents the Hot and Cold plates using the same image but converted using different multiplicative and additive constants. It also uses this image to represent the RTE source's large plate. The small disk is a separate image that contains a matte which allows it to be composited onto the larger room temperature plate. Figure 3.10 shows the 8-bit images used to define the temperature and emissivity of the calibration plates as well as the large background plate of the RTE source. Figure 3.11 shows the temperature, emissivity and matte images for the small disk which is composited with the background image to generate the RTE background.

In all of the IRIMAGE verification simulations, we assumed that the temperature maps did not have a spatially varying temperature (i.e., each map was fixed at a single temperature). For the RTE experiments, these temperatures were assigned using the values specified in the provided tables. Although we fix the temperature for each temperature map, the emissivity was varied using an image processing technique of lightly speckling the surface of the map. This was designed to emulate the physical painted texture of the calibration and RTE plates. This variance, in conjunction with the multiplicative and additive constants, causes the individual points on the map to vary between .88 and .89 (the emissivity of the black paint is specified to be approximately .88). Finally, this experiment avoids a polluted region and sets the atmospheric conditions to reflect a standard 1976 U.S. atmosphere at 293K and 1 atm. Using these conditions, IRIMAGE generated a series of images that correspond to the experimental conditions and the subsequent images collected by AmberView.

3.4.4 Comparison of Image Results

In this section, we show four figures that compare the experimental results of the RTE with the simulated results. Each figure corresponds to a single temperature difference (10K, 5K, or 1K) and contain an experimental and simulated image for each filter used. These figures are organized with the experimental images on the left and the corresponding simulated images on the right. In addition, a fourth figure is provided that shows the ΔT = 1K case with a much narrower pixel value range (and voltage range for the simulated images). By directly comparing the experiment images and the simulated ones, it is possible to show IRIMAGE's effectiveness as well as some of its problems.

In figures 3.12 through 3.14, we see how the three filter ranges image the RTE source for three different temperature ranges. The experimental images and the simulated images for the 10K difference in figure 3.12 appear to be quite close. Although a few minor differences are apparent, IRIMAGE successfully images the RTE source for all bandwidths.

The results are similar for the 5K difference between the large plate and small disk shown in figure 3.13. However, the narrow filter, N03222, in image A does diverge more than the other filters. This divergence is reasonable because this filter has a very narrow bandwidth (ν = .071μm) and peaks near 3.3 μm, indicating that the detector

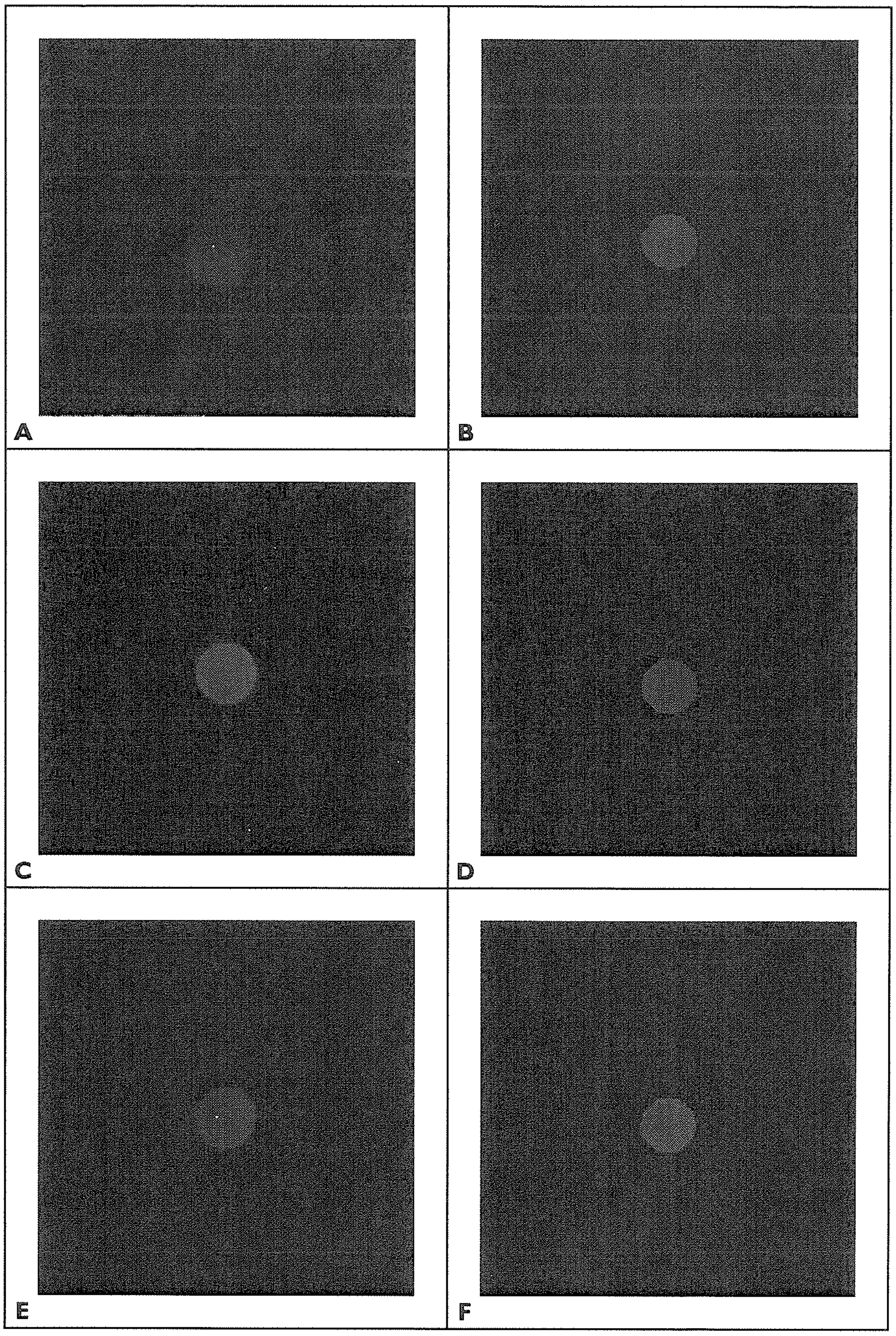


Figure 3.12: Images for the ΔT = 10K case of the RTE.
(a) Narrow filter (N03322-8, Pixel Range = 0 to 3800).
(b) Narrow filter (N03322-8, Voltage Range = 39mV to 407mV).
(c) Wide filter (N03990-4, Pixel Range = 0 to 3900).
(d) Wide filter (N03990-4, Voltage Range = 310mV to 1.37V).
(e) Open filter (none, Pixel Range = 0 to 3700).
(f) Open filter (none, Voltage Range = .72V to 2.47V).

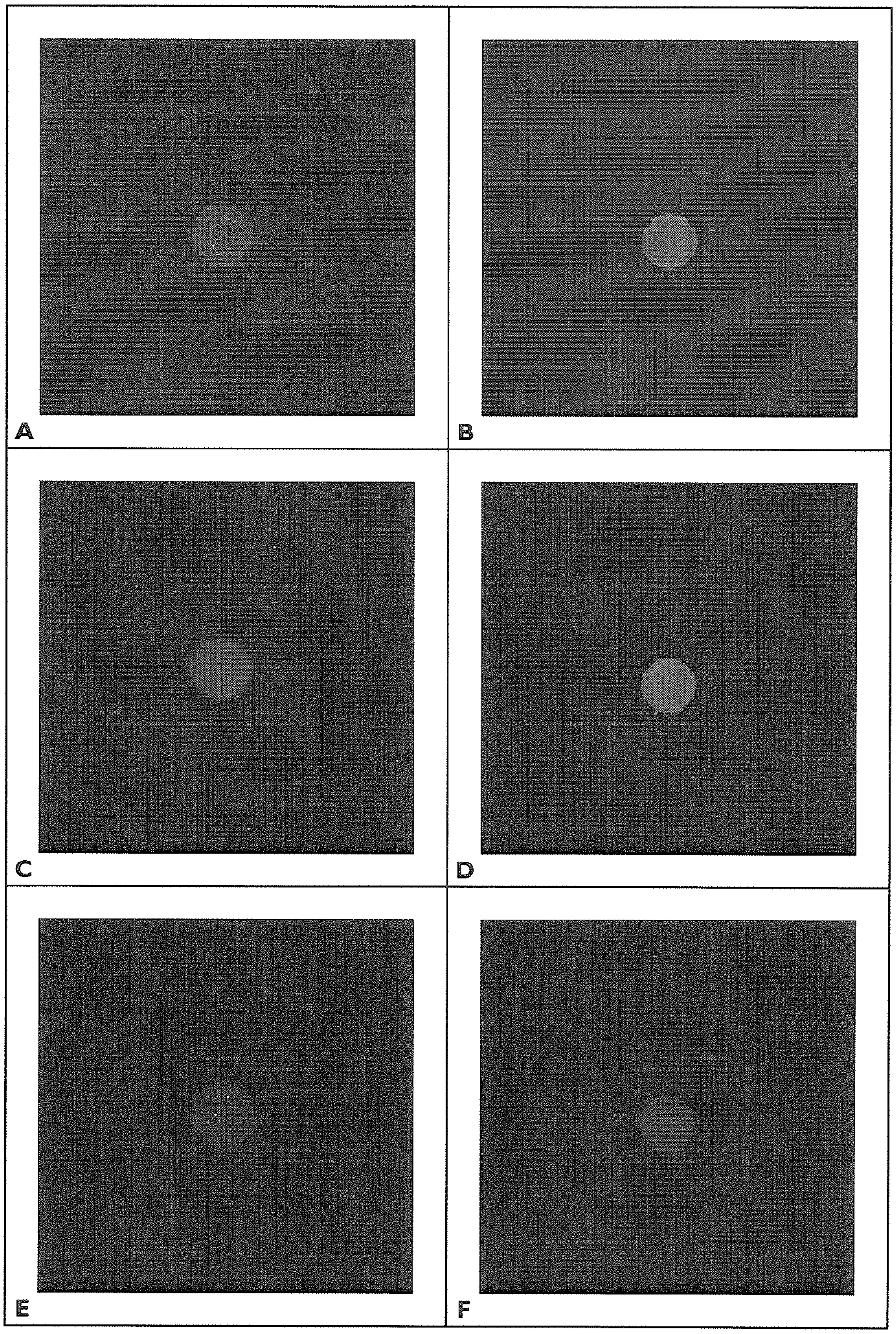


Figure 3.13: Images for the $\Delta T = 5\text{K}$ case of the RTE.
(a) Narrow filter (N03322-8, Pixel Range = 0 to 3600).
(b) Narrow filter (N03322-8, Voltage Range = 55mV to 140mV).
(c) Wide filter (N03990-4, Pixel Range = 0 to 3900).
(d) Wide filter (N03990-4, Voltage Range = 293mV to 568mV).
(e) Open filter (none, Pixel Range = 0 to 3700).
(f) Open filter (none, Voltage Range = .745V to 2.08V).

noise may be a contributing factor. In addition, the heated disk is not entirely isolated from the larger plate and any thermal contact will increase the temperature of the localized region of the large plate near the disk. None of these factors are accounted for in the simulation because IRIMAGE assumes only background flucuation noise, fixes T for the two plates and assumes they are completely isolated.

For the case of $\Delta T = 1\text{K}$, figure 3.14 shows that the divergence in the narrow filter is more pronounced and there is even divergence in image C, filter N03990. Besides the detector noise discussed above, the randomness in image A may also be partially systematic. The calibration procedure for the camera relies on a uniform hot and cold source. Since in this range a minor temperature difference can have a significant effect, it is possible that the calibration plates are another source of the degredation in the experimental images. Furthermore, as we reduced ΔT, we increased the global offset and gain of the camera. By doing so, we increased the effects of any flawed calibration. IRIMAGE does not factor in a bad calibration either.

In figure 3.15, we enhanced the six images of $\Delta T = 1\text{K}$ further. This enhancement introduced an artifact that was apparent in many of the images. In images 3.15c and 3.15e, we can see a triangular ring pattern surrounding the heated disk. This artifact appears to be related to the optical system. It changes shape and position when the filter wheel is rocked back and forth, but it appears for every filter and is more pronounced with a higher global gain value. In addition to this artifact, it is also possible to see the initial effects of the noise model in the simulated image. Although the small disk looks unaffected, the cooler background plate appears to have a slightly random noise pattern across it. This random pattern appears in all three simulated images.

In general, the images generated by IRIMAGE do compare reasonably well with the experimental images. The noticeable exceptions appear to occur under conditions where the approximations in IRIMAGE may begin to breakdown. Other causes of problems may also include the uncertainty of reading the thermocouple values especially around 1K when the error is a significant part of the measurement. Also, the transmission curves are only approxmations while the responsivity curve, assumed

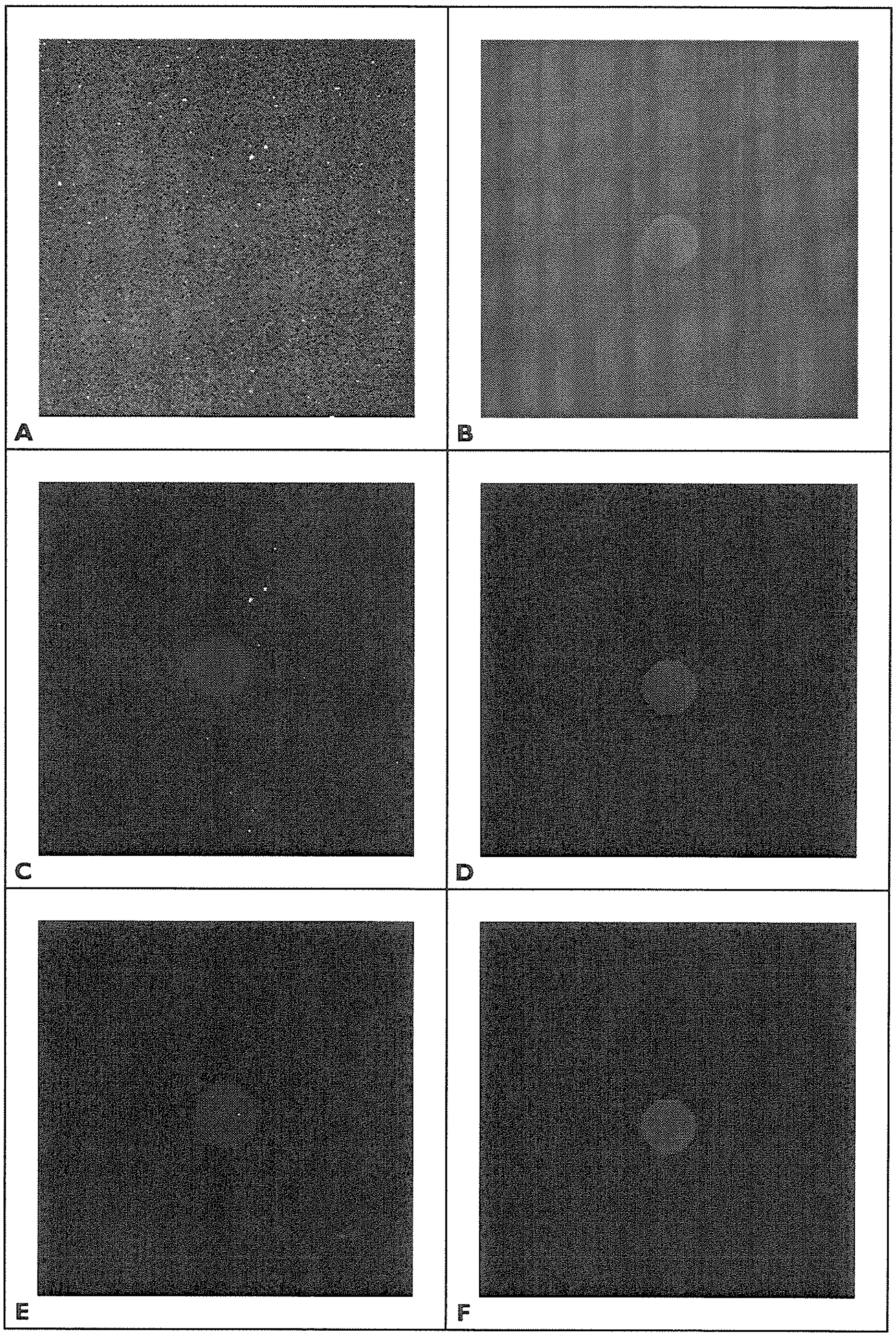


Figure 3.14: Images for the $\Delta T = 1\text{K}$ case of the RTE.
(a) Narrow filter (N03322-8, Pixel Range = 3400 to 4000).
(b) Narrow filter (N03322-8, Voltage Range = 52.5mV to 97.5mV).
(c) Wide filter (N03990-4, Pixel Range = 0 to 3700).
(d) Wide filter (N03990-4, Voltage Range = 310mV to 425mV).
(e) Open filter (none, Pixel Range = 500 to 4000).
(f) Open filter (none, Voltage Range = 815mV to 985mV).

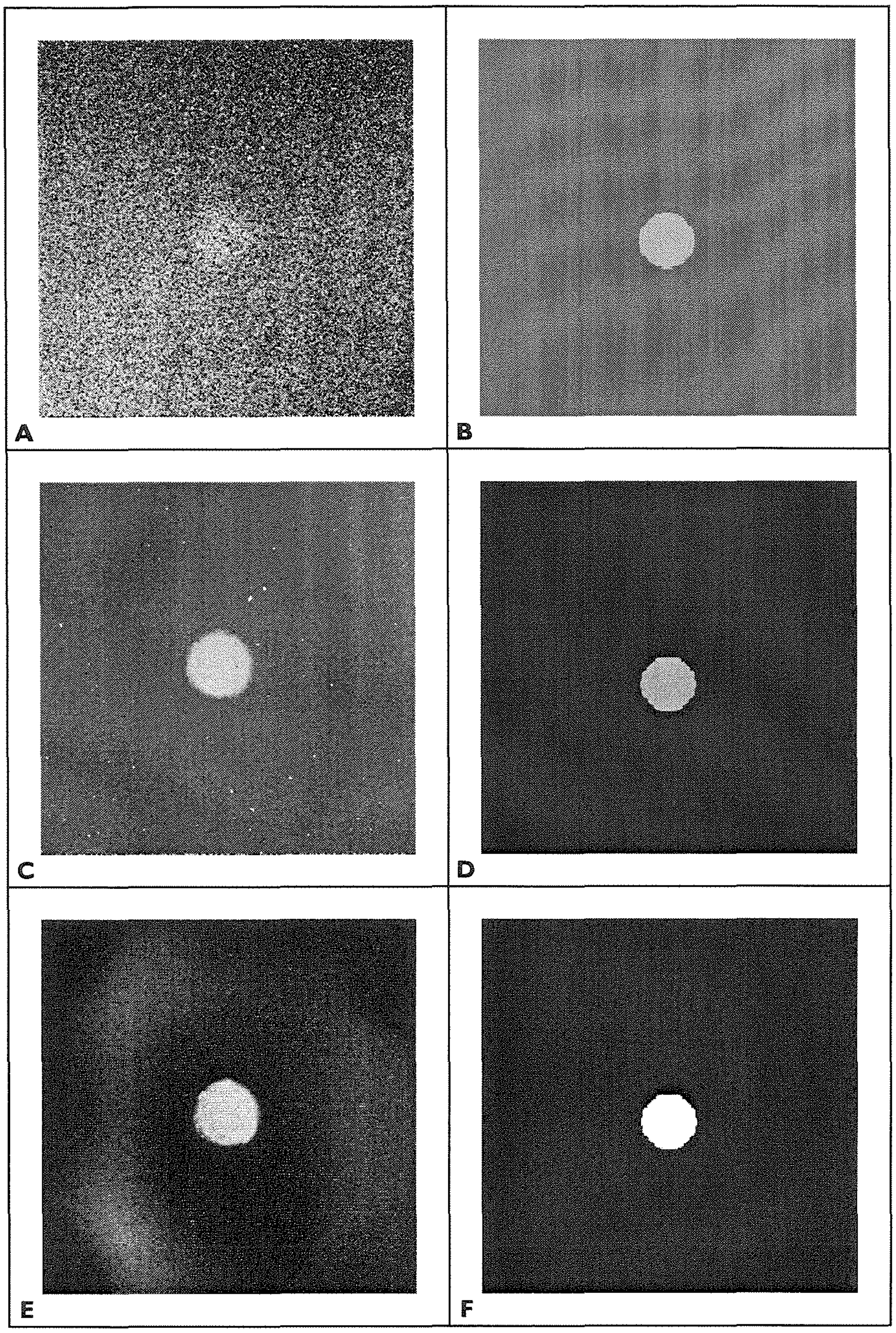


Figure 3.15: Enhanced images for the ΔT = 1K case of the RTE.
(a) Narrow filter (N03322-8, Pixel Range = 3500 to 3700).
(b) Narrow filter (N03322-8, Voltage Range = 60mV to 75mV).
(c) Wide filter (N03990-4, Pixel Range = 0 to 800).
(d) Wide filter (N03990-4, Voltage Range = 310mV to 334.9mV).
(e) Open filter (none, Pixel Range = 500 to 1000).
(f) Open filter (none, Voltage Range = 815mV to 839mV).

to be flat, may have some variance as well. Besides the image comparison, the voltage values generated by IRIMAGE all lie well within the acceptable voltage ranges specified by Amber for this InSb array ($\Delta V \approx 2.1V$). In addition, the size and physical location of the small heated disk appears to be consistant with the experimental images. Besides demonstrating that IRIMAGE does model the background scene effectivly, this also demonstrates that the calibration procedure for comparing simulated images with experiment works well.

3.5 The Methane Experiments

Hydrocarbons are a common pollutant in the atmosphere. Experimental infrared spectroscopy results show that most of the common hydrocarbons (methane, butane, propane, etc.) have significant infrared signatures around 3.4 microns. This wavelength is associated with the carbon-hydrogen fundamental frequency of vibration. Since this is the only bond capable of generating a dipole in most hydrocarbons, it is nearly impossible to discriminate between the different hydrocarbons. However, it is usually less important to know which hydrocarbon is emitted and more important to know that they are emitted. Therefore, an imager with a set of narrow band-pass filters should be able to isolate the hydrocarbon band from another band. By comparing images, it should be possible to determine the existance and rough concentration of hydrocarbons flowing into the atmosphere from an exhaust source. IRIMAGE must be able to simulate what an imager whould "see" under these conditions. The methane experiment investigates these issues.

The methane experiment explores IRIMAGE's abilities to represent a small but highly concentrated plume of natural gas flowing into a standard atmosphere using a static cone model. Like the RTE, the methane experiments use a simple background. However, it is not the RTE background source, but just one of the calibration plates at a specific temperature. In addition, this experiment introduces a small diameter copper tube that exhausts natural gas from the laboratory gas line. IRIMAGE models the background using a background image that is a composite of one of the calibration plate images and a scaled and translated version of the same image that represents the tube. IRIMAGE represents the plume using the Atmosphere object and two static cylinders that represent two different concentrations of natural gas. Therefore, we verify IRIMAGE's ability to model localized atmospheric pollution and compare it with experiment. In addition, we are able to demonstrate how a series of narrow band filters can isolate emission or transmission bands of various pollutants.

We report the results of three separate filter experiments and their subsequent simulations. These filters isolate the CH_4 band, the CO_2 band and a relatively clean band that lies between the CO_2 and CH_4 bands. For each filter, we looked at four possible scenarios for the temperature of the natural gas and the background plate:

- A Hot plume against the Hot background plate.
- A Hot plume against the Cold background plate.
- A Cold plume against the Cold background plate.
- A Cold plume against the Hot background plate.

The simulations generate calibration images which are based on the experimental image parameters. Using the calibration procedure, we are able to generate simulated images that can be compared to the experimental results for each filter.

3.5.1 Experimental Setup

As figure 3.18 shows, the experimental setup of the methane experiment is very similar to the experimental setup for the RTE experiment. The background source is either the 12"x12" Cold plate or the 12"x12" Hot plate. Both plates are placed at a distance of 2.43 m from the front surface of the optics. In addition to providing the background for the methane experiments, the calibration plates again are used to generate calibration images. These images will be used to calibrate the images generated by IRIMAGE with ones captured by the AE-256 imaging system.

The natural gas plume exits from the end of a 1/4 inch copper tube that is approximately 25 feet long and is coiled up into a 1 ft in diameter coil. This coil lies

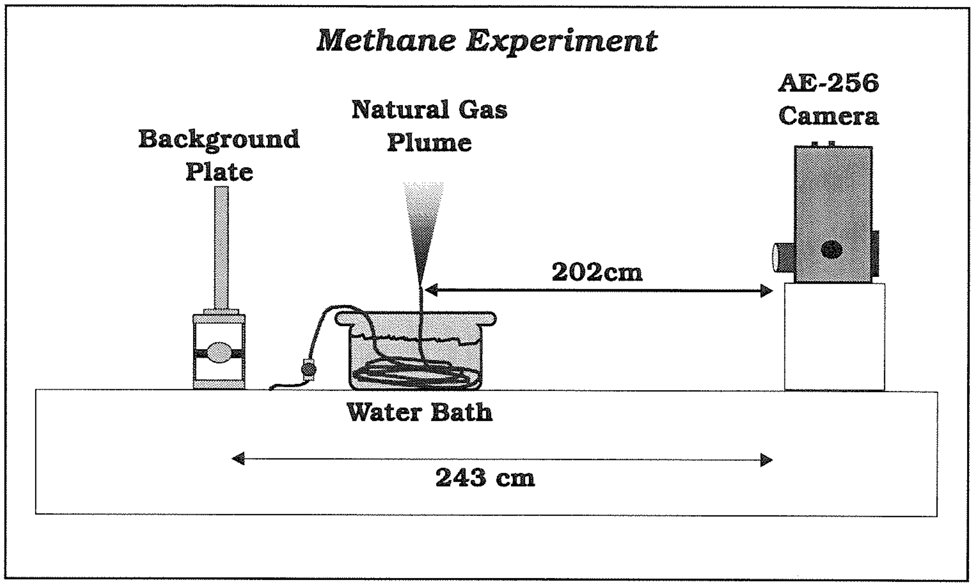


Figure 3.16: A side view of the experimental setup of the Methane experiment.

in a plastic tub filled with water. The temperature of the exhaust gas is regulated by the temperature of water in the tub. The thermal conductivity of the copper tube and the length of the coil turns the tub-coil combination into a heat exchanger making it possible to either produce a room temperature or a heated plume of gas. We measure the temperature of the gas using a commercial gas thermocouple that has accuracy to the .1K. We also measure the temperature of the exhaust tube and the background surface using a commercial surface thermocouple capable of the same accuracy. The 3 mm exhaust tube opening lies approximately 8 cm below the optical axis of the camera and is located 202 cm from the front of the optical surface.

The natural gas comes directly from the laboratory gas line. Table 3.6 shows the concentrations of the various gases in the natural gas mixture. The information in this table was provided by the Gas Company.

Molecule	*Mole Percent*
N_2	0.03
O_2	1.48
CO_2	0.22
CH_4	92.88
Other CH compounds	5.39

Table 3.6: Consituents of natural gas.

The only constituents we are concerned with are the concentrations of nitrogen, oxygen, methane and carbon dioxide. Since MODTRAN does not deal with any of the other CH compounds and they make up less than 6% of the total gas, we neglect their contribution.

3.5.2 Experimental Procedure

Calibration procedures

The camera calibration procedure is exactly the same as in the RTE experiment (see section 3.4.2). Although the camera must be recalibrated when the filter changes, the calibration images do not need to be generated for each of the four cases investigated. Instead, the global offset and gain remain the same for all four cases making it unnecessary to capture multiple calibration images under the same filter. Therefore, the image calibration procedure only needs to be conducted after the camera calibration procedure (See section 3.4.2).

Setting up Hot Plume Cases and Capturing images

We start out each filter by measuring a hot plume against a hot and cold background. The following procedure describes how we setup the experiment and capture images of these two cases.

1. Place the Hot background plate 243 cm from the lens and centered on the optical axis. Measure its temperature.

2. Place the coil in a bath of hot water which is approximately 320K.

3. Place water bath and coil at position where the end of the exhaust tube is 202 cm from the lens and is 8cm directly below the optical axis.

4. Capture the initial image of the background prior to flowing natural gas through the coil.

5. Open gas valve and let gas flow through the coil. After 10 seconds, capture a sequence of 30 frames.

6. Measure the temperature of the plume, the ambient temperature of the air and of the tip of the exhaust tube.

7. Close the gas valve.

8. Replace the Hot background plate with the Cold plate (still 243 cm from lens) and measure the plate temperature.

9. Replace the water in the bath with more hot water and reposition the coil to same position as step 3.

10. Repeat steps 4 through 7 with the Cold background plate.

Setting up Cold Plume Cases and Capturing images

Once we have measured the hot plume cases for the current filter, we repeat the same procedure using a cold plume. For these two cases, the only thing that changes in the ten steps described above is that the hot bath is replaced with a cold bath. This cold bath uses water that is approximately 295K. In order to make sure that the coil is thermally stable at the cold bath temperature, it is necessary to place the coil in the bath and let it stand for 5 minutes and then replace the cold bath with a fresh amount of cold water. Other than this minor adjustment, the procedure is exactly the same.

3.5.3 Simulation Setup

Like the RTE experiment, tables 3.7 through 3.9 contain the various temperature measurements and settings for the four cases investigated under each of the three filters. These parameters are used by IRIMAGE to model the background and calibration images as well as the parameters necessary to model the imaging system for each filter. In addition, the coordinates of the background scene (0m, 0m, 2.43m) and the exit point of coil (0m, -.08m, 2.02m) are already known in the world coordinate system. Using these values, we are able to place the Scene object's imaging VGO in the right location and are able to determine the position and location of the burner image. Finally, the position of the end of the coil also provides the starting location for the gas plume model.

Parameter	***Hot/Hot***	***Cold/Hot***	***Cold/Cold***	***Hot/Cold***
Hot Plate T	328.85 K	328.85 K	328.85 K	328.85 K
Cold Plate T	298.65 K	298.65 K	298.65 K	298.65 K
Integration Time	32.81 ms	32.81ms	32.81 ms	32.81 ms
Frame Rate	30 Hz	30 Hz	30 Hz	30 Hz
T_{source}	329.15 K	298.35 K	298.65 K	329.25 K
T_{gas}	310.85 K	310.85 K	297.25 K	297.25 K
T_{tube}	311.75 K	311.75 K	297.35 K	297.35 K
$T_{ambient}$	297.75 K	297.75 K	297.65 K	297.65 K

Table 3.7: Parameters for methane experiments using the filter N03322-8 ($\lambda_{central}$ = 3.319 μm, FWHM = .071 μm, τ_{peak} = .86).

Parameter	***Hot/Hot***	***Cold/Hot***	***Cold/Cold***	***Hot/Cold***
Hot Plate T	326.75 K	326.75 K	326.75 K	326.75 K
Cold Plate T	297.25 K	297.25 K	297.25 K	297.25 K
Integration Time	7.813 ms	7.813 ms	7.813 ms	7.813 ms
Frame Rate	30 Hz	30 Hz	30 Hz	30 Hz
T_{source}	327.55 K	298.05 K	297.25 K	327.55 K
T_{gas}	310.95 K	310.95 K	297.25 K	297.25 K
T_{tube}	311.65 K	311.65 K	297.25 K	297.25 K
$T_{ambient}$	297.75 K	297.75 K	297.05 K	297.05 K

Table 3.8: Parameters for methane experiments using the filter N03990-4 ($\lambda_{central}$ = 3.930 μm, FWHM = .19 μm, τ_{peak} = .898).

The gaseous plume of methane is represented using two static cylindrical objects. One of these cylinders represents a gas model that is 100% natural gas while the other cylinder represents a gas model that is only 25% natural gas. Figure 3.17 demonstrates how these two cylinders are placed with respect to each other. The 100% cylinder begins with a radius of 3mm which is approximately the size of the opening at the end of the coil. At the other end of the 100% cylinder, the radius is only 1mm wide. Thus, it approximates this region shrinking as the gas moves further away from the end of the coil. The other cylinder is reversed and has a 1mm radius at the end of the coil and its final radius is 30 mm. Thus, these two objects are a rough approximation of a gas becoming less dense as it moves further away from the

Parameter	*Hot/Hot*	*Cold/Hot*	*Cold/Cold*	*Hot/Cold*
Hot Plate T	326.85 K	326.85 K	326.85 K	326.85 K
Cold Plate T	297.85 K	297.85 K	297.85 K	297.85 K
Integration Time	7.813 ms	7.813 ms	7.813 ms	7.813 ms
Frame Rate	30 Hz	30 Hz	30 Hz	30 Hz
T_{source}	326.85 K	297.80 K	297.15 K	326.85 K
T_{gas}	309.95 K	309.95 K	297.55 K	297.55 K
T_{tube}	311.25 K	311.25 K	298.05 K	298.05 K
$T_{ambient}$	297.35 K	297.35 K	297.05 K	297.45 K

Table 3.9: Parameters for methane experiments using the filter N04235-4 ($\lambda_{central} = 4.235$ μm, FWHM = .181 μm, $\tau_{peak} = .87$).

end of the coil. Both cylinders are 25 cm long and are rotated -90 degrees around the X axis. Their centers both lie at the world coordinate (0m, .036m, 2.02m). We use an Atmosphere VGO that is centered around the optic axis at the point 202 cm from the lens and has a grid point resolution of 1 mm in all three directions. In addition to a standard atmospheric gas model, we also define gas models for 100% natural gas and 25% natural gas/75% standard atmosphere. The two gas models are associated with the respective cylinders while the rest of the atmosphere is associated with the standard atmosphere gas model. The temperatures of these gas models are all set by the experimental measurements except for the 25% model which is a linear interpolation between the ambient temperature and the temperature of the 100% natural gas model.

The background scene is modeled using the calibration image in 3.10. In this case, we model the exhaust end of the copper tube by using the Scene object's ability to scale and translate an image and place it in a specific location of the imaging grid. Like the RTE, the temperature of this image is constant, but the emissivity varies between .88 and .885. For simplicity, we modeled the end of the coil using the same image and then scaled and translated it to match the scale and location of the end of the coil in the experimental images. By doing so, we are able to generate images that appear to have the plume coming out of the end of the coil. The temperature of both the background plate and the end of the coil are set according to the recorded experimental data.

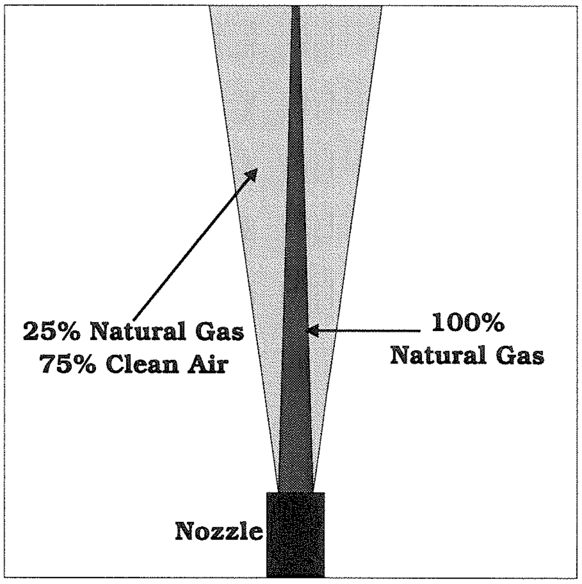


Figure 3.17: Example of how the two cylinders are combined to model the static exhaust cone that represents the plume.

3.5.4 Comparison of Results

Since the experiments examined the four cases of the methane plume against a background, we report the experimental and simulation results here. Like the RTE discussion, the figures we present here have the experiment images on the left side and the simulation images on the right. In this case, we are interested in how IRIMAGE models the atmosphere versus the actual experimental evidence. It should be noted that the cone representation used by IRIMAGE becomes less accurate near the top of the wider portion of the cone. This occurs because the cone model assumes that each cylinder has a constant concentration. However, physically this is not true because as the plume expands, the fraction of the pollutant in a plume decreases. Therefore, our discussions about the experimental plume versus the cone model will center around the exit point of the nozzle.

The images presented are enhanced using the technique described in section 3.2. For most of the images, the enhancement uses equation (3.2.1) to scale the simulation images to match the experimental images. However, a few images were better represented by a minor offset in one direction or another. We believe that this offset is probably due to minor errors in the measurement of the calibration plate temperatures and in the measurement of the gas and background temperatures. For the CO_2 filter, we used images that were the difference between the captured image and a standard image taken prior to opening the gas valve. For these images, we set the voltage range of the simulations so that the scale of the range (ΔV) matched the scale of this difference image. We did so by using (3.2.1) and incorporating an additive constant. In any case, we will note either an offset or difference image in the figure caption.

In figure 3.18 we see the hot plume of methane against a hotter background. It is clear that the IRIMAGE cone representation produces a close approximation of the experimental plume for the CH_4 filter. Near the nozzle end, they look almost identical. As the IRIMAGE cone expands, it exhibits the behavior we discussed above. The background of the simulated image is much brighter for this filter than the experimental image. This is expected since IRIMAGE does not increase the methane content in the surrounding atmosphere while it actually does increase in the experiment. The clean filter appears washed out in the experiment, but a very faint cone is visible in the simulation. Since this filter does overlap the CO_2 bands slightly, we expect that we are seeing just a minor loss of transmission due to this band. We suspect that the experimental image may also have been saturated by an integration time too high. Finally, the CO_2 filter appears to have isolated a very faint carbon dioxide plume. Experimentally, the only way to clearly see this plume was to subtract a standard frame from the polluted frame. By removing the static clutter from the background, we are able to clearly see the plume of CO_2. In the simulated image, the corresponding cone model is also visible. The bright lines around each cylinder in the simulation are artifacts of MODTRAN's computation. Clearly, this figure demonstrates the atmospheric modeling capabilities of IRIMAGE as well as demonstrates how an active approach to multi-spectral imaging can be effective.

In figure 3.19, the hot plume remains, but the background is replaced with a room temperature background. Again, IRIMAGE seems to accurately reproduce images that are very close to the experimental evidence (near the nozzle). In this case, we can see both in the simulation and experimentally, that the methane and

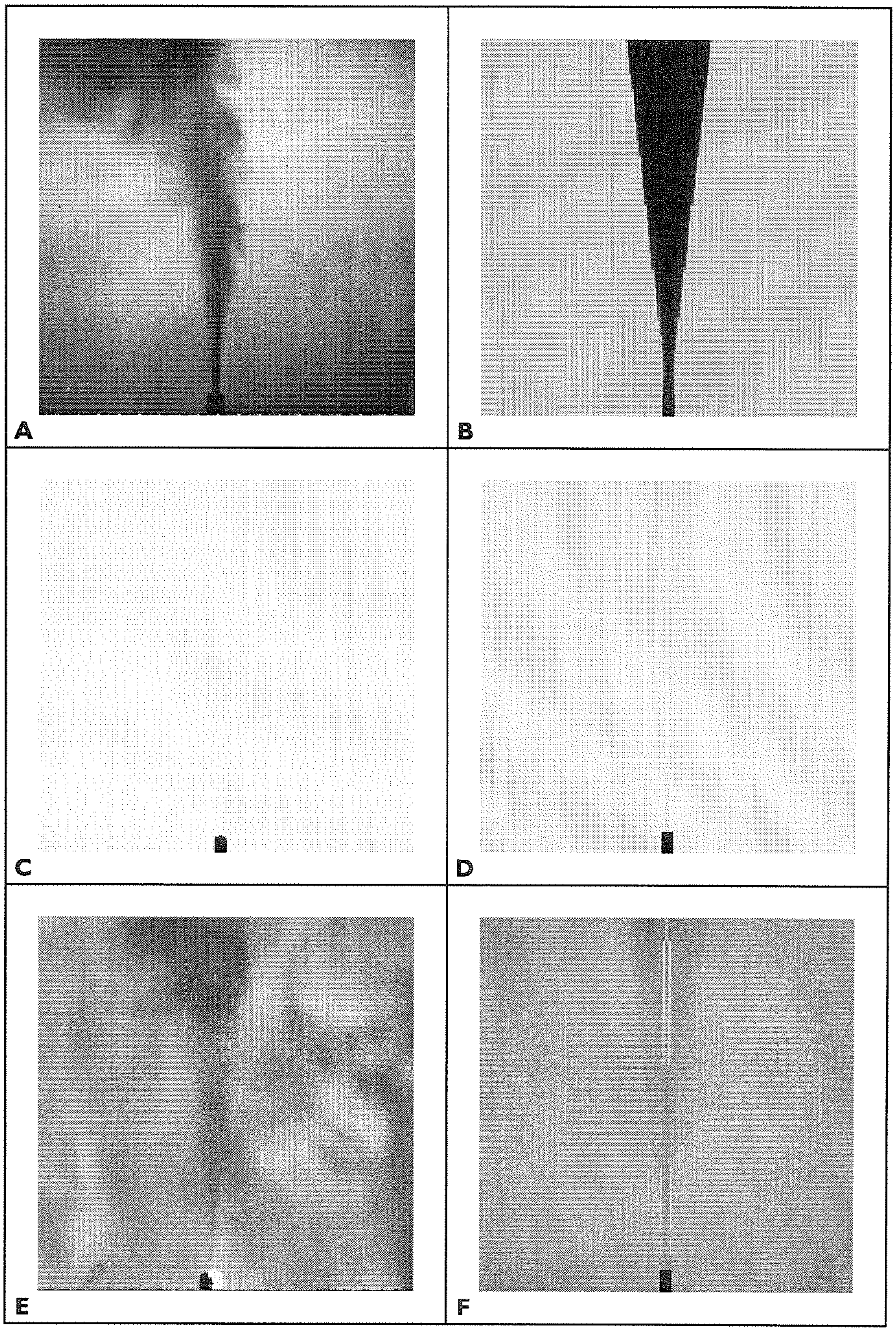


Figure 3.18: Images for the hot plume of methane against a hot background.
(a) CH_4 filter (N03322-8, Pixel Range = 2800 to 3400).
(b) CH_4 filter (N03322-8, Voltage Range = 467mV to 555mV).
(c) Clean filter (N03990-4, Pixel Range = 2900 to 3400).
(d) Clean filter (N03990-4, Voltage Range = 1.3V to 1.45V (offset +150mv)).
(e) CO_2 filter (N04235-4, Pixel Range = -75 to 25 (difference image)).
(f) CO_2 filter (N04235-4, Voltage Range = 1.327V to 1.347V (diff. image)).

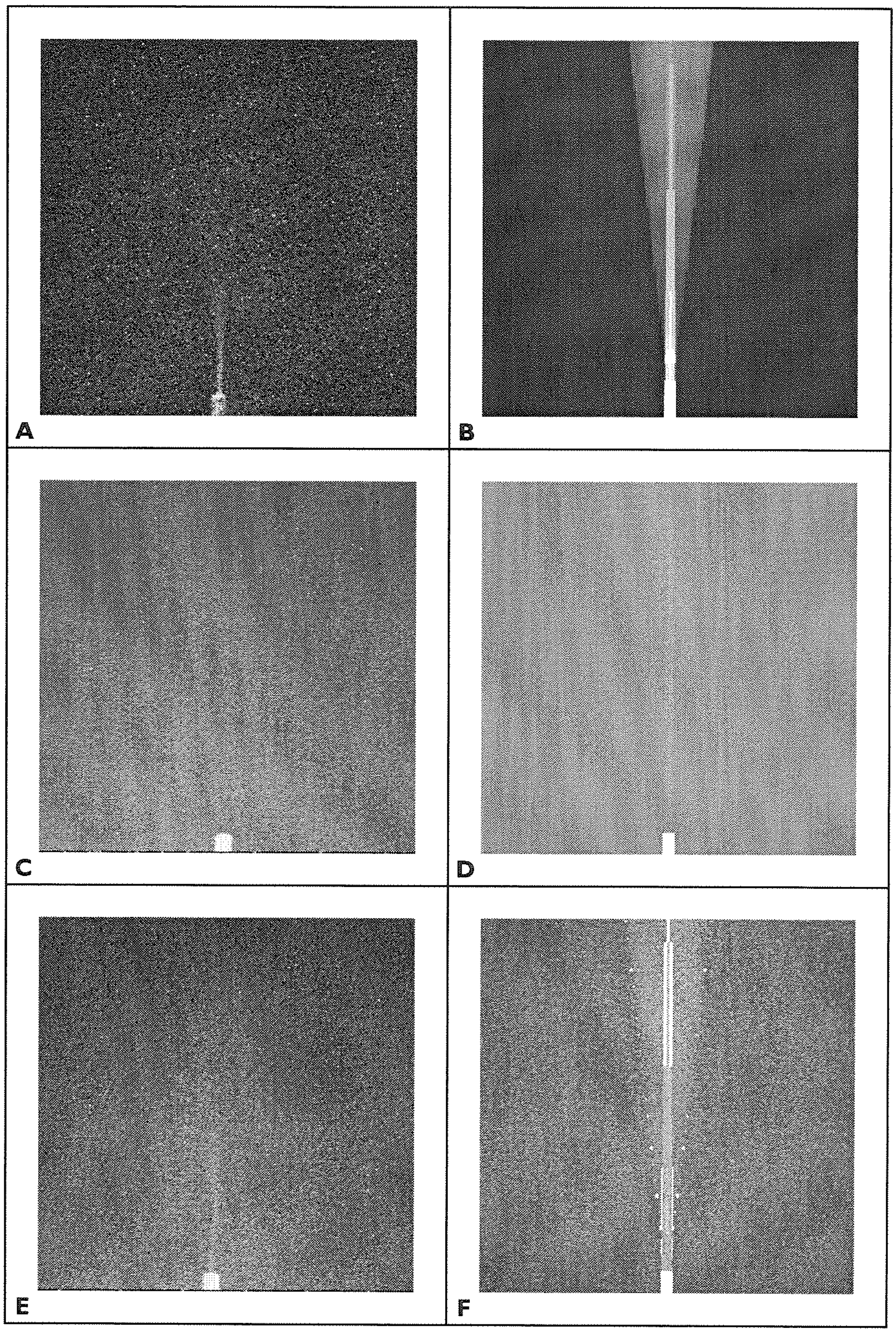


Figure 3.19: Images for the hot plume of methane against a cold background.
(a) CH_4 filter (N03322-8, Pixel Range = 350 to 550).
(b) CH_4 filter (N03322-8, Voltage Range = 132mV to 161mV (offset +25mv)).
(c) Clean filter (N03990-4, Pixel Range = 600 to 700).
(d) Clean filter (N03990-4, Voltage Range = 460mV to 490mV).
(e) CO_2 filter (N04235-4, Pixel Range = 500 to 600).
(f) CO_2 filter (N04235-4, Voltage Range = 701.2mV to 709.1mV).

carbon dioxide filters can isolate and detect a hot plume of each of these gases. The simulated image for the clean filter shows the faintest visible cone. Furthermore, the clean filter in the experiment is not washed out, but still does not appear to exhibit this plume. Clearly, this figure demonstrates the viability of passive detection of gaseous pollutants. However, it does appear that it will be more difficult to determine the concentration of these pollutants. Another important aspect of this figure is that the noise modeling of IRIMAGE appears to closely match the experimental images (especially in the case of the CO_2 filter). All three simulated images are definitely affected by the background fluctuation noise model. Thus, this figure serves as further evidence of IRIMAGE's ability to generate accurate representations of the physical world.

Figures 3.20 and 3.21 also demonstrate that IRIMAGE is capable of reproducing experimental results. Figure 3.20 shows the cold plume of natural gas against a cold background. IRIMAGE's simulated cone model predicts that we should see a faint plume of both CH_4 and CO_2. However, we don't really see this in the experimental images. We believe this may be due to slight discrepancies in the temperature measurements of the gas and the background plate. Since the voltage ranges are very small for all three filters, such minor flucuations could cause the gas to appear in the simulation, but not in the actual experiment. Furthermore, the noise model does appear to be less accurate for filters with small bandwidths in the shorter wavelength regions. Therefore, a signal due to the plume may exist in the experiment, but it is lost in the noise. Fiqure 3.21 looks very similar to 3.18. In this case, the plume is cold. However, the results are nearly the same and one can make the same conclusions as we did for figure 3.18. One interesting note is the minor artifact of the imaging system in 3.18e. Although not visible in the normal images, this artifact appears when we subtract off the standard image. We believe this problem is related to the calibration for this filter. Overall, these two figures further demonstrate the viability of multi-spectral pollution detection and validate IRIMAGE's ability to model such experiments.

In all four cases, IRIMAGE demonstrates that it is capable of accurately modeling

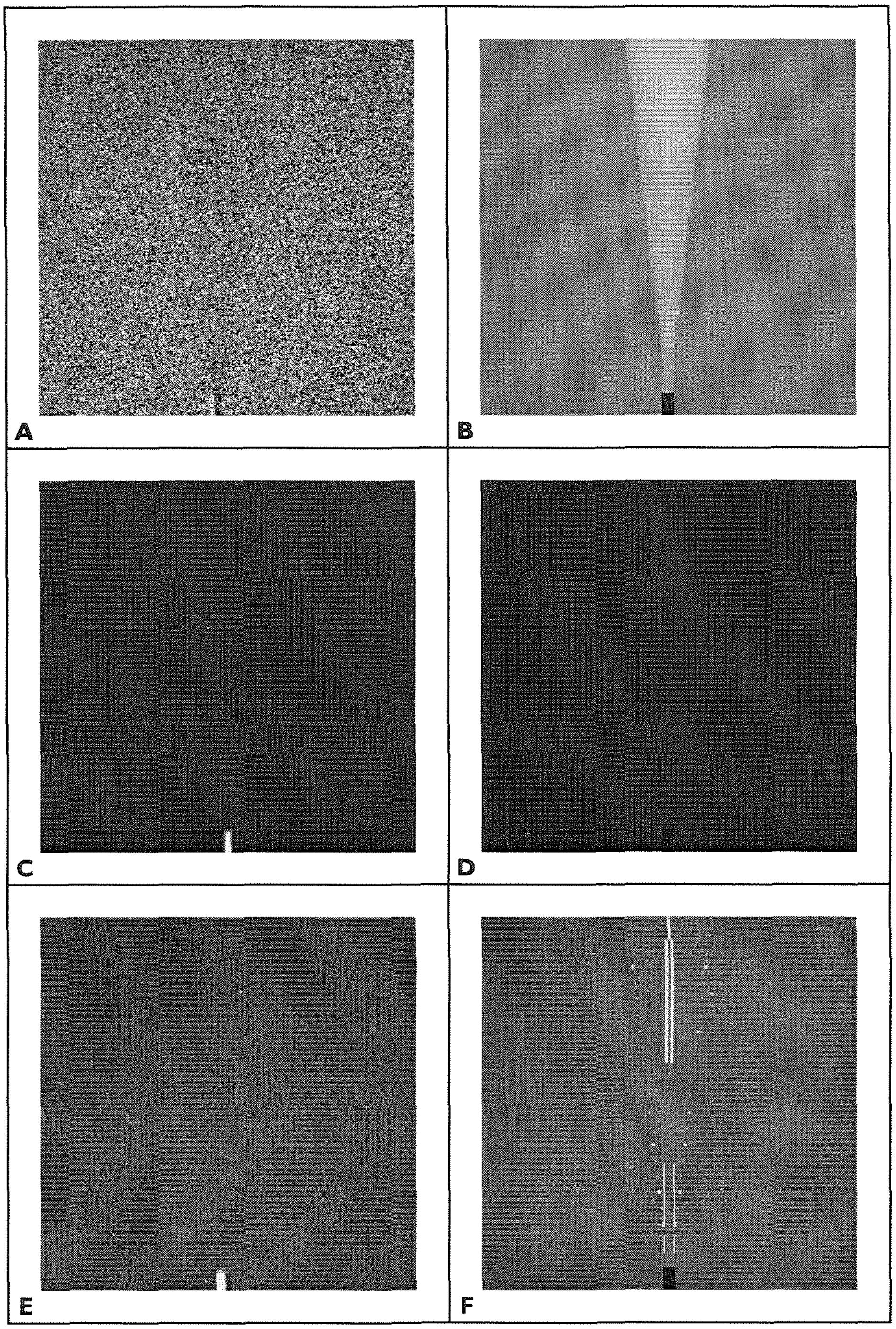


Figure 3.20: Images for the cold plume of methane against a cold background.
(a) CH_4 filter (N03322-8, Pixel Range = 325 to 425).
(b) CH_4 filter (N03322-8, Voltage Range = 103mV to 118mV (offset +30mv)).
(c) Clean filter (N03990-4, Pixel Range = 600 to 700).
(d) Clean filter (N03990-4, Voltage Range = 460mV to 490mV).
(e) CO_2 filter (N04235-4, Pixel Range = 500 to 600).
(f) CO_2 filter (N04235-4, Voltage Range = 701mV to 709mV).

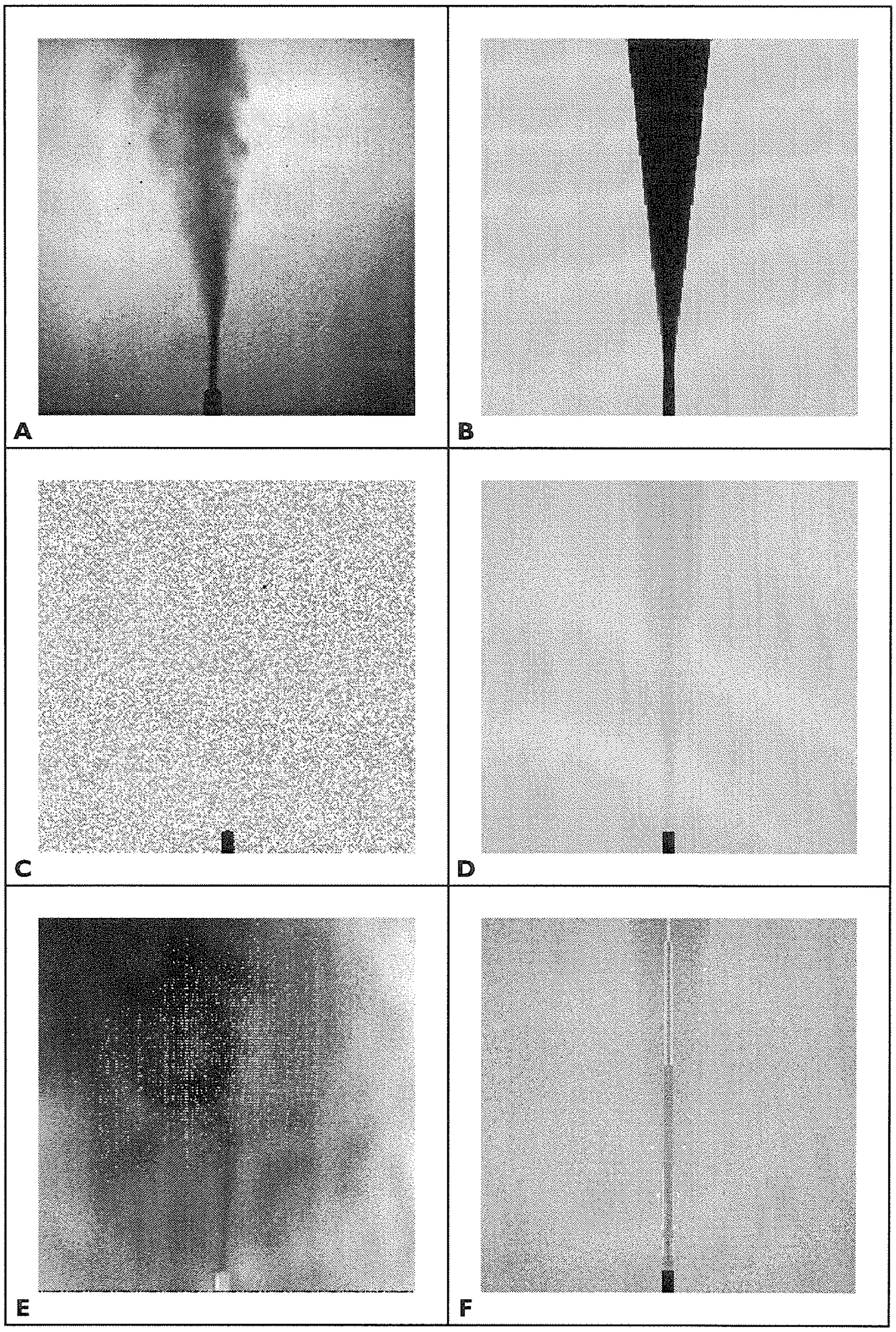


Figure 3.21: Images for the cold plume of methane against a hot background.
(a) CH_4 filter (N03322-8, Pixel Range = 2800 to 3400).
(b) CH_4 filter (N03322-8, Voltage Range = 467mV to 553mV).
(c) Clean filter (N03990-4, Pixel Range = 3650 to 3850).
(d) Clean filter (N03990-4, Voltage Range = 1.4V to 1.46V (offset +25mv)).
(e) CO_2 filter (N04235-4, Pixel Range = -75 to 25 (difference image)).
(f) CO_2 filter (N04235-4, Voltage Range = 1.327V to 1.347V (diff. image)).

a set of atmospheric conditions. It is also clear that the noise modeling and detector modeling generate reasonable results. However, we believe that the noise model does need some improvement. Our assumption has always been that the FPA operated under BLIP conditions. For filters at shorter wavelengths with small bandwidths, the detector noise most probably dominates causing the simulated images to look "cleaner." A further improvement to IRIMAGE will be to incorporate detector and readout noise models. More importantly, the results of this experiment clearly demonstrate that multi-spectral imaging, both passive and simple active, show promise and should be explored further both experimentally and by computer simulation.

3.6 The Gas Cell Experiments

Two of the primary pollutants in automobile exhaust are carbon monoxide (CO) and carbon dioxide (CO_2). According to Seinfeld, an internal combustion engine operating under normal conditions is capable of producing 0 to 10 percent CO and 4 to 14 percent CO_2 in a dry air mixture [9]. Figure 3.22 shows roughly how the concentration of these two pollutants are related with respect to air-fuel ratio. The concentration of CO goes down linearly with respect to air-fuel ratio while the CO_2 concentration increases. The use of a catalytic converter reduces the concentration of CO further to meet most state and federal standards. The 1986 federal standards for CO are 3.4 g/mi for light-duty automobiles and 17 g/mi for light and medium-duty trucks [10]. In the course of our experiments, we discovered that Linde Specialty Gas company provides a series of specialty gases that are used to calibrate automobile emissions testing equipment. The lowest concentration of both gases is a mixture of 1.6% CO and 10% CO_2 [11]. The intent of the Gas Cell Experiment is to emulate a car exhaust plume (for an idling vehicle) on a small scale and demonstrate that it is possible to measure CO and CO_2 independently using a set of small narrow band-pass filters. At the same time, the experiments also provide a real world test of the abilities of IRIMAGE to accurately generate images that are comparable with the results from an experiment.

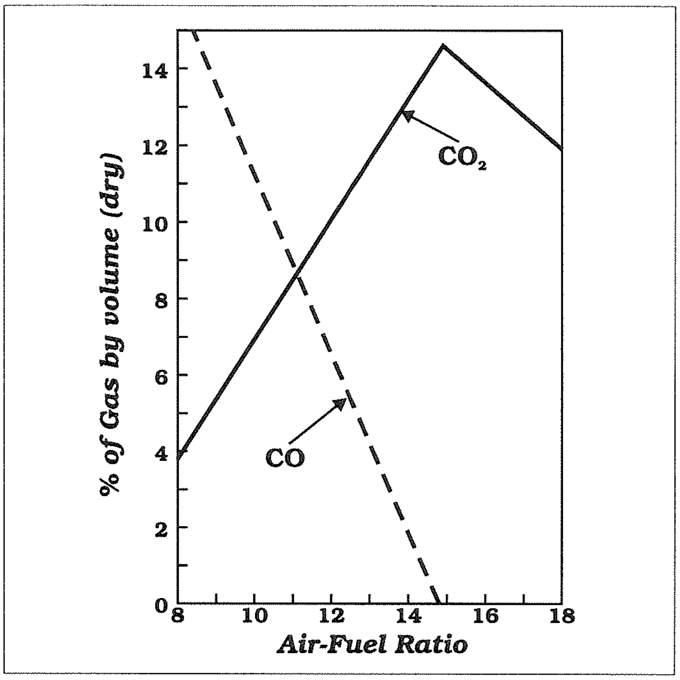


Figure 3.22: Relationship between CO and CO_2 in the exhaust of internal combustion engine. [9]

Because of the hazardous nature of these experiments, we designed and built a gas cell that would contain the exhaust plume of CO and CO_2 and prevent the gas from mixing with the atmosphere of the laboratory. For each experiment, we purged the cell with dry nitrogen to create a clean atmosphere. Once the cell was filled with a dry nitrogen atmosphere, we introduced hot and cold plumes of carbon monoxide, carbon dioxide and a mixture of both against either a hot or cold background. Using the lessons learned from the methane experiments, this cell uses the same copper tubing with the water bath to generate a heated plume. The GCE also uses the calibration plates as the backgrounds for these experiments. Using methods similar to those in the methane simulations, IRIMAGE models the atmospheric conditions using two cylinders. These cylinders are linked to atmosphere models which define different temperatures and concentrations of CO and CO_2. Besides further verifying IRIMAGE's abilities, this experiment investigates the viability of imaging a polluted plume using an imaging system equipped with several narrow bandwidth filters.

Using three narrow filters, we isolated the CO and the CO_2 bands as well as a

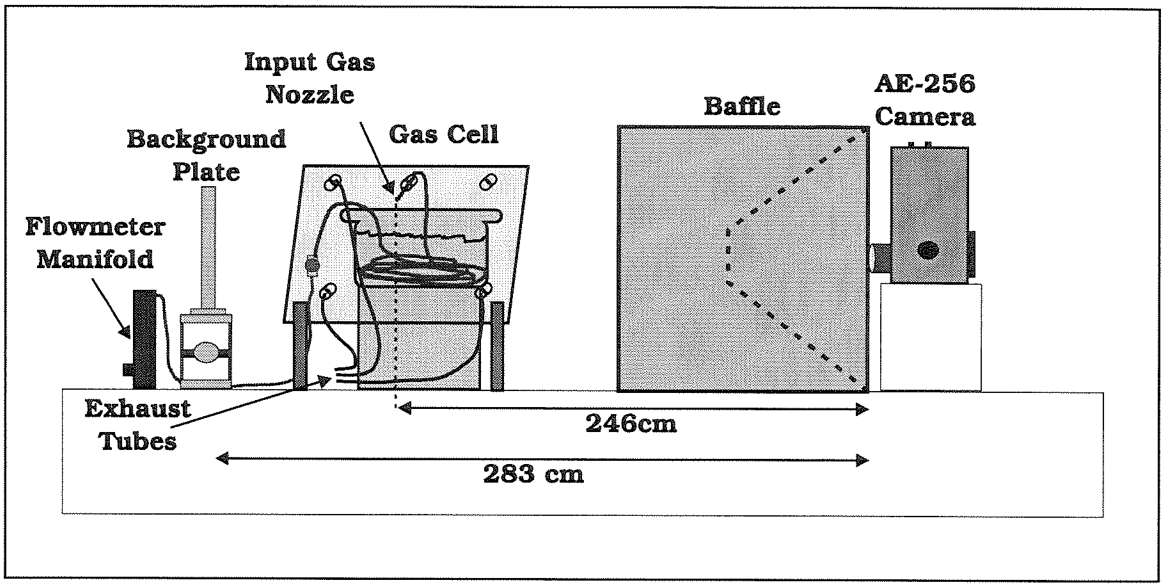


Figure 3.23: The Gas Cell experimental setup.

realtively clean band (not in CO or CO_2 band). In the gas cell experiments, we investigated three plume types using these three filters:

- A plume of 10% CO_2 and 90% N_2.
- A plume of 1% CO and 99% N_2.
- A plume of 1% CO, 6.6% CO_2 and 92.4% N_2 (automobile emission).

Similar to the methane experiments, we imaged a cold plume against a hot background and a hot plume against a cold background. Unfortunately, certain conditions regarding our gas cell windows made it difficult to generate decent images in any of these cases. Therefore, in order to compare the simulated images to a set of experiments, we used a simple subtractive image processing technique to improve the experimental images.

3.6.1 Experimental Setup

As shown in figure 3.23, the experimental setup for the GCE is similar to the methane setup except for the introduction of the gas cell. The background source is either the 12"x12" Cold plate or the 12"x12" Hot plate. The size of the gas cell and the desire to

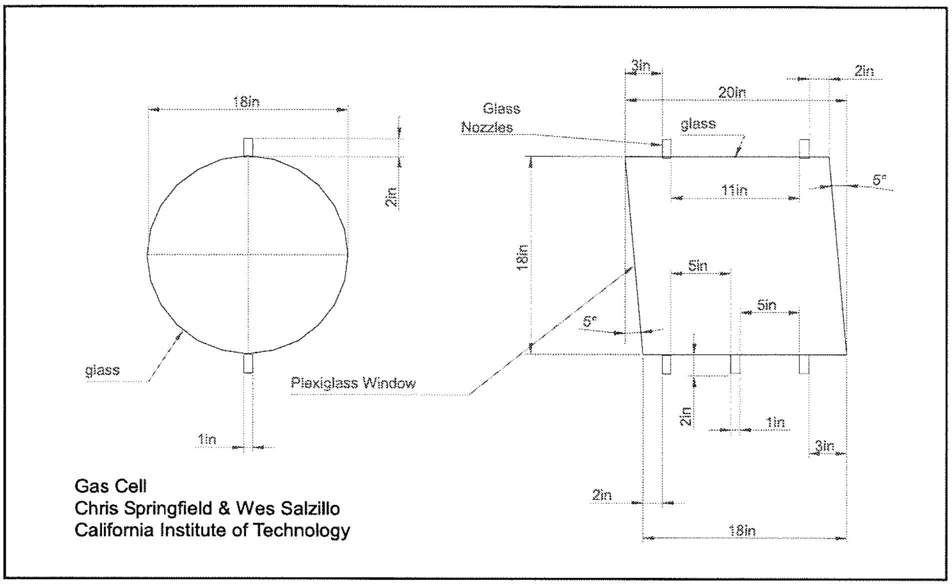


Figure 3.24: Drawing of gas cell design.

minimize certain depth of field problems made it necessary to place the source plate a distance of 2.83 meters from the imaging system. As in the previous experiments, these plates serve as both the background sources and the calibration sources.

The gas cell provides a controllable atmosphere that allows us to investigate hazardous pollutants like carbon monoxide. The gas cell is a glass tube that is 50 cm long and 43.8 cm in diameter with walls that are 3/8" thick (see figure 3.24). There are five ports on the cell. The four outer ports can be used for exhausting the gas inside the gas cell to the outside (we used three of them and plugged the fourth). The middle port is for the gas input tube and the gas thermocouple. The gas input tube is the same copper tube used in the methane experiment, submerged in a water bath to regulate the temperature of the gas flow. Since there is long portion of the tube between where it exits the water bath and where it ends inside the gas cell, we applied a surface heater to the end of the tube to prevent the heated gas from cooling before it enters the gas cell. All of the ports are sealed by rubber stoppers which have pass through holes for the exhaust or input tubes. In addition, the stopper in the middle port has a second hole for passing the gas thermocouple into the gas cell. Using the gas thermocouple, we are able to measure the ambient temperature of the cell as well as the gas plume temperature.

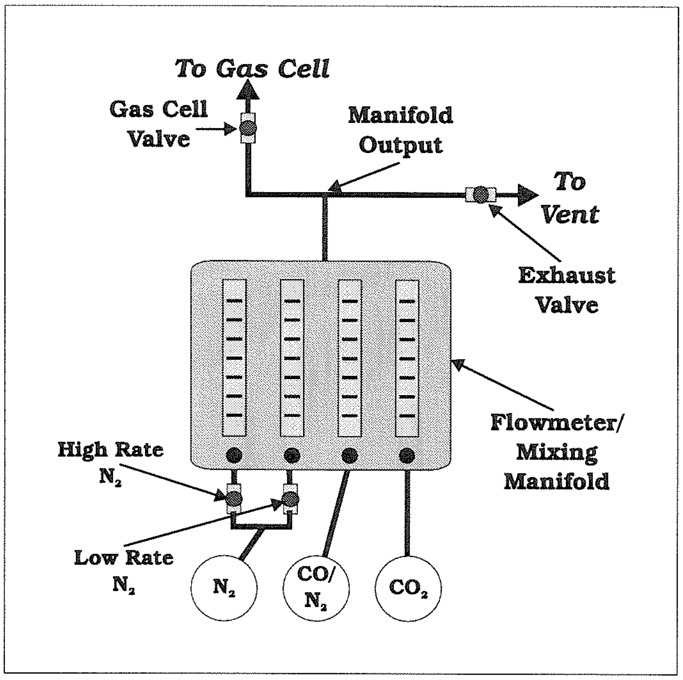


Figure 3.25: Schematic of gas mixing setup.

The CO, CO_2 and N_2 are mixed using a standard gas mixing manifold. This manifold contains four individual flow meters which have separate inputs but have a common output through the mixing manifold. In order to form our three plumes and purge the polluted atmosphere between experiments, these individual flow meters are connected to a cylinder of dry N_2, a cylinder of dry CO_2 and a cylinder of 3% CO/97% N_2. The N_2, CO_2 and the CO/N_2 mixture are each connected to separate flow meters; the N_2 is connected to a high-rate flow meter for purging the gas cell and a low-rate flow meter for use in mixing the other two gases to the appropriate concentrations. By controlling the flow of each of the gases into the manifold, we can accurately set the plume's exit concentration.

In order to avoid reflections of the camera and the interior of the cell from the windows back into the camera, the ends of the gas cell were ground to an angle of 5°. Initially, the windows were made of 1/16 inch thick Plexiglas. Plexiglas is approximately 88% transmissive over the 3-5 micron range and has the strength to avoid flexing during an experiment. Unfortunately, the camera was unable to see a 10% CO_2 plume through the windows. Since the methane experiment demonstrated that a plume would be visible for concentrations above .2appeared in the images. For this reason, we replaced the Plexiglas windows with a polyethelyne-based plastic wrap stretched over the two ends of the gas cell. With these makeshift windows, we were able to see the CO_2 plume and keep a tight seal to prevent the gas from leaking out of the gas cell. Although more transmissive, these new windows introduced another set of problems. Because the plastic wrap is so thin and capable of stretching and flexing, it exhibits a certain amount of non-uniform transmission. In addition, varations of the pressure in the cell caused the windows to bow in and out, resulting in unwanted reflections of the cell and the end of the copper tube.

Although the windows do introduce unwanted reflections, we constructed a baffle which would minimize the reflections for any surrounding objects (lab bench, camera case, other lab equipment, etc.). As shown in figure 3.23, the baffle is a cardboard box which is painted black on the inside and contains a pyramid shaped set of walls inside. This shape prevents most direct radiation and reflections from entering the optical system and thus reduces the unwanted reflections to a much smaller number of sources.

3.6.2 Experimental Procedure

Calibration Procedures

The camera calibration procedure is exactly the same as in the previous two experiments (See section 3.5). For this experiment, we imaged the CO_2 plume through all three filters before moving on to the CO and the mixture plumes. After we completed the CO_2 plume experiments, we decided that it was unnecessary and less accurate to examine only one type of plume at a time. Therefore, the CO and the CO/CO_2 mixture experiments were both done for the various backgrounds and gas temperatures before changing the filters. In order to minimize the difference between the CO_2 experiments and the other two experiments, we made every effort to duplicate the conditions for the camera and image calibration.

Again, the image calibration was very similar to the methane case. Once the camera was calibrated for a filter, we captured the hot and cold calibration images. We did not capture different calibration images for each different background or plume concentration. Instead, we maintained the same global gain and offset for every experiment conducted using a specific filter. Like the camera calibration, we made every effort to duplicate the image calibration conditions of the CO_2 experiment with the conditions for the CO and gas mixture experiments.

Purging Gas Cell

Prior to any experiment, the gas cell is purged of all of the pollutants introduced in the previous experiment. The following procedure describes this purging process.

1. Check to make sure the CO_2 and CO cylinders are closed.
2. Make sure water bath is filled with cold water ($\approx 15\,^{\circ}\mathrm{C}$).
3. Close the valve to the gas cell and open the direct exhaust valve.
4. Purge the manifold with N_2 at a rate of 7 l/min for 1 minute.
5. Open the gas cell valve and close the direct exhaust valve.
6. Purge the gas cell at the same N_2 flow rate for 20-30 minutes.
7. Check for CO/CO_2 levels using combustion analyzer.
8. Continue purge until CO is under 15 ppm and CO_2 is under 1%.

Once a purge cycle is complete, the next experiment can be conducted.

Imaging a Hot Plume Against a Cold Background

For each filter and polluted plume, the procedure for generating an image of a hot plume against a cold background is the same. The following procedure describes how we setup the experiment and captured the standard and plume images for this case.

1. Flow N_2 at approximately 2 l/min through gas cell.
2. Place the Cold background plate at 283 cm from the lens and centered on the optical axis. Measure its temperature.

3. Measure ambient temperature inside the cell.
4. Capture 15 frames for use as the standard in the image difference computation.
5. Close the valve between the gas cell and the manifold.
6. Fill the water bath with hot water which is approximately 320K.
7. Apply 100mA of power to heater at end of copper tube.
8. Open the direct exhaust valve and turn off the high flow N_2 valve.
9. Set the flow rates of N_2, CO and CO_2 to match desired concentration percentages with gas flowing out the direct exhaust tube.
10. Once the flows have stabilized, open the valve to the gas cell and close the direct exhaust valve.
11. Capture 30 frames of gas plume.
12. Measure the plume temperature with the gas thermocouple.
13. Close the valve to the gas cell and close CO and CO_2 cylinder valves.
14. Begin purge process.

Imaging a Cold Plume Against a Hot Background

Once all of the images of the hot plumes against a cold background have been captured for the current filter, we repeat the same procedure for cold versions of the same plumes against a hot background. For these plumes, the water must be at 290K and the heater at the end of the tube must be off. In addition, we replace the cold background with the hot background. Other than these changes, the procedure is the same as the hot plume against a cold background.

3.6.3 Simulation Setup

Like the methane experiment, tables 3.10 through 3.12 contain the camera settings, background temperature measurements and gas parameters for each of the three filters. Since several parameters are the same for the hot plume/cold background and cold plume/hot background cases, we placed the parameters for both cases in the same column of these tables. Using these parameters, IRIMAGE models the background and calibration images as well as the atmospheric conditions and the imaging system for each filter. The coordinates of the imaging VGO are (0m, 0m, 2.83 m) in the WCS while the end of the copper tube in the gas cell is at an angle of 6 degrees below the horizontal (X-axis) and is at the WCS coordinates (-.127m, .09m, 2.464m). Although the tube is barely visible in the experimental data, we have ignored its presence in the simulations because we were unable to acquire its surface temperature.

Parameter	***10% CO_2***	***1% CO***	***6.6% CO_2, 1% CO***
Hot Plate T	319.45 K	319.55 K	319.55 K
Cold Plate T	294.65 K	294.75 K	294.75 K
Integration Time	5.21 ms	5.21 ms	5.21 ms
Frame Rate	30 Hz	30 Hz	30 Hz
H/C: T_{source}	319.45 K	319.55 K	320.25 K
H/C: T_{gas}	294.65 K	294.15 K	294.60 K
H/C: $T_{ambient}$	293.65 K	294.00 K	294.45 K
C/H: T_{source}	294.65 K	294.05 K	293.85 K
C/H: T_{gas}	299.95 K	301.95 K	301.95 K
C/H: $T_{ambient}$	295.95 K	295.25 K	295.40 K

Table 3.10: Parameters for CO and CO_2 experiments using the filter N04235-4.
($\lambda_{central} = 4.235$ μm, FWHM = .181 μm, $\tau_{peak} = .870$)
(C/H: Cold Background, Hot Plume - H/C: Hot Background, Cold Plume)

IRIMAGE represents a static approximation of the gaseous plumes of CO, CO_2 or a mixture of both using the same two cylinder construct as in the methane experiment. The small cylinder contains 100% of original concentration of the pollutant (CO, CO_2 or mixture) while the larger cylinder is reduced to 25% of the original pollutant concentration. The physical dimensions of the two cylinders vary slightly from the

Parameter	*10% CO_2*	*1% CO*	*6.6% CO_2, 1% CO*
Hot Plate T	316.45 K	316.75 K	316.75 K
Cold Plate T	293.35 K	294.70 K	294.70 K
Integration Time	32.81 ms	32.81 ms	32.81 ms
Frame Rate	30 Hz	30 Hz	30 Hz
H/C: T_{source}	316.45 K	316.75 K	316.80 K
H/C: T_{gas}	294.05 K	294.95 K	295.00 K
H/C: $T_{ambient}$	293.55 K	294.95 K	295.05 K
C/H: T_{source}	293.35 K	295.50 K	295.00 K
C/H: T_{gas}	300.95 K	301.45 K	301.25 K
C/H: $T_{ambient}$	293.85 K	295.65 K	295.65 K

Table 3.11: Parameters for CO and CO_2 experiments using the filter N03689-4H. ($\lambda_{central}$ = 3.686 μm, FWHM = .042 μm, τ_{peak} = .727)
(C/H: Cold Background, Hot Plume - H/C: Hot Background, Cold Plume)

Parameter	*10% CO_2*	*1% CO*	*6.6% CO_2, 1% CO*
Hot Plate T	318.75 K	318.15 K	318.15 K
Cold Plate T	294.65 K	294.25 K	294.25 K
Integration Time	5.21 ms	5.21 ms	5.21 ms
Frame Rate	30 Hz	30 Hz	30 Hz
H/C: T_{source}	318.75 K	318.15 K	318.15 K
H/C: T_{gas}	295.25 K	294.75 K	294.65 K
H/C: $T_{ambient}$	295.35 K	294.25 K	295.75 K
C/H: T_{source}	293.75 K	293.75 K	293.95 K
C/H: T_{gas}	302.15 K	303.35 K	302.05 K
C/H: $T_{ambient}$	294.25 K	296.85 K	295.45 K

Table 3.12: Parameters for CO and CO_2 experiments using the filter N04693-4. ($\lambda_{central}$ = 4.693 μm, FWHM = .167 μm, τ_{peak} = .785)
(C/H: Cold Background, Hot Plume - H/C: Hot Background, Cold Plume)

methane experiment. The small cylinder is 25cm long with the end closest to the nozzle having a radius of 4mm while the other end has a radius of 2mm. The larger cylinder is also 25cm long, but the radius nearest to the nozzle is only 2mm while the other end has a radius of 30mm. We approximate the orientation of the exhaust cone by rotating these two cylinders so that their centerlines lie in the XY plane of the WCS and are rotated around the Z axis at an angle of 6°. The center point of both cylinders is placed at (.01m, .085m, 2.464m) in the WCS.

The background scene is modeled just using the standard image used for the calibration plates (See figure 3.10). Since we are not modeling the end of the exhaust tube in this case, the Scene object only loads a single image. This image either represents the hot background plate or the cold background plate. Each simulation sets the plate temperature based on the experimentally measured temperature while the emissivity is set to lie between .88 and .885.

IRIMAGE is unable to simulate the problems related to the thin plastic windows. This would require modeling a spatially variant filter outside of the imaging system. Unfortunately, IRIMAGE can only simulate a partially transmissive surface on the imaging VGO outside of the imaging system (Scene object using a soft matte). Therefore, these problems must be ignored in the simulations. This factor must be kept in mind during the image comparisons that follow in the next section.

3.6.4 Comparison of Results

The primary goal of these experiments was to determine the effectiveness of detecting carbon dioxide and carbon monoxide using passive multi-spectral IR imaging. We present the results of the experiments and subsequent simulations here. We used a three filter setup so that we could isolate the CO and CO_2 bands from each other. The third filter is a clean filter which allows us to discriminate the background sources from the plume. For each plume type (CO and CO_2 concentration) and filter, we only looked at the two cases of a hot plume against a cold background and cold plume against a hot background. These two cases represent the extremes that one might encounter when passively imaging a plume of polluted gas. Since the validity of IRIMAGE has been demonstrated by the previous two experiments, the comparisons discuss the simulated results as predictions of what the expected images should be. This discussion will demonstrate that passive imaging of these pollutants is possible but does face certain problems.

As mentioned above, the windows of the gas cell created several problems with detecting the plumes. This made comparing experiment images with simulated images very difficult because IRIMAGE was unable to simulate these problems. The

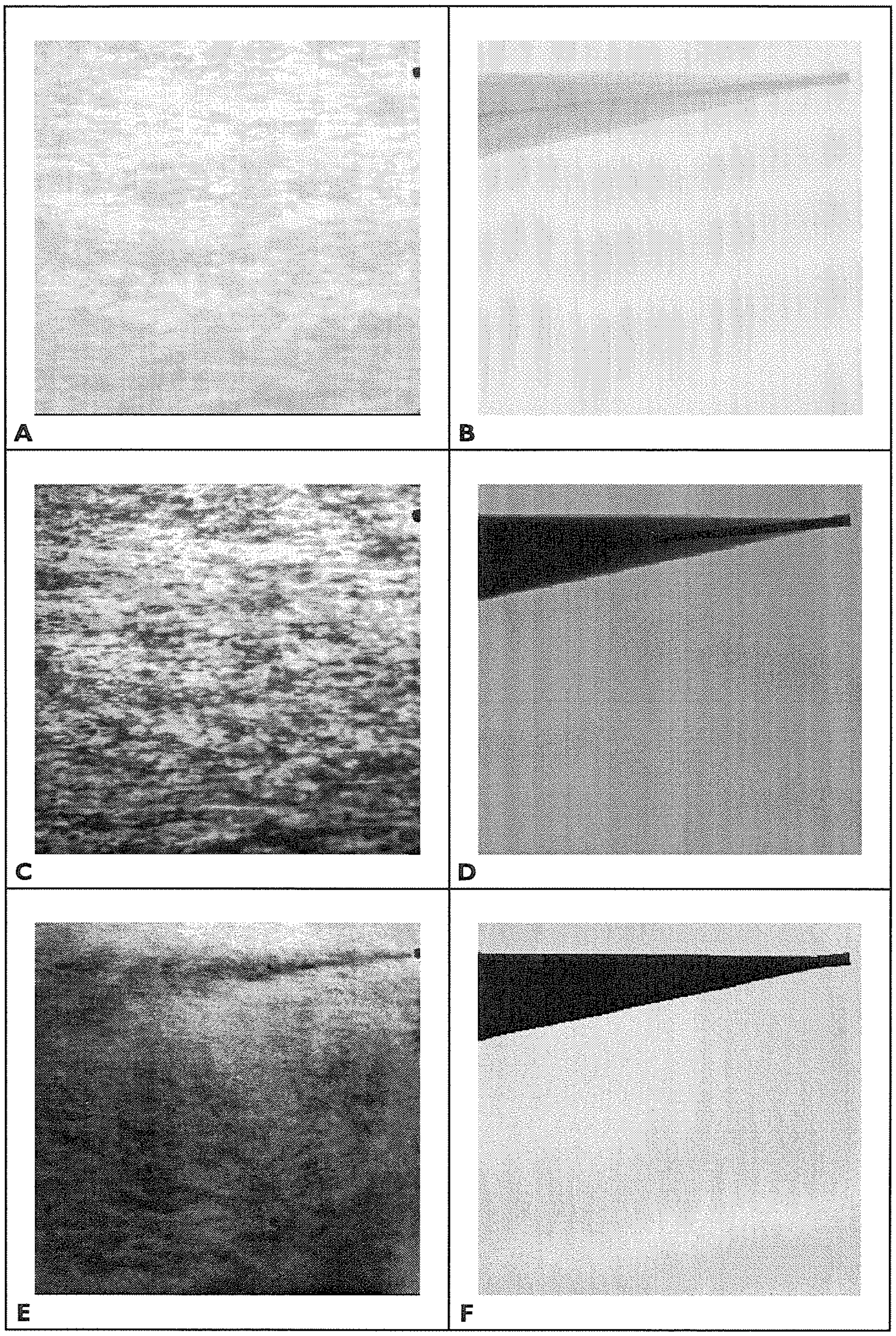


Figure 3.26: Images of a cold plume of CO_2 against a hot background looking through the N0-4235 filter. Demonstrates the problems caused by the gas cell windows and shows the image processing solution.
(a) Expt: Full Range = 0 to 3800. (b) Sim: Full Range = 372.5mV to 740mV.
(c) Expt: Nar. Range = 2800 to 3800. (d) Sim: Nar. Range = 643.3mV to 740mV.
(e) Expt: Diff. Range = -25 to 175. (f) Sim: Diff. Range = 689.2mV to 706.6mV.

technique of enhancing the captured images by reducing the size of the range and its starting point failed to generate images that clearly showed the plumes. However, using a standard image processing technique of subtracting a standard image (no polluted plume) from these images, we were able to significantly reduce the effects of the windows. Figure 3.26 shows the experimental and simulated images of a plume of 10% CO_2 over the full range, a narrow range and the image processed range of values. Although the IRIMAGE cone model predicts that the plume should be visible in all three ranges, the problems with the windows mask out any noticeable change in the full and narrow range images. But the image processed version does show the expected CO_2 plume. In most cases, the expected plumes became easier to see in the image processed images. Some of these plumes were only visible when watching a series of frames and seeing changes in the area of the image that we expected the plume. Therefore, in this discussion, there may be images where a plume is expected but not noticeable. For those cases, we will point out if a plume was visible when looking at series of frames. All of the experimental images presented have been processed by this subtraction technique. The simulated images use a voltage range consistant with the range of the experimental image and an offset to make the simulated image match the overall appearance of the experimental image. In all of these figures, the experimental images are on the left and the simulated images are on the right.

Figures 3.27 through 3.29 present the images of cold versions of the three plume types against a hot background. By looking at the results of all three experiments together, we can make some simple observations about each experiment and also compare them together. In figure 3.27, the plume consists of 10% CO_2 and 90% N_2. We can see that IRIMAGE predicts that the plume will only be visible through the CO_2 filter. The images from the experiment agree with this prediction. A clear plume of carbon dioxide does appear to be flowing out of the nozzle in figure 3.27c, but nothing seems to be visible through the other two filters. For the case of a plume of 1% CO in figure 3.28, IRIMAGE predicts that we should see a plume of CO that is fainter than the CO_2 plume. We also note that the MODTRAN artifacts in 3.28d may indicate that the CO band is slightly visible through the CO_2 filter. An

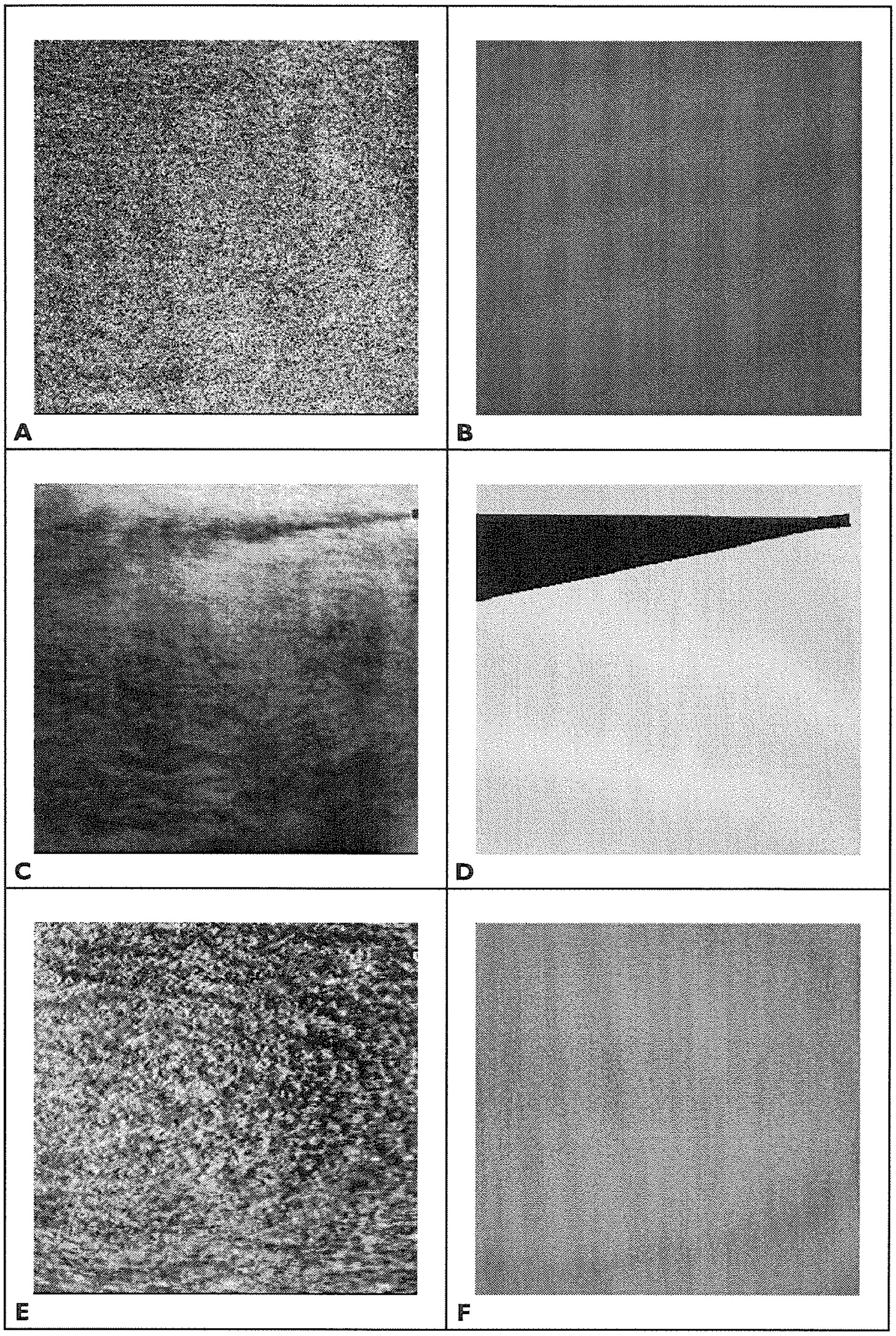


Figure 3.27: Images of a cold plume of 10% CO_2 against a hot background.
(a) Clean filter (N03689-4H, Pixel Range = 0 to 150).
(b) Clean filter (N03689-4H, Voltage Range = 481mV to 494.1mV).
(c) CO_2 filter (N04235-4, Pixel Range = -25 to 175)
(d) CO_2 filter (N04235-4, Voltage Range = 689.2mV to 708.6mV).
(e) CO filter (N04693-4, Pixel Range = -75 to 75).
(f) CO filter (N04693-4, Voltage Range = 1.519V to 1.557V).

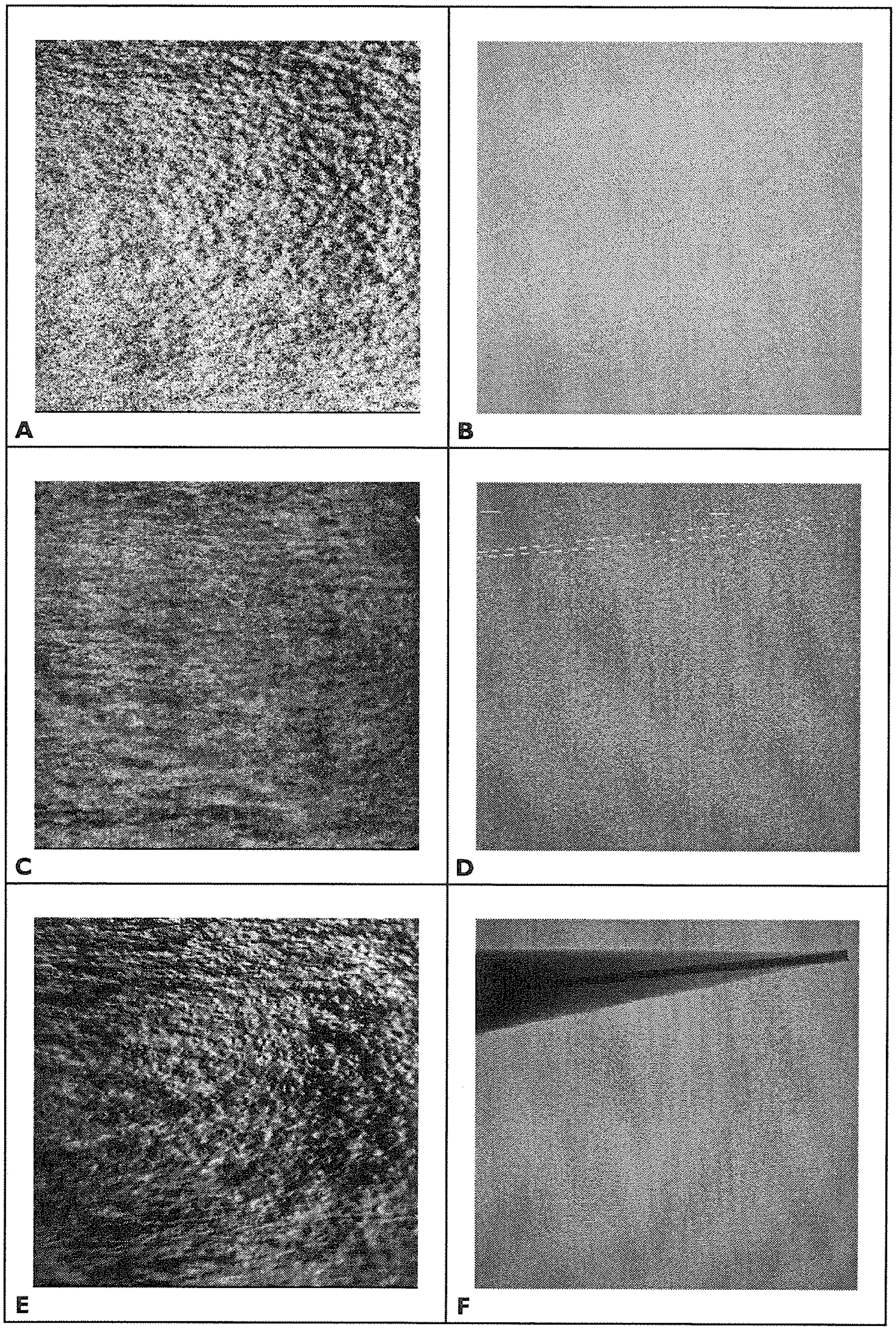


Figure 3.28: Images of a cold plume of 1% CO against a hot background.
(a) Clean filter (N03689-4H, Pixel Range = -150 to 0).
(b) Clean filter (N03689-4H, Voltage Range = 483.1mV to 496.3mV).
(c) CO_2 filter (N04235-4, Pixel Range = 70 to 170).
(d) CO_2 filter (N04235-4, Voltage Range = 705mV to 714.4mV).
(e) CO filter (N04693-4, Pixel Range = -25 to 75).
(f) CO filter (N04693-4, Voltage Range = 1.508V to 1.534V).

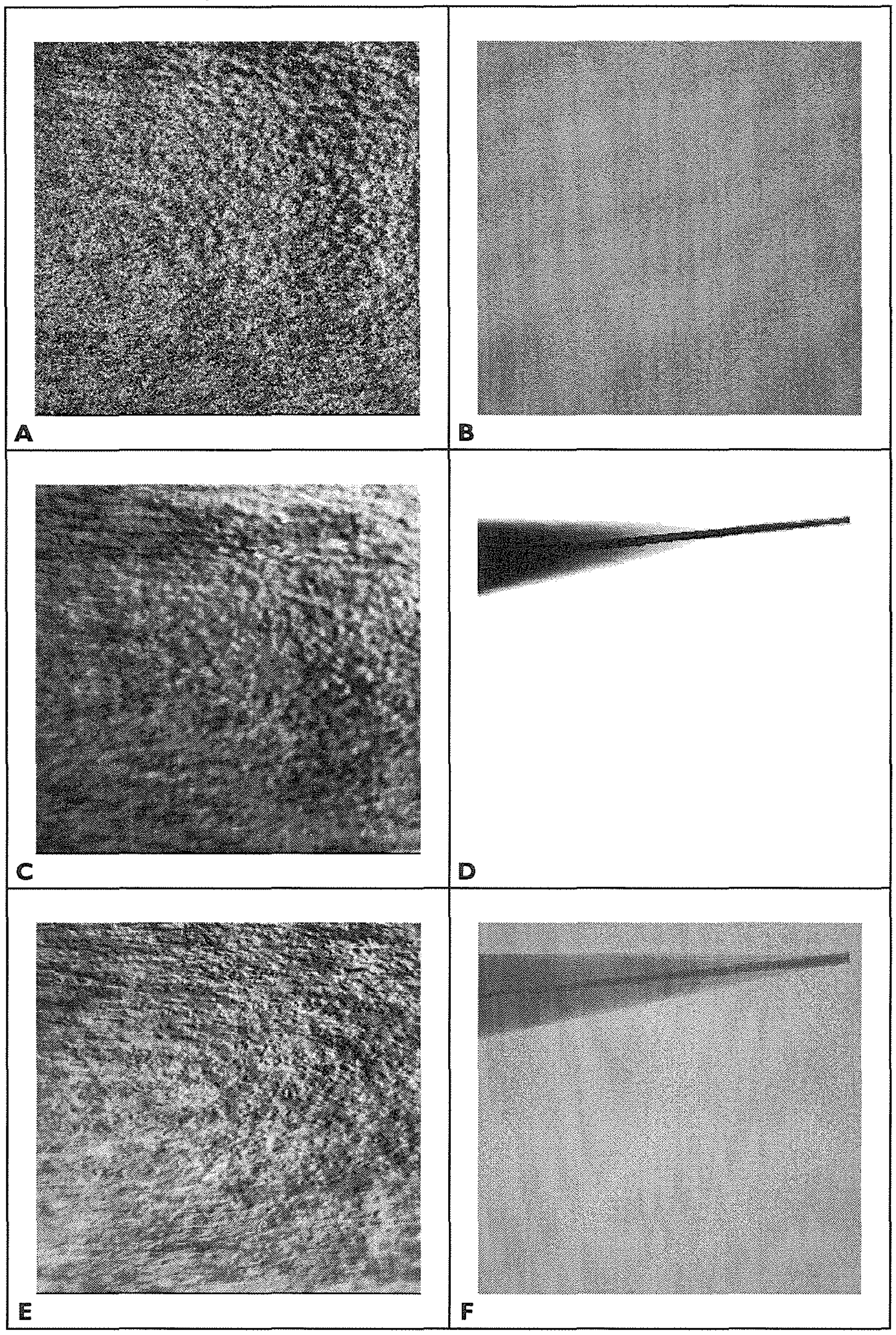


Figure 3.29: Images of a cold plume of 6.67% CO_2/1% CO against a hot background.
(a) Clean filter (N03689-4H, Pixel Range = -75 to 75).
(b) Clean filter (N03689-4H, Voltage Range = 485.3mV to 498.4mV).
(c) CO_2 filter (N04235-4, Pixel Range = -225 to -25).
(d) CO_2 filter (N04235-4, Voltage Range = 677.8mV to 697.3mV).
(e) CO filter (N04693-4, Pixel Range = -200 to 0).
(f) CO filter (N04693-4, Voltage Range = 1.486V to 1.540V).

important result here is that the experimental results seem to indicate that the CO plume is not visible through the CO filter. However, this occurs because the image processing is unable to eliminate the effects of the windows without washing out the plume. By watching a set of 30 captured frames, the CO plume becomes noticeable. Unfortunately, a single image is not enough for this case. As predicted, the plume is not visible through the other two filters. As a result of these first two experiments and the subsequent simulations, it is clear that the CO and CO_2 filters are capable of isolating the two consituents from each other while the clean filter prevents any part of the background source from being mistaken as part of the plume.

For the case of a plume that may be a mixture of both gases, it should be possible to determine the existence and relative concentrations of the two gases based on the above results. The experimental and simulation results presented in figure 3.29 are from an example of such a case. In this experiment, the plume consists of nitrogen with 6.67% CO_2 and 1% CO. IRIMAGE predicts that both a plume of CO_2 and CO should be visible. Experimentally, the results are less clear. A faint CO_2 plume is visible through the CO_2 filter, but again the CO plume is not visible in a static image. Like before, the CO plume is only visible when watching an animation of the 30 captured frames. The clean filter does remain clean, which confirms this plume is not part of the background. We conclude that for the case of a hot plume against a cold background, it is possible to determine the existence of both CO and CO_2 using multi-spectral methods. However, further experimental work must be done to determine the ability to accurately measure the concentration for cases like this.

Figures 3.30 through 3.32 present the imaging results for the other case; a hot plume containing CO, CO_2 or a mixture of both against a cold background. While the previous case relied on the absorption of the background radiation by the plume, this case relies on the emission of radiation by the CO and CO_2 in the various plumes. Before discussing these three experiments, it must be noted that a reflection of the nozzle is visible for all three filters and should be ignored (one of the reasons for the clean filter). In addition, the edge of the gas cell and other reflections are visible in the CO filter; these should also be ignored. In figure 3.30, the plume consists of a

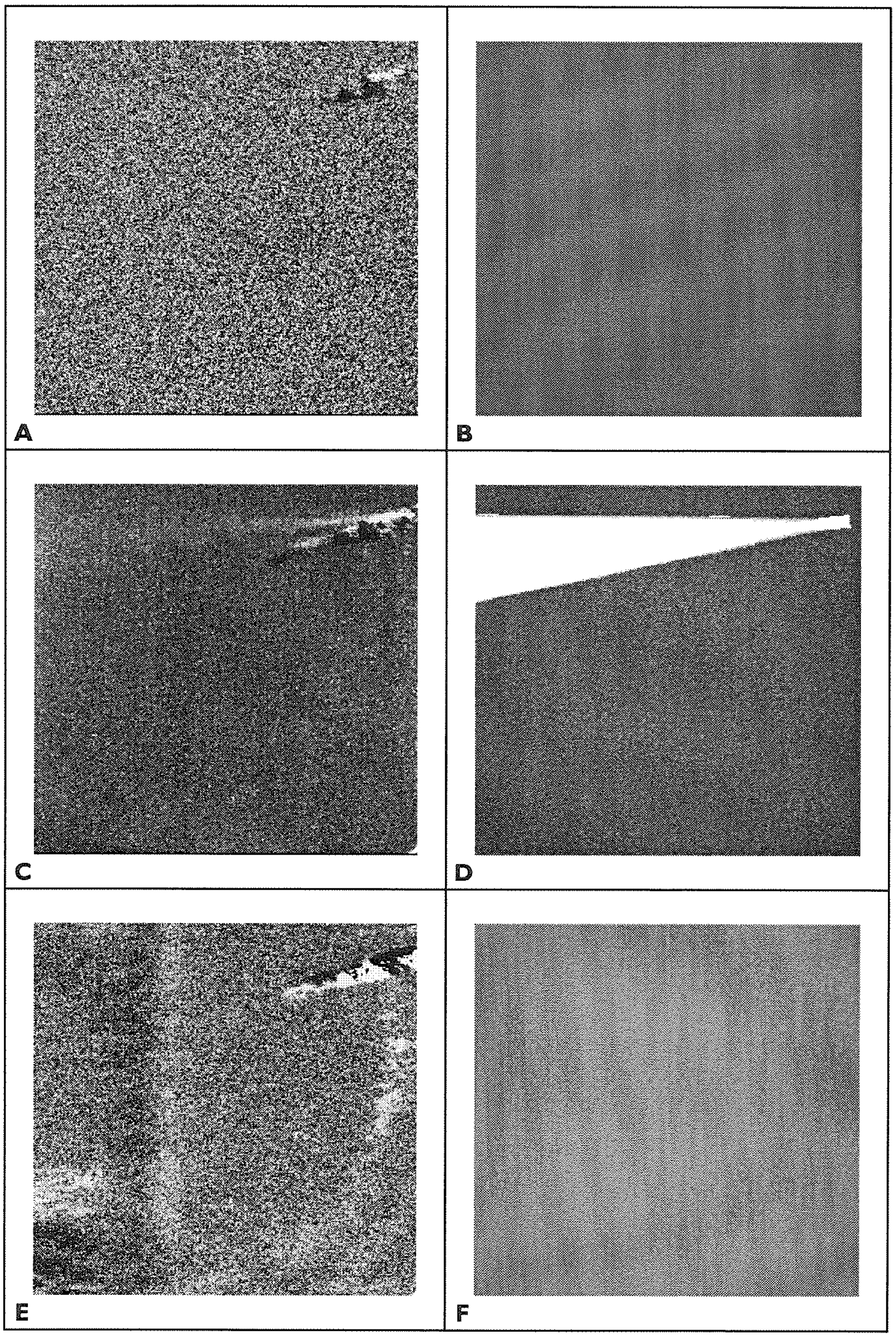


Figure 3.30: Images of a hot plume of 10% CO_2 against a cold background.
(a) Clean filter (N03689-4H, Pixel Range = -50 to 50).
(b) Clean filter (N03689-4H, Voltage Range = 180.3mV to 189.1mV).
(c) CO_2 filter (N04235-4, Pixel Range = -10 to 35).
(d) CO_2 filter (N04235-4, Voltage Range = 409.3mV to 413.6mV).
(e) CO filter (N04693-4, Pixel Range = 0 to 100).
(f) CO filter (N04693-4, Voltage Range = 670.5mV to 696mV).

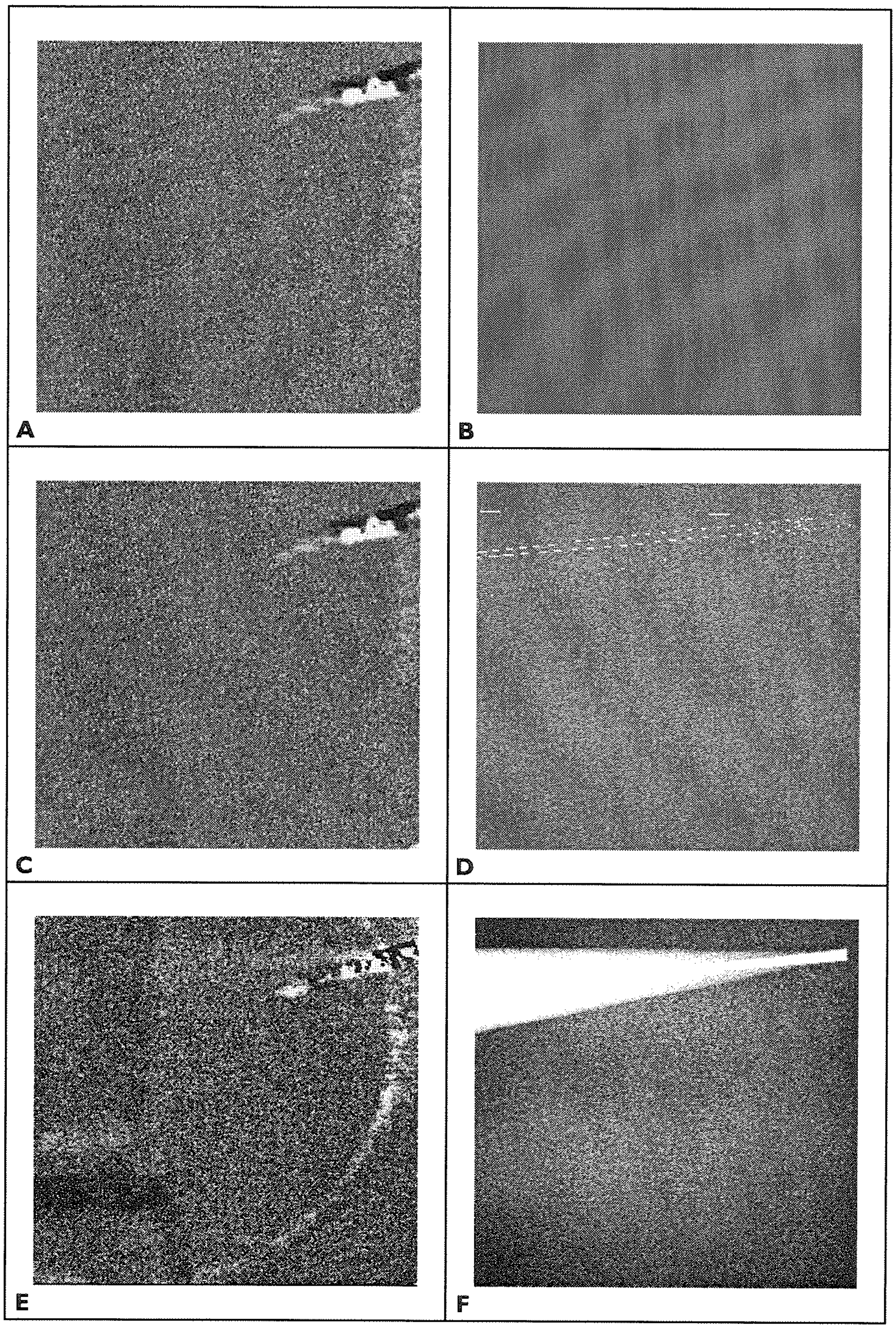


Figure 3.31: Images of a hot plume of 1% CO against a cold background.
(a) Clean filter (N03689-4H, Pixel Range = -100 to 100).
(b) Clean filter (N03689-4H, Voltage Range = 194.4mV to 211.9mV).
(c) CO_2 filter (N04235-4, Pixel Range = -25 to 30).
(d) CO_2 filter (N04235-4, Voltage Range = 416.7mV to 421.9mV).
(e) CO filter (N04693-4, Pixel Range = 5 to 20).
(f) CO filter (N04693-4, Voltage Range = 683.8mV to 687.7mV).

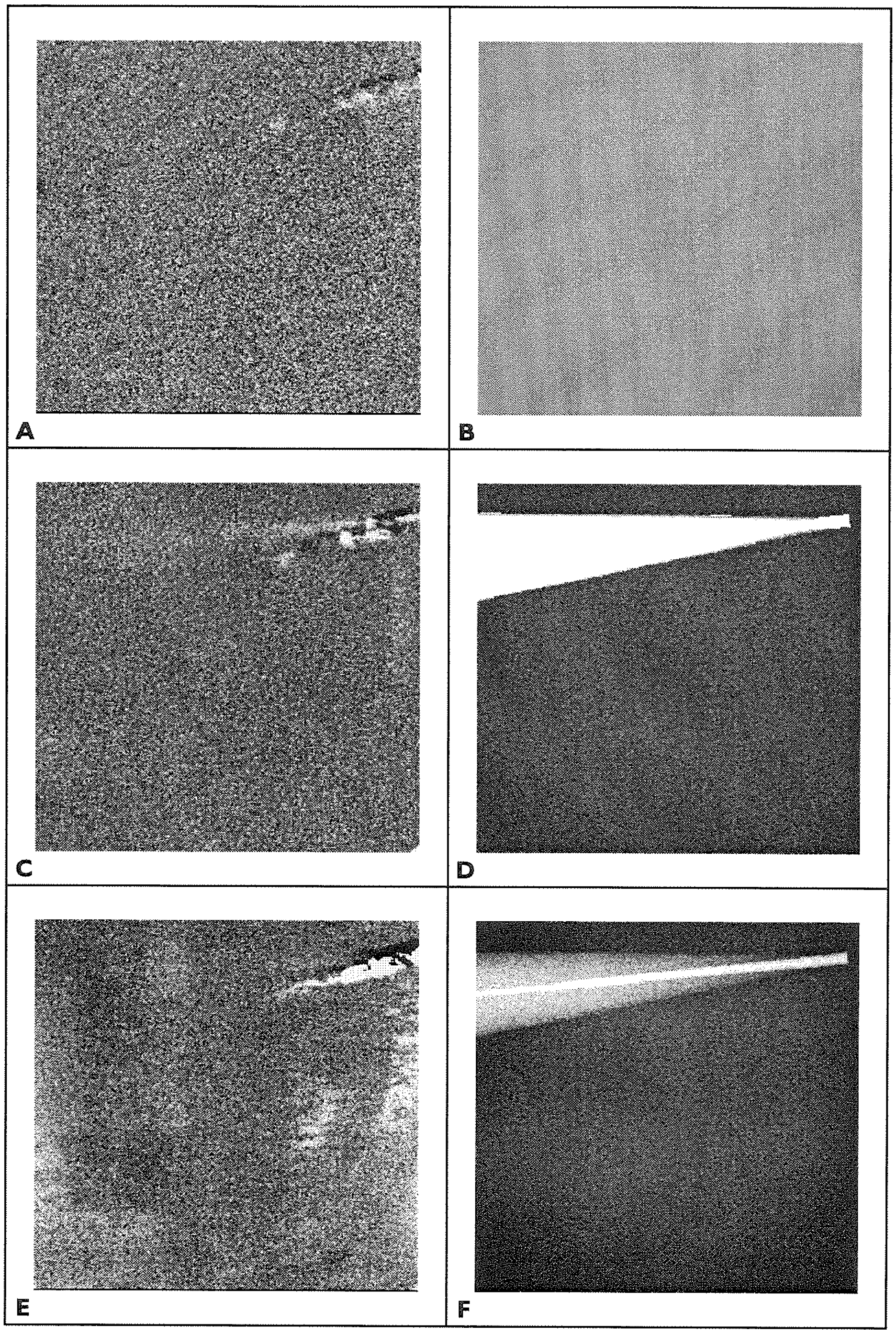


Figure 3.32: Images of a hot plume of 6.67% CO_2/1% CO against a cold background.
(a) Clean filter (N03689-4H, Pixel Range = -100 to 50).
(b) Clean filter (N03689-4H, Voltage Range = 190mV to 203.1mV).
(c) CO_2 filter (N04235-4, Pixel Range = -15 to 25).
(d) CO_2 filter (N04235-4, Voltage Range = 418mV to 421.9mV).
(e) CO filter (N04693-4, Pixel Range = 0 to 20).
(f) CO filter (N04693-4, Voltage Range = 688.9mV to 694.2mV).

mixture of 10% CO_2 and 90% nitrogen. The IRIMAGE static cone model predicts that a bright plume of CO_2 will be visible through the CO_2 filter, but neither of the other two filters. The experimental results agree. The image looking through the CO_2 filter does have a bright plume against the cold background. The other two filters show no trace of the plume. In the case of the plume containing 1% CO and 99% nitrogen, IRIMAGE predicts that the plume should be visible through the CO filter. In addition, IRIMAGE predicts the presence of a very faint plume through the CO_2 filter. This confirms our earlier observation that the CO band is wide enough to be detected by this filter. However, the signal is very faint making it likely that it may be lost in the noise or overwhelmed by the existance of CO_2. IRIMAGE does predict the clean filter to remain clean. Unlike the other case, the plume of CO is visible in the static image and is not visible through the other two filters. Like the results from the hot background/cold plume case, it is still possible to isolate the CO and CO_2 bands using the two filters. In addition, it appears that it is easier to detect a plume of CO under these conditions than in the other case.

As before, we investgated the case of a hot plume containing a mixture of nitrogen, 6.67% CO_2, and 1% CO against a cold background. In figure 3.32, the simulations predict that the CO and CO_2 plumes will both be present and detectable. Similar to the case of the hot background and cold plume, the experimental results agree with what we expect. A plume of CO is visible through the CO filter and a plume of CO_2 is visible through the CO_2 filter while the clean filter remains clean. These results in conjunction with the results from the other cases prove that pollution detection using passive multi-spectral infrared imaging is possible under a variety of conditions. Although possible, these experiments also point out that futher experimentation and simulation is necessary before a system should be built. Another important result of these experiments is the successful use of IRIMAGE as a supplemental tool for predicting the outcome of an experiment.

3.7 Conclusions

These experiments lead to several conclusions regarding passive, multi-spectral infrared imaging. The experiments conclusively verify that IRIMAGE is capable of simulating the infrared imaging process with reasonable accuracy. They also point out certain areas that should be addressed in future versions of IRIMAGE. Although this verification is important, the results related to pollution detection are even more significant. These experiments clearly demonstrate the viability of gaseous pollution detection using passive, multi-spectral infrared imaging. In addition, it is clear that further experimentation and simulation leading to certain advances in technology and system design will also be necessary. Overall, these experiments were quite successful in achieving the goals set forth and should serve as the starting point for further investigation into multi-spectral IR imaging.

It was important to prove that IRIMAGE could generate simulated images that were comparable to images generated by experiment. Although certain discrepancies existed, most of these were related to the calibration procedure and certain approximations made in representing the gas plumes and the surfaces. The calibration procedure relied on a constant temperature across the entire plate surface. Unfortunately, the methods employed to heat the surfaces, could not successfully maintain such a situation. In any case, the effects were minor. Although the approximations were made when the simulation was setup and not inside IRIMAGE, they were neccessary because of certain limitations in IRIMAGE. For instance, we could not properly approximate the plumes using a many cylinders because MODTRAN requires the individual layers of the profile to be greater than a 1mm to avoid artifacts. Adding more cylinders would have caused more artifacts which would have made it more difficult to tell the difference between a correct calculation and an artifact In addition, an acceptable plume model would have required additional code to generate a properly diluted plume that did not incur the same problems with MODTRAN . Until a better atmospheric model is available, this problem will persist.

The noise model was another major problem with the simulated images. For the filters greater than 4 microns, the noise model seemed to work quite well. As the filter bandwidths got smaller and their peak transmissions became less than 4 microns, the noise model began to under-estimate the amount of noise in the image. This is most likely due to the detector noise becoming the dominant noise in these cases. For the cases of short wavelength and small bandwidth, the number of incident photons approaches, and may go below, the minimum number of photons required for background limited performance (BLIP). When this occurs, the detector noise begins to dominate (definition of the BLIP limit). Therefore, IRIMAGE needs to incorporate detector and readout noise for filters with short wavelengths and small bandwidths.

The methane and gas cell experiments conclusively demonstrated the promise of multi-spectral infrared imaging in pollution detection. It was possible to isolate and detect the constituents of several different plumes using a multi-filter arrangement. Although we were able to qualitatively detect the plumes, quantitative determinations of the gas concentrations may be more difficult. This difficulty lies in the fact that the detected change in the incident photons due to a polluted plume relies on the temperature of the background source. For a hot background, the plume reduces the number of photons through absorption. For a cold background, the plume increases the number of photons by radiating them. A solution might be to use the third filter (or more) as a temperature sensor. If we could determine the temperature and emissivity of the background, it might be possible to process the images from the two filters to account for the background source. Further experimentation and simulation should be conducted to determine the best method of solving this problem.

Despite the success of these experiments, certain limitations do exist. The need to recalibrate the camera for each filter made it impossible to conduct these experiments in real time. The recalibration process took between 15 and 30 minutes each time. In order for multi-spectral IR imaging to be useful, this calibration time must be on the order of a few milliseconds or eliminated entirely. Using current technology, this might be possible using filters that have central peaks and bandwidths that do not require recalibration after a new filter is rotated into view. In this case, the average number of photons collected by each filter must be of the same order. If each filter had similar transmission curves, the bandwidths of the shorter wavelength filters would need to be larger than the longer wavelength filters. Furthermore, the temperature range of the measurement would greatly influence the filter selection. Although a system like this would be reasonable in an academic or research setting, the expense and maintenance of such a system makes it impractical as a commercial system. Therefore, a new type of imaging system is needed such as an imager with a micro filter on each detector; a stack of detectors, each with a different cutoff; or even multiple FPA systems with a different filter per FPA. The challenge is to determine the best systems for passive, multi-spectral infrared imaging and implement it, and IRIMAGE provides the basic tools to make this determination.

Bibliography

[1] *Amber-View Software Manual, Release 1.0.a* (Amber Engineering, Goleta, CA, 1992).

[2] *Implementation of the Flexible Image Transport System (FITS)* NOST 100-0.3b, (Nasa/OSSA Office of Standards and Technology, Greenbelt, MD, 1991).

[3] AVS is a trademark of Advanced Visual Systems Inc.
More information about AVS can be obtained from: Advanced Visual Systems Inc., 300 Fifth Ave., Waltham, MA 02154.

[4] For an excellent discussion of terrestrial monitoring using imaging spectroscopy: G. Vane and A.F.H. Goetz, "Terrestrial Imaging Spectroscopy," *Remote Sensing of Environment*, 24, 1988, 1-29.

[5] G. Vane, R.O. Green, T.G. Chrien, H.T. Enmark, E.G. Hansen, and W.M. Porter, "The Airbourne Visible/Infrared Imaging Spectrometer (AVIRIS)," *Remote Sensing of Environment*, 44, 1993, 127-143.

[6] A.F.H. Goetz and Mark Herring, "The High Resolution Spectrometer (HIRIS) for Eos," *IEEE Transactions on Geoscience and Remote Sensing*, 27(2), March 1989, 136-144.

[7] An excellent review of vibrational/rotational modes is in: R.M. Eisberg and R. Resnick, *Quantum physics of atoms, molecules, solids, nulcei, and particles*, (John Wiley $ Sons, New York, 1974).

[8] Curves based on data from: C.J. Pouchert, *The Aldrich library of infrared spectra, third ed.* (Aldrich Chemical Co, Milwaukee, WS, 1981).

[9] J.H. Seinfeld, *Atmospheric Chemistry and Physics of Air Pollution* (John Wiley & Sons, New York, 1986), p. 93.

[10] J.H. Seinfeld, *Atmospheric Chemistry and Physics of Air Pollution* (John Wiley & Sons, New York, 1986), p. 101.

[11] Linde Gases of Southern California, *Specialty Gases & Equipment Catalog*, Volume 25, p. 106.

Appendix A Block Diagrams of IRIMAGE

Part A of this appendix contains a series of block diagrams of IRIMAGE. These block diagrams describe how the major programming objects are organized. They are designed to give the reader a graphical view of how the various pieces of IRIMAGE are put together to form the major elements of the simulation. The following is a listing of the five diagrams:

- Diagram A.1 shows how the six major elements of IRIMAGE are connected together.
- Diagram A.2 shows how the Image Database, the VIO Database, the Layout Database and the Imaging VGO are contained within the Scene Object.
- Diagram A.3 shows how the MODTRAN object, the Geometric Model Database and the Atmosphere VGO are contained within the Atmosphere Object.
- Diagram A.4 shows how the four detector databases are contained within the Detector Object.
- Diagram A.5 shows how the Optics Object, the Output Object, Element Database and Unit Cell Database are connected through the FPA Object.

These diagrams designate the major objects and databases as rectangles; databases that are part of larger databases or part of elements of a database (like key frame databases) as diamonds; and non-database elements of a major database as well as objects that store important data as circles. These diagrams do not attempt to demonstrate how IRIMAGE works, only how it is put together.

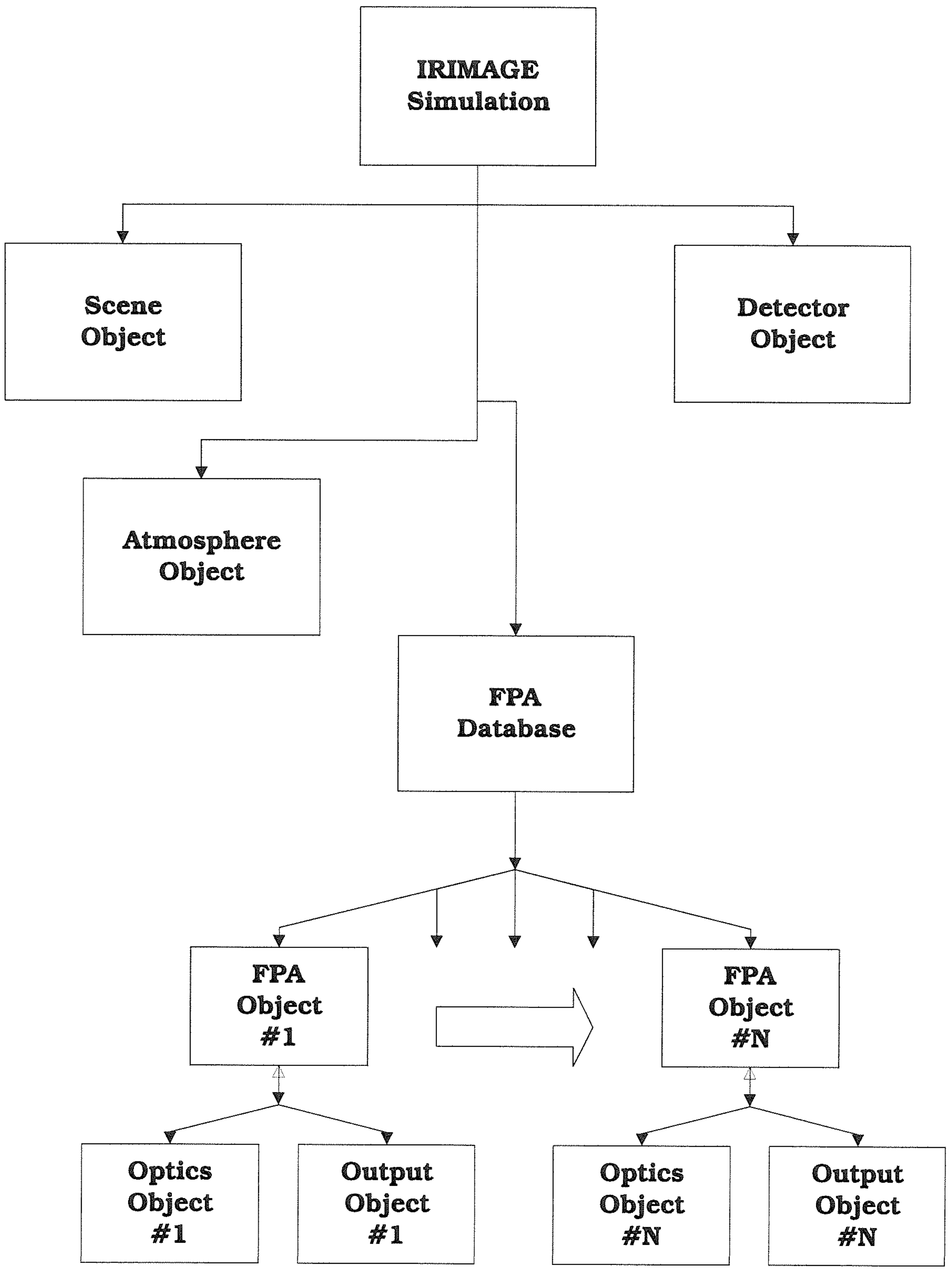


Figure A.1: Diagram showing how the six major objects are connected in IRIMAGE.

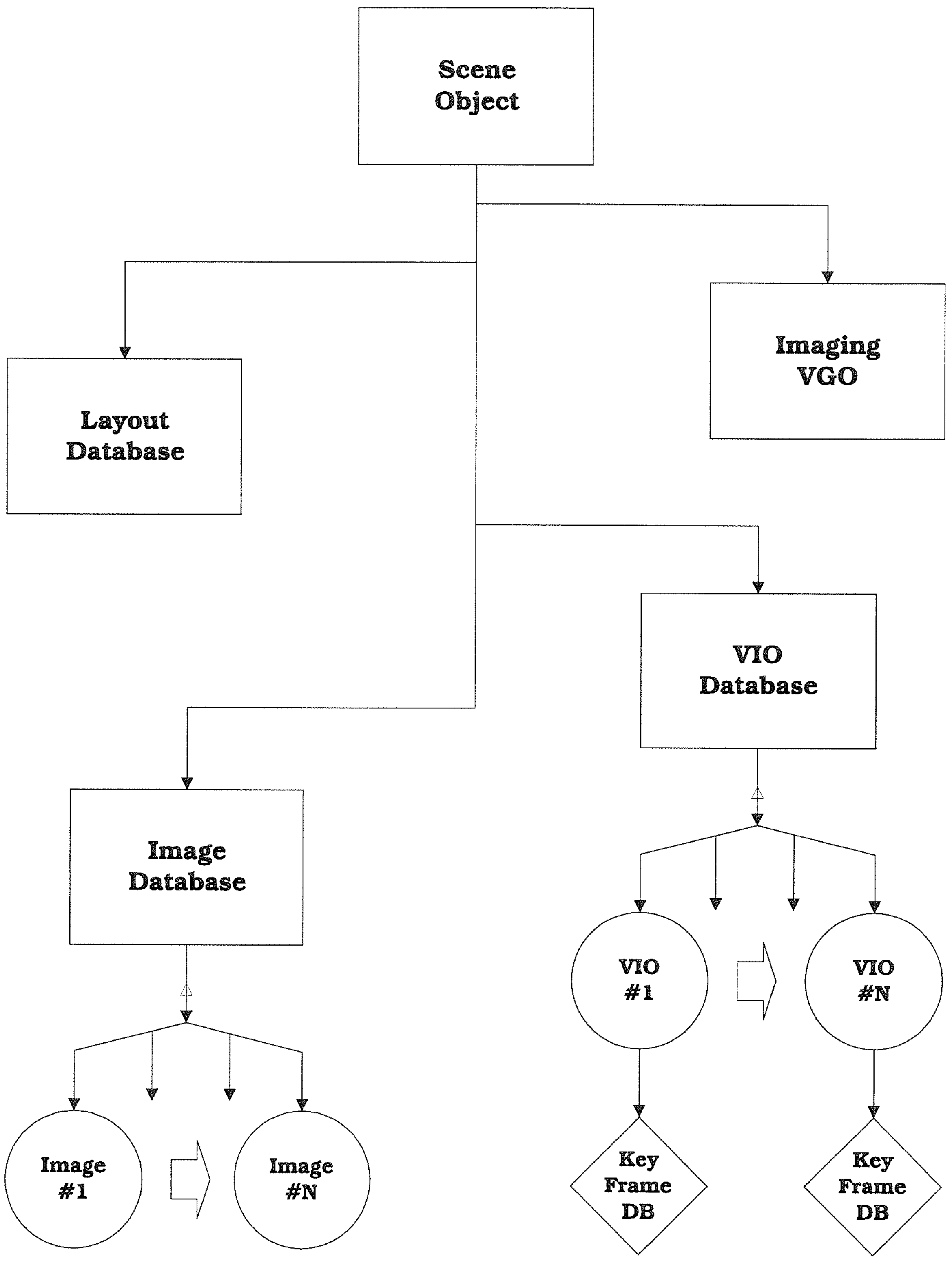


Figure A.2: Diagram showing the supplementary objects in the Scene Object.

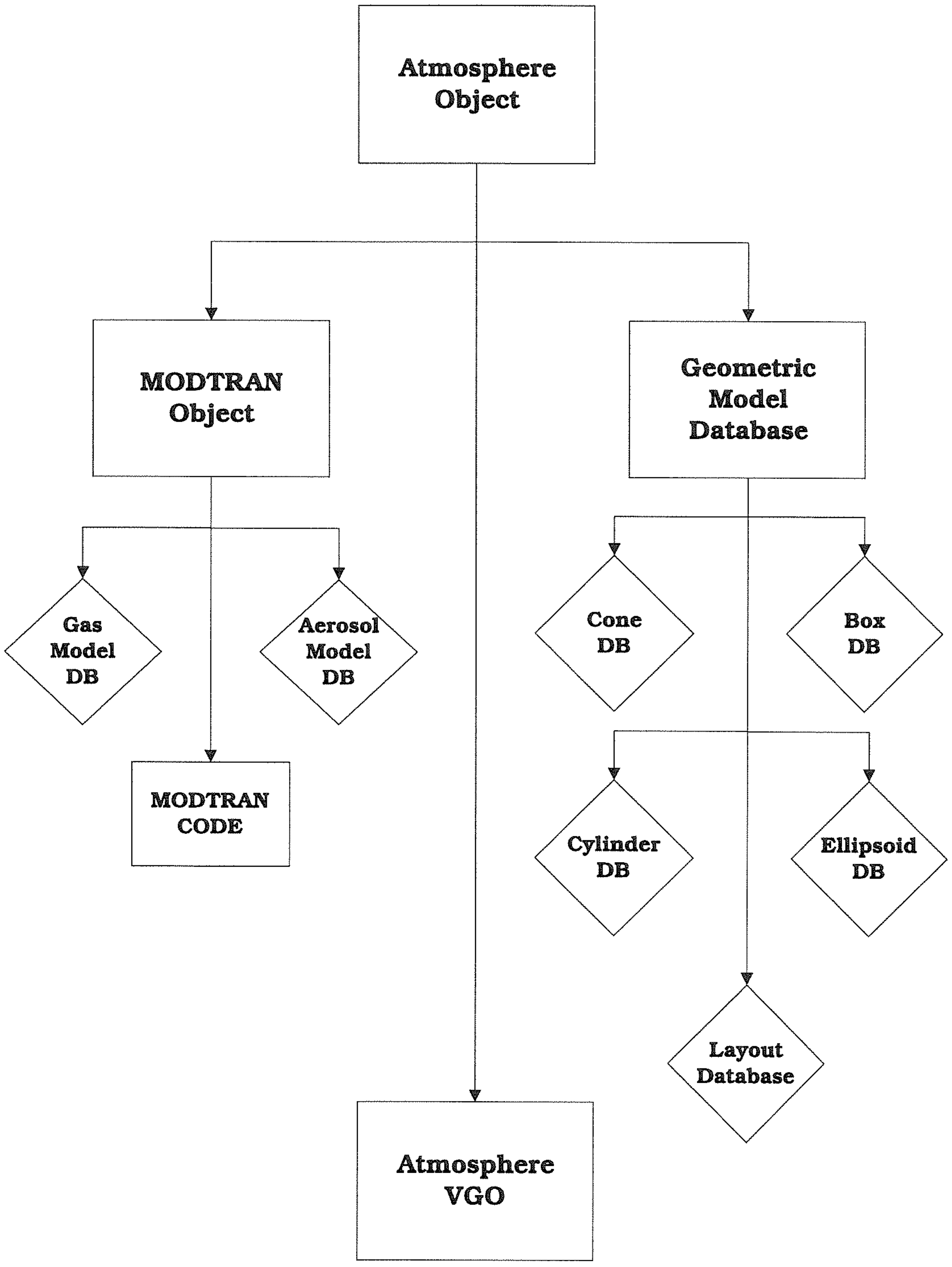


Figure A.3: Diagram showing the supplementary objects in the Atmosphere Object.

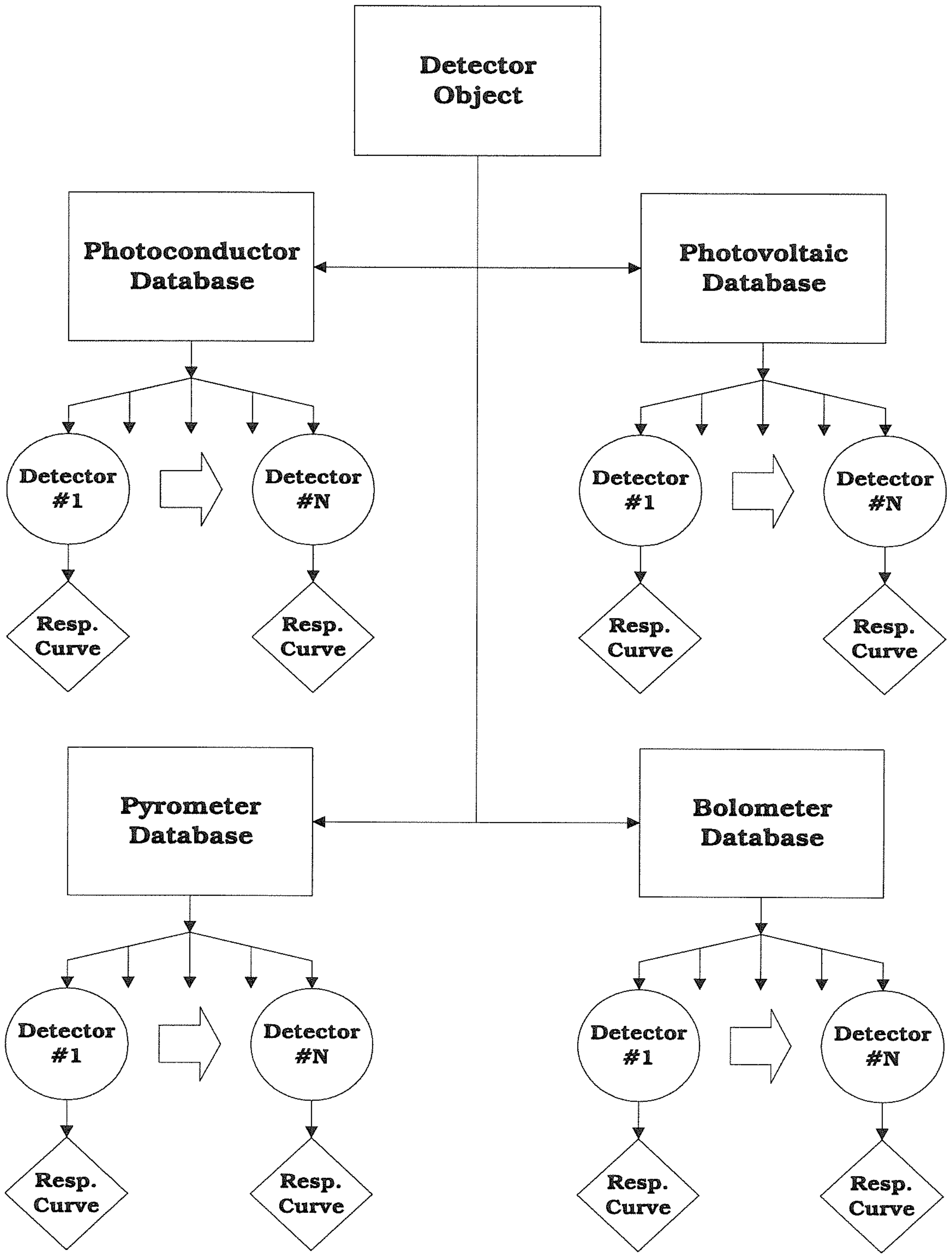


Figure A.4: Diagram showing the supplementary objects in the Detector Object.

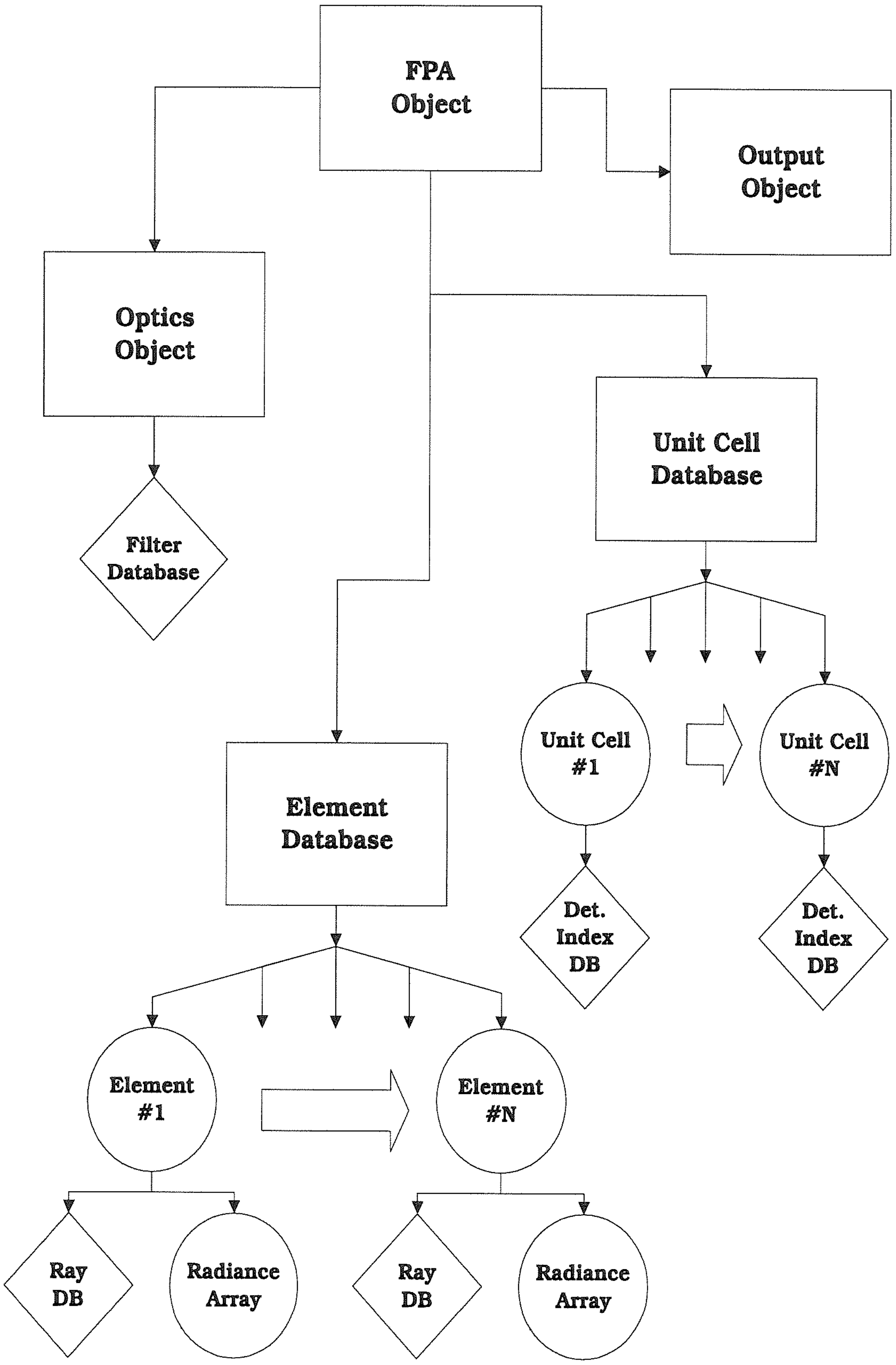


Figure A.5: Diagram showing the supplementary objects in the FPA Object.

Appendix B Interpolating Volumetric Data onto a 3D Orthogonal Grid

B.1 Introduction

There are several approaches to defining a three-dimensional scene in a computer environment. The most straightforward approach is to define geometric models as bounding surfaces and assign each model a certain location and orientation [1]. This method can produce amazingly photo-realistic images and has become the most popular form for computer graphics. Another method creates primitive volumetric models (spheres, boxes, etc.) and places them in the 3-D environment to form more complex models. Modeling techniques like Constructive Solid Geometry [2] and voxel-based octrees [3] utilize this method. Volumetric representations may not produce photo-realistic images, but they are quite useful for imaging applications which require faster image production or contain lots of volumetric data. The third technique employs some form of mesh generation algorithm, such as the Advancing Front method [4, 5], to represent the surface of an object by filling the space surrounding the object with a 3D mesh of points. We call this technique *Negative Space Filling (NSF)* because the interior of objects is not filled with the mesh, only the empty space around the objects. This technique is useful when solving three-dimensional differential equations numerically or conducting a finite element analysis (e.g., aerospace applications, mechanical engineering, etc.).

A scene composed of volumetric representations may not give a completely accurate surface representation, but it does contain important volumetric information about the models in the scene. The most popular form of this type of data representation utilizes the concept of *voxels*. Voxels are box structures which act like building blocks. When they are stacked in a particular order, they form a volumetric representation of a geometric model. The simplest form of a voxel representation is called *Space Occupancy Enumeration (SOE)* [6]. An SOE representation divides the space surrounding and containing a geometric model into a regular set of 1-bit boxes. Each box is either inside the object ($bit = 1$) or outside ($bit = 0$).

Our approach builds on the basic SOE representation. In most cases, only the voxels that make up the geometric model are stored; the rest are discarded. Such a procedure only represents individual geometric models and not entire scenes. We have chosen to modify this SOE-Voxel approach by using a concept from the NSF method. In our case, we divide our entire working space into a regular lattice of identical *grid boxes*, i.e., voxels, to form an orthogonal, three-dimensional grid. We define, store and manipulate this grid and the individual voxel information through the use of the *Versatile Grid Object (VGO)* which is discussed in section B.2. Unlike the NSF method, we do not remove points and/or adjust the lattice to define bounding surface. Instead, each voxel stores a specific set of values that associate it with a particular geometric model or the space between models. As one can see, this extends the SOE representation beyond the simple 1-bit methodology.

In order to effectively use the VGO to represent an entire scene, we developed an object-oriented construct which facilitates the placement of volumetric data into the individual voxels of the VGO. In section B.3, we introduce the concept of the *Universal Primitive Object (UPO)*. Each UPO represents a specific set of volumetric data using a geometric model that defines the bounding surface of the data and an accompanying data index which links a set of data to the volume enclosed by the surface. The geometric model employs a *polygonal-based* surface representation which approximates a three-dimensional surface with a patchwork of two-dimensional polygons. The UPO is flexible enough to build virtually any primitive object while it is also well suited for our "Slice and Dice" interpolation algorithm.

In section B.4, we discuss the "Slice and Dice" algorithm for interpolating UPO-based models onto a VGO. This algorithm employs a set of cutting planes that slice each UPO along the Z-axis of the VGO. The result is a series of curves on the cutting plane that represent the intersection of the plane with the surface of the UPO. These curves are then interpolated with a two-dimensional grid using a published interpolation algorithm. This 2D grid is a projection of the 3D VGO on the cutting plane and represents the voxels that intercept the cutting plane. Finally, our algorithm associates a voxel with the volumetric data of the UPO if its associated 2D grid rectangle has more than 50% or its area inside the intercepted curve. In this way, bounded surfaces are interpolated onto the VGO.

In section B.5, this chapter introduces how the VGO, UPO and the interpolation algoithms are used in the application IRIMAGE. IRIMAGE uses the concepts developed in this paper to model the conditions of the atmosphere in an infrared scene. This application takes advantage of the various aspects of these program objects and the interpolation algorithm that links them together.

B.2 The Versatile Grid Object (VGO)

The Versatile Grid Object is a self-contained program object that stores either integer or floating point data in a 1, 2, or 3 dimensional, regularly spaced, orthogonal lattice construct, i.e., grid. The grid contains its own coordinate system and transformation matrices which link any point in the *Grid Coordinate System (GCS)* to a point in *World Coordinate System (WCS)*. Grid point data entry and retrieval can be conducted using either a set of three integer indexes (X, Y, Z) or real coordinates (x, y, z) in either the GCS or WCS. In addition, support exists for using 3-Vector representation of these coordinates or indexes $(\vec{r} = x\hat{\mathbf{x}} + y\hat{\mathbf{y}} + z\hat{\mathbf{z}})$. Furthermore, support exists to assign a specific "object index" to grid points that contain data from a particular object (image, UPO, etc.). These properties make it possible to store, manipulate and retreive data associated with sources like images and three-dimensional bounded volumes.

B.2.1 The Grid Coordinate System (GCS)

The VGO has its own internal coordinate system called the Grid Coordinate System (GCS). The origin of the GCS lies at the center of the VGO's grid, defining a

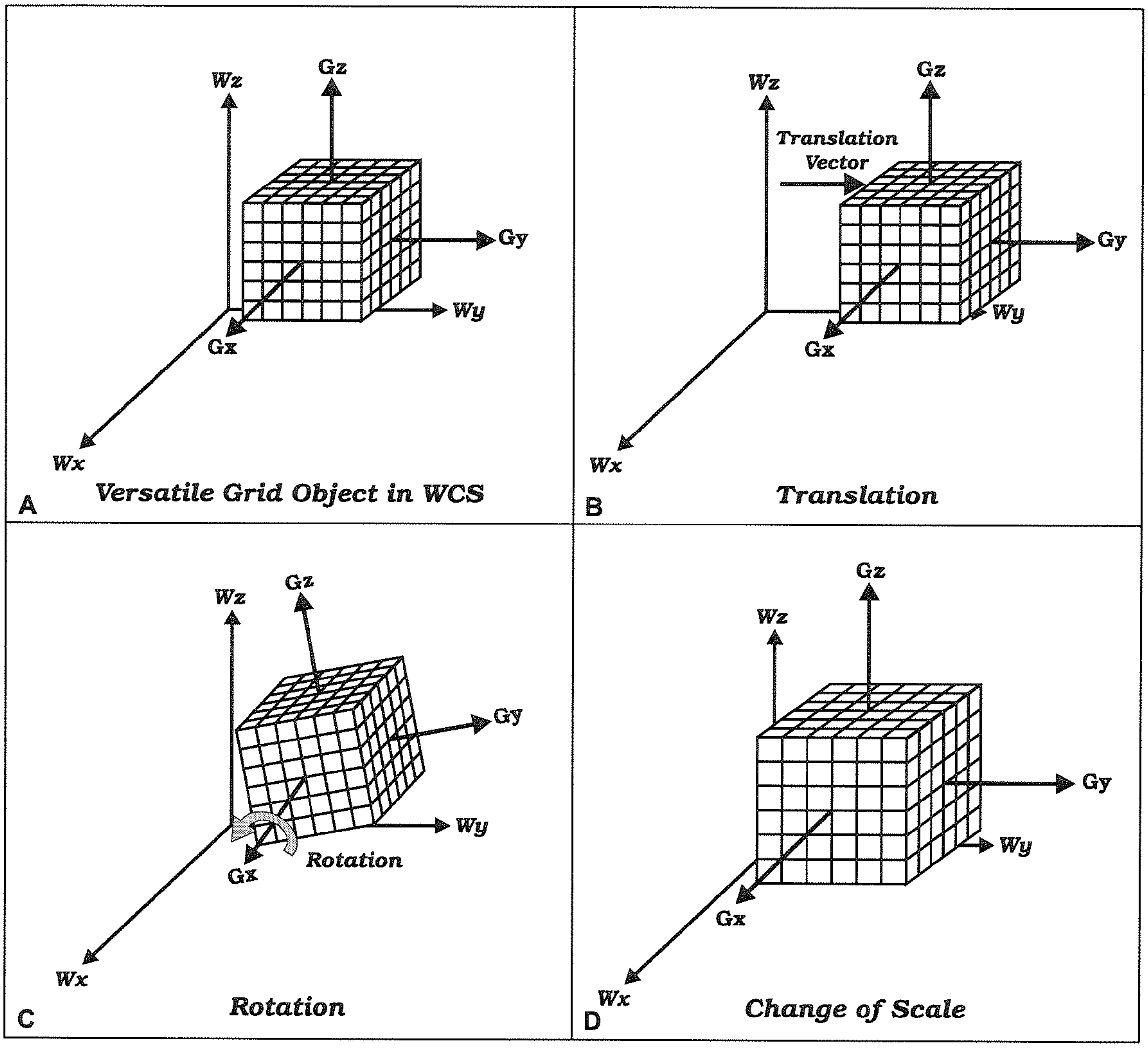


Figure B.1: Various VGO transformations: (a) Original GCS to WCS relationship. (b) Translation in Wy direction. (c) Rotation about Gx axis. (d) Change of Scale adjusts grid point spacing.

symmetric octant (quadrant in 2-D) structure. Since most of the applications that will use the VGO have their own coordinate system, we define the overall coordinate system of an application to be the World Coordinate System (WCS). Given the location and orientation of the VGO's grid in the WCS, the GCS constructs a set of transformation matrices that link the GCS to the WCS. These matrices map a point in the WCS to a point in the GCS and vice versa by translating and/or rotating the point. The only requirement for the GCS, with regards to the WCS, is that the GCS and WCS are based on the same units. In other words, the distance between any two points in the WCS is the same after they are transformed to the GCS. The spacing between each grid point is defined with respect to the GCS so that each grid point's location is defined in the GCS. The physical size of the VGO may be scaled, but this does not affect the transformation matrices because it only scales the spacing between grid points and not the GCS itself. Figure B.1 shows the relationship between the GCS and WCS and how various transformations effect this relationship. Note in this figure that change of scale only makes each grid box larger, but does not affect the coordinate system. This relationship between the GCS and the WCS allows all operations internal to the VGO to be completely disconnected from the WCS, thereby simplifying many of these VGO operations.

The following equations define the basis matrices for the corresponding transformation matrices. These matrices correspond to the standard form for three-dimensional computer graphics [7].

The Translation Matrix ((T($\vec{d}$)) and Scale Matrix (S($\vec{s}$))

$$\mathbf{T}(\vec{d}) = \begin{bmatrix} 0 & 0 & 0 & d_x \\ 0 & 0 & 0 & d_y \\ 0 & 0 & 0 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{S}(\vec{s}) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The Rotation Matrices $(\mathbf{R_x}(\theta), \mathbf{R_y}(\omega), \mathbf{R_z}(\phi))$

$$\mathbf{R_x}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{R_y}(\omega) = \begin{bmatrix} cos\omega & 0 & sin\omega & 0 \\ 0 & 1 & 0 & 0 \\ -sin\omega & 0 & cos\omega & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R_z}(\phi) = \begin{bmatrix} cos\phi & -sin\phi & 0 & 0 \\ sin\phi & cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Using simple matrix multiplication, we can generate the transformation and inverse transformation matrices. The order of this multiplication is extremely important and care must be taken to define the angles properly. With this in mind, equations (A.2.1) and (A.2.2) define this order.

$$\mathbf{M_{WCS \to GCS}} = \mathbf{R_z} \cdot \mathbf{R_y} \cdot \mathbf{R_x} \cdot \mathbf{T} \tag{A.2.1}$$

$$\mathbf{M_{GCS \to WCS}} = \mathbf{T^{-1}} \cdot \mathbf{R_x^{-1}} \cdot \mathbf{R_y^{-1}} \cdot \mathbf{R_z^{-1}} \tag{A.2.2}$$

Changing the grid point spacing vector $(\vec{G})$ conforms to a simple matrix-vector multiplication of the scaling matrix, $\mathbf{S}$, and $\vec{G}$:

$$\vec{G'} = \mathbf{S} \cdot \vec{G} \tag{A.2.3}$$

Equation (A.2.3) transforms the grid point spacing for all three axes of the VGO. As mentioned, a change in scale only effects translating the GCS coordinates of the grid point into the integer indexes required for retrieving data from the grid. In addition, the scaling matrix is always stored and acts on the spacing vector whenever the spacing vector is called. Thus, the original spacing remains unchanged so that every time a change of scale occurs, it is an absolute change of scale and not relative to the previous scaling.

B.2.2 Data Storage in the VGO

The internal structure of the VGO is made of a one-dimensional, integer-based *Object Index Array (OIA)* and a one-dimensional *Flexible Data Array (FDA)* that stores either integer or floating point data. The OIA is an array of indexes that link particular data set objects (images, UPO's, etc.) with the grid points that are affected by the object. For instance, if a geometric model like a box or a sphere is interpolated onto the VGO, each grid point residing inside the model would have its OIA index set to the unique integer that represents the model. This makes it possible to quickly determine which points are associated with a particular object when changing the VGO or removing that object from the VGO. If this link is unnecessary, the OIA may store a link to an external data structure which eliminates the need for the other data array.

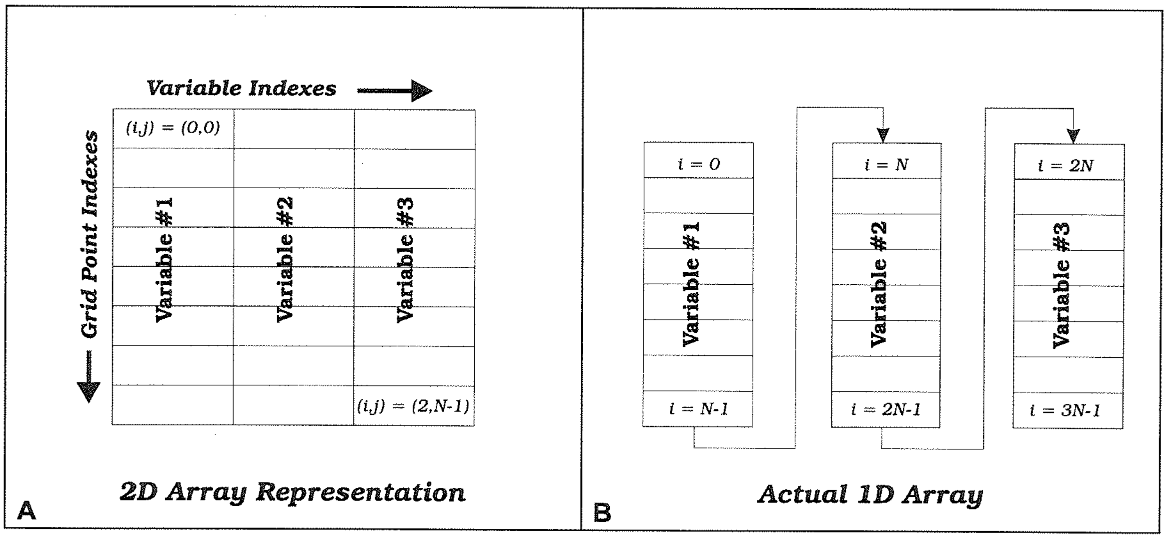


Figure B.2: Example of how data array is stored in the VGO: (a) Perception of data stored in a 2D data aray. (b) Actual 1D array structure.

The FDA can store multiple variable values in each grid point of the VGO. For a VGO with N grid points and M data values stored at each point, the FDA stores $N \cdot M$ values. Figure B.2 demonstrates how the data is stored as a one-dimensional array although its access may appear more two-dimensional, i.e., the coordinates and a data index. The one-dimensional nature makes it possible to dynamically allocate the memory at run-time instead of fixing one or more of the dimension sizes as required for a multi-dimensional array. The variable order and number of variables may be defined at anytime including during run time. There are several easy to use routines for loading and retrieving both the OIA index and the various FDA values. This protects the integrity of the VGO, while allowing the programmer to manipulate the VGO's contents. The VGO also maintains a *default array* to store a set of default values for each grid point. This array is useful whenever the program needs to return a grid point to a default setting.

Although the OIA and FDA are one-dimensional in nature, their association with a specific grid point location is defined by three integer indexes corresponding to the three primary axes of the GCS. In addition, a fourth index determines which one of variables stored at each grid point to access. The integer indexes are always positive where the $(0, 0, 0)$ position represents the minimum physical coordinate of the VGO and the $(X_{max}, Y_{max}, Z_{max})$ respresents the maximum physical coordinate. Since the VGO's grid is orthogonal and the grid points are regularly spaced, any lattice point can be defined by a simple integer triplet (X, Y, Z) which does not need to be stored. Instead, the regular nature of the VGO allows it to use equations (A.2.4) and (A.2.5) to convert a multi-dimensional integer index into a single unique index for accessing the one-dimensional arrays.

Object Index Array (OIA) & Flexible Data Array (FDA) Indexes

$$OIA_{ind}(X, Y, Z) = X + (N_X)Y + (N_X N_Y)Z \tag{A.2.4}$$

$$FDA_{ind}(X, Y, Z, V) = OIA_{ind}(X, Y, Z) + (N_X N_Y N_Z)V \tag{A.2.5}$$

where

N_X, N_Y, N_Z = Total # of grid points along each axis

X, Y, Z = Integer index values for the current grid point

V = Index that indicates which element of the FDA should be set or retrieved

The VGO may use the integer index of a grid point to calculate its physical coordinate, (x, y, z), in the GCS. Using the quantities defined above with the grid point spacing vector, $\vec{G}$, the VGO can compute (x, y, z):

$$x = (X - \frac{N_X}{2})G_x, \qquad y = (Y - \frac{N_Y}{2})G_y, \qquad z = (Z - \frac{N_Z}{2})G_z \tag{A.2.6}$$

Similarly, the nearest integer index can be found by reversing (A.2.5) and rounding to the nearest integer value.

Using the WCS to GCS transformation matrices, it is even possible to retrieve the data for a grid point by using its WCS coordinate. By transforming the coordinate to GCS, it can then use the above schemes to retrieve the nearest point in the grid and its associated values. In addition, the accessing scheme may be used for 1D and 2D cases by setting the number of points along one or two of the axes to a value of one, e.g., $N_Z = 1$ for a 2D grid. The accessing scheme then adjusts instantly.

The VGO does not contain routines which interpolate any type of image, UPO or other objects onto it. Instead, each of these objects that define 2-D images and 3-D bounding surfaces must have their own interpolation algorithms. By doing so, the VGO is separated from any of the objects that might be interpolated on it. This avoids the VGO being constantly adjusted for new geometric objects and interpolation algorithms.

B.3 Defining Bounded Volumes for Interpolating on the VGO

With a VGO in place, it is possible to discuss how one goes about defining a bounded volume to be interpolated on the VGO. The two major components of a bounding volume are the bounding surface which defines the physical extent of the volume and the parameters that make the volume different from its surroundings. As discussed, various methods exist for defining the bounding surface of an object. Polygon-based surface modeling is accepted to be the most flexible and straightforward methodology for generating geometric surfaces. With this in mind, we introduce a hybrid form of polygon surface modeling that improves the interpolating of surface models on the VGO. In addition to the surface modeling technique, it is also necessary to incorporate the volumetric data into our approach to modeling bounded volumes. To accomplish this, each bounded volume is represented by a three-dimensional, "smart" geometric model called a *Universal Primitive Object (UPO).* We use the term "universal" because it is capable of building any type of simple primitive surface model (sphere, box, cone, etc.) as well as many other more complex objects. Each UPO not only contains the algorithms and variables ncecessary to build the surface model, but it also stores the volumetric data to be interpolated on the VGO.

B.3.1 Defining Bounded Surfaces Using Polygonal Surface Modeling

In the process of defining the algorithm for intercepting a plane with a bounding volume, we surveyed the methods for creating geometric models in computer graphics (CG). The most prevalent method for defining models is to build the bounding surface from a set of two-dimensional polygons wrapped around the surface. This produces a piece-wise approximation of the surface. Furthermore, all other surface representations can be approximated using this method. Since this method is the most straightforward and popular method, we chose to use it as our basis for interpolating three-dimensional volumes onto the VGO.

In standard CG, polygon surface modeling can be defined in two ways. The *polygon-based* method treats and stores each polygon of the bounding surface as a separate geometric object. Each polygon stores its own point coordinates and is independent of all other polygons. As can be seen in figure 3-1a, every point that is shared by more than one polygon is stored multiple times. Although this is the quickest method to generate a surface, it is highly inefficient.

In the *edge-based* method, each polygon stores an ordered list of edges which define its circumference [8]. In addition, each edge end point on the surface is kept in a separate *point array.* By doing so, each edge just stores the point array indexes of the two end points in the edge array. Since the edges are also stored in an array, each polygon just stores a list of array indexes as well. Figure B.3b demonstrates how the polygons are inherently linked to each other through their common edges. This method does require a few more steps to generate the model, but there is a significant improvement in memory use.

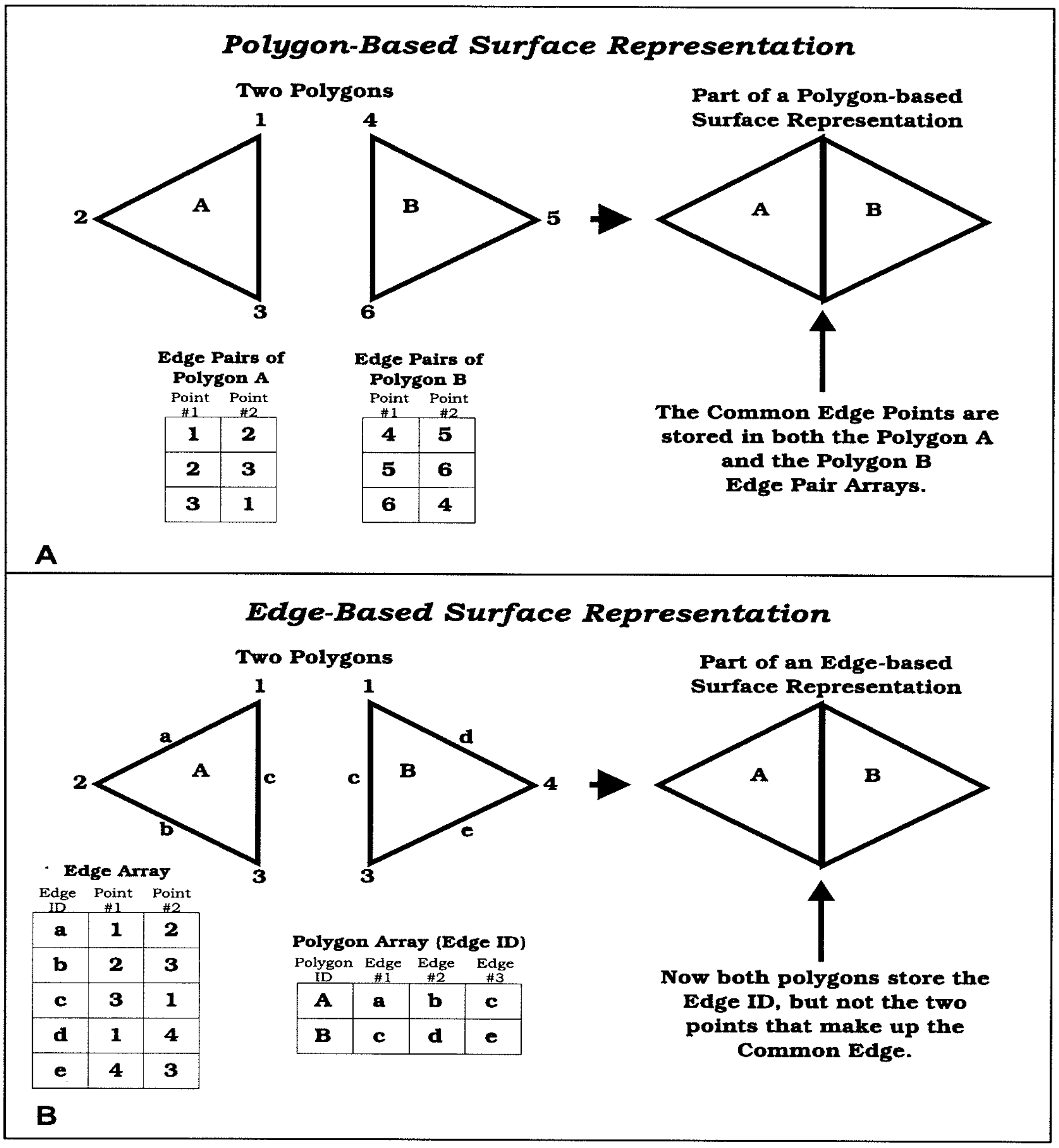
In a polygon-based surface model, each polygon is independent so that any surface-plane intersection may require cycling through the same edge twice. However, the edge-based method will only need to intercept any edge once, making it more efficient in both memory and speed. In our algorithm, we utilize a hybrid approach which is basically the edge-based method with a few minor additions. This algorithm utilizes three distinct data structures:

The Point Array Structure

This structure simply stores the three-dimensional coordinates for every vertex point on the surface using a 3-Vector data type. The array structure is just an unordered one-dimensional array of 3-Vectors. The array can be unordered since the points will be linked to edges via their array index.

The Edge Array Structure

Like the edge-based approach, this array stores the pointers (integer array indexes) to the end points of each edge line. Since the model will be intercepted by a cutting plane, we also incorporate two more integer pointers which store the two polygons


Polygon-Based Surface Representation
Two Polygons
1
4
2
A
B
5
3
6
Part of a Polygon-based Surface Representation
A
B
The Common Edge Points are stored in both the Polygon A and the Polygon B Edge Pair Arrays.
A
Edge-Based Surface Representation
Two Polygons
1
1
a
d
2
A
c
c
B
4
b
e
3
3
Part of an Edge-based Surface Representation
A
B
Now both polygons store the Edge ID, but not the two points that make up the Common Edge.
B

Edge Pairs of Polygon A

Point #1	Point #2
1	2
2	3
3	1

Edge Pairs of Polygon B

Point #1	Point #2
4	5
5	6
6	4

Edge Array

Edge ID	Point #1	Point #2
a	1	2
b	2	3
c	3	1
d	1	4
e	4	3

Polygon Array (Edge ID)

Polygon ID	Edge #1	Edge #2	Edge #3
A	a	b	c
B	c	d	e

Figure B.3: Two methods of defining a bounded surface. (a) In a polygon-based system each edge that is common to two polygons is stored twice (once by each polygon). (b) In an edge-based system, each edge that is common to two polygons is stored only once since polygons only store pointers to edges.

that share the common edge. The usefulness of this additional information will be discussed later.

The Polygon Array Structure

The polygon array structure stores the pointers (integer array indexes) to the edges that make up the polygon. We use triangular polygons which automatically maintains an ordered list of edges. In any polygon surface model, it is best to use triangular polygons when possible. Although less efficient memory wise, using triangular polygons ensures that each polygon is always planar (required for correct modeling) since 3 points can only define a plane while more can describe a non-planar surface.

B.3.2 The Universal Primitive Object (UPO) Model

In the process of developing the VGO-Bounded Surface interpolation algorithm, the idea of a *Universal Primitive Object* was developed. The UPO has the same structure and generation properties of the classic primitive shapes (ellipse, cone, cylinder and box). In addition, the UPO's structure allows defining and modeling of extruded shapes and surfaces of rotation. Although it uses the edge-based concepts, the UPO can easily be extended to the classic polygon-based method as well. In either case, each surface is built using triangular polygons. While the UPO may be slightly memory inefficient, it is these characteristics that allow the universal idea as well as improve the speed of the building and interpolation process.

The UPO model utilizes two basic constructs, *multiple control contours* and *two end points.* The series of control contours define the external point structure of the UPO. The contours can define most of the surface, but they can't close the surface at the two ends of the surface. Instead, the UPO caps off the surface by connecting the first end point to the first control contour and the last end point to the last control contour. Both the control contours and the end points must be defined in the UPO's own coordinate system which is called the *Primitive Object Coordinate System (POCS).* This coordinate system does not need to coincide with the WCS, but like the GCS it must maintain the same base unit (e.g., feet, meters, etc.) so that points in the POCS can be transformed to either the WCS or the GCS.

Control Contours

These contours circumscribe the outer surface of the object. Every contour has the same number of points. Each contour is defined using either a pre-defined set of points or is built using a center point, radius value and orientation vector. Most primitive shapes may be defined by contours which are built using a center point, radius and orientation vector. However, more complex shapes may require the contours to be pre-defined. The only restriction is that only one of these methods may be used to define a particular UPO.

Once the individual contours are set up, they can be connected together or to one or both of the end points. If the contours are built, the points are regularly spaced around the contour by the angle θ. In addition, every other contour is rotated by $\theta/2$ with respect to its adjacent contours. This provides a smoother surface as well as a more generalized method of connecting multiple rings.

Adjacent contours can be connected in two ways. The first method simply connects the individual points of each contour to form the edges of a closed contour. Then the adjacent contours (or end points) are connected to the current contour to form the edges of a set of triangles that wrap around the surface. Figure B.4 demonstrates how an ellipsoidal object is created using this method. The concept of creating edge connections between points on each contour and adjacent contours is described as follows:

Method 1 *Given that each contour has N points. For contour R_n, each Point P_j is connected to the point P_{j+1}. In addition, P_j of R_n is also connected to points P_j and P_{j+1} on contour R_{n+1}. For the case of $P_{j=N+1}$, use point P_1.*

The problem with the first method is that it is unable to create parallelepiped models because every contour is out of phase with its adjacent contours by $theta/2$. This phase difference makes smoother looking curved surfaces, but can't make a simple box. However, eliminating the phase difference also has problems because if a point is displaced, the effect on the surface depends greatly on how it is connected

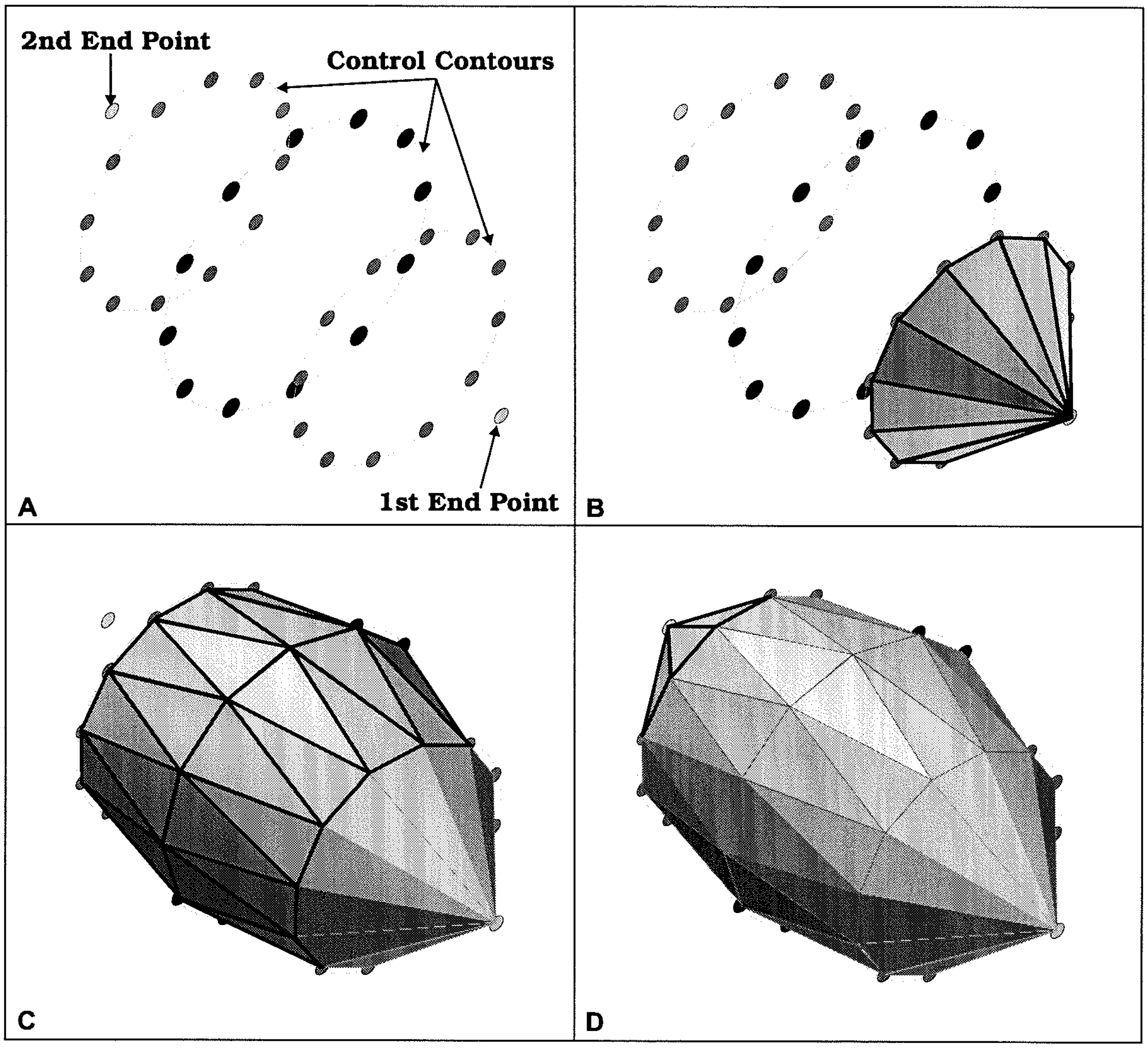


Figure B.4: Example of normal connection method for creating an Ellipsoid: (a) The control contours and end points of the Ellipsoid. (b) The first contour's edges are connected and the first end point is connected to the points on the first contour to form the top cap of the Ellipsoid. (c) The rest of the contour edges are connected. Adjacent contours are connected using the method of connecting point P_i on one contour to the points P_{i-1} and P_{i+1} on the adjacent contours. (d) The last contour and the second end point are connected to form the bottom cap.

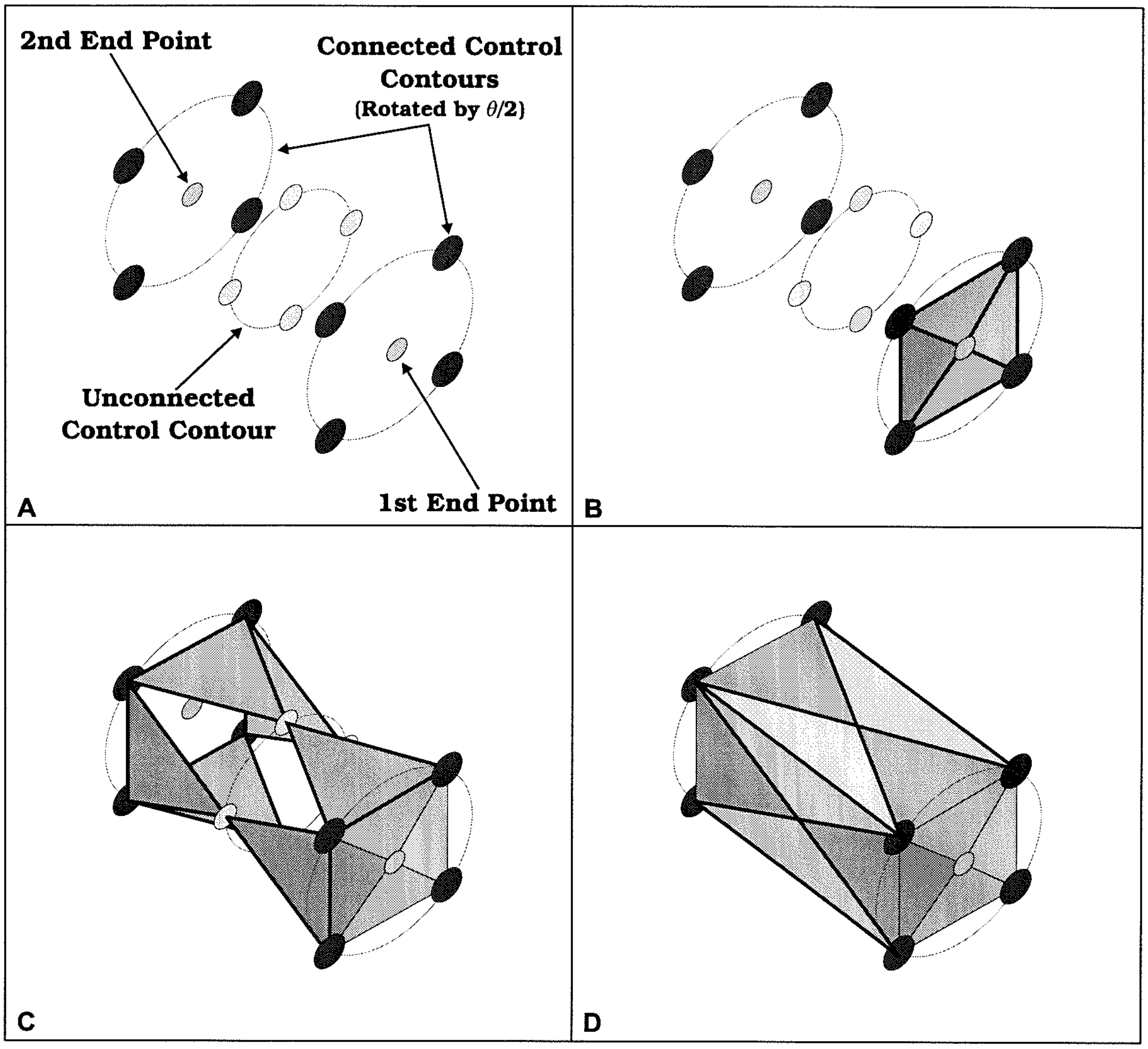


Figure B.5: Example of alternating method of connecting contours for creating a Box: (a) The control contours and end points of the Box. Note middle contour is smaller than the outer two contours and the end points lie in the same planes as the first and last rings. (b) The first contour's edges are connected and the first end point is connected to the points on the first contour to form the top face of the Box. (c) The back contour's edges are connected, but the middle one remains unconnected. Connect point P_i on the middle contour to points P_{i-1} and P_{i+1} on the other two contours to form one set of polygons. (d) Connect point P_i on the first contour to point P_i on the contour ring to form the other polygons (with two edges from the previous edge creation step).

to the other points. The second method requires that the points along every other contour remain unconnected (i.e., points along the contour path are not connected). In order to build triangular polygons, each connected contour is connected to both of the adjacent unconnected contours as well the next pair of connected contours. This forms two distinct forms of triangles. One set are long and are formed by corresponding points on two connected contours and one point on the unconnected contour. The other set of triangles are formed by two points on a connected contour and one on the unconnected contour. This method has one restriction; the UPO must use an odd number of control rings (3 or more). This method produces extruded surfaces and prism objects accurately. Figure B.5 demonstrates how a rectangular box can be built using this method. The general principle for creating geometric models from this method is:

Method 2 *Each contour has N points. The points along contour R_n are unconnected and the points along contours R_{n+1} and R_{n-1} are connected. Point P_j on R_n is connected to points P_j and P_{j+1} on R_{n+1} and R_{n-1}. In addition, P_j on R_{n-1} is connected to P_j on R_{n+1}. For the case of $P_{j=N+1}$, use point P_1.*

End Points

In addition to control contours, the UPO has two end connection points. These tie all of the points of the first and last contours to a central end point to form a cap for the surface. This preserves the desired triangular polygon requirement. As an added benefit, it is easier to define certain objects like cylinders with cone tops and bi-pyramidal models.

Parameters used by UPO to build any primitive object:

- Number of points per control contour (N_{pt}).
- Number of control contours (N_{ring}).
- Flag whether control contours are to be built or are pre-defined.
    1. Pointer to Building Contour Parameters (stored in order).

- Center location: $\vec{C}(n)$
- Scale (before orientation): $\vec{G}(n)$
- Center orientation: $R(n)$
- Use to generate the Array of Points used to build object.

2. Pointer to Pre-defined Contour Parameters (stored in contour order).
   - Array of predefined points.
   - This array is simply copied to final object.

- Flag whether to alternate contour connections or not.
- Location of two end points.

Algorithm for Building UPO

1. Create Point, Edge and Polygon Arrays.
   - Create Point Array (Array Size = $N_{ring}N_{pt} + 2$).
   - Create Edge Array (Array Size = $3N_{ring}N_{pt}$).
   - Create Polygon Array (Array Size = $2N_{ring}N_{pt}$).
2. Load UPO Point Array
   (a) Contours Built by UPO
   - Load first end point.
   - Determine angle between points on each contour: $\theta = 2\pi/N_{pt}$
   - Build successive contours beginning with contour closest to first end point.
     - Create temporary point array for contour n.
     - Determine extra half angle rotation: $\alpha = \theta(1 - mod(n, 2))/2$
     - Use unit radius to compute points in temporary XY plane: $P_x(i) = cos\phi_i, P_y(i) = sin\phi_i$ where $\phi_i = i\theta + \alpha$
     - Use 2D scaling matrix to scale contour: $\vec{P}'(i) = \mathbf{S_n} \cdot \vec{P}(i)$.

- Rotate contour about its own origin: $\vec{P}''(i) = \mathbf{R_n} \cdot \vec{P}'(i)$.
- Translate contour to center point in POCS and store point: $\vec{P}_{final}(j) = \mathbf{T_n} \cdot \vec{P}''(i)$ where $j = nN_{pt} + i$.
- Repeat for each contour.

• Load second end point.

(b) Contours Pre-defined by User

- Load first end point.
- Load pre-defined array of points.
- Load second end point.

3. Build First Cap of UPO (Edge and Polygon Relationships).
   - Build edges from first end point to first contour's coordinates.
   - Build first contour's edges.
   - Build top polygons from the three edges.
   - Associate edges with cap polygons.
4. Build Multiple Contour Connections.
   (a) Method 1 - All contours are connected.
   - Build diagonal edges from first contour to second contour.
   - Build polygons using adjacent diagonals and edge of first contour.
   - Build edges for second contour.
   - Build polygons using adjacent diagonals and edge of second contour.
   - Associate edges with polygons.
     - First contour: second polygon defined now.
     - Second contour: first polygon defined now.

   (b) Method 2 - Alternating between connected and unconnected contours.
   - Build diagonal edges from connected contour to unconnected contour.

- Build vertical edges between connected ring and next connected ring.
- Build polygons with two adjacent diagonal edges and the edge of first connected contour.
    - Adjacent diagonals do not have a vertical edge between them.
    - Both connected contours form polygons with diagonals.
- Begin to build vertical polygons with vertical edge and one diagonal.
    - Two of three edges of polygon are defined.
- Build diagonal edges from unconnected contour to second connected contour.
- Finish building vertical polygons with the corresponding diagonal edge.
- Build edges for second connected contour.
- Build third set of polygons with two adjacent diagonal edges and second contour edge.
- Associate edges with polygons
    - First connected contour: second polygon defined now.
    - Diagonal and Vertical edges: both polygons defined.
    - Second connected contour: first polygon defined now.
- Build next set of contours if necessary.

5. Build Second Cap of UPO.

    - Build edges from last contour to end point.
    - Build polygons from two adjacent edges to end point and contour edge.
    - Associate edges with polygons.

In order to define various point symmetric or axial symmetric objects, the user only has to define a few general parameters. A couple of examples are given in table B.1.

The addition of storing polygon information in the edge structure requires no special procedure. One simply adds the step of linking the polygon to the edge right

Parameter	*Box*	*Ellipsoid*
N_{ring}	3	3 or more
N_{pt}	4	$>= N_{ring}$
Build Contours	Yes	Yes
Alternate Contours	Yes	No
End Point Locations	Center of R_1 & R_3	Use Equation
Contour Radius	$R_1 = R_3$, $R_2 = \sqrt{2} \cdot R_1$	Use Equation with N_{ring} & N_{pt}

Table B.1: Parameters for building Ellipsoid and Box UPO

after linking the edge to polygon. In addition, the fact that only the point array stores information makes translation, scaling and rotation a simple exercise of transforming the points; the integrity of the model is maintained.

Once the geometric model is defined, the volumetric data is associated with a specific UPO. This data is stored either as an array of floating point or integer values or as a single integer index which links the UPO to another structure that contains the data. In either case, the UPO simply must load and store this data. The main advantages of the UPO are seen when interpolating UPO volumes onto the Grid.

B.4 Interpolating a UPO onto a VGO

As mentioned earlier, the "slice and dice" methodology for interpolating a bounding surface with a three-dimensional, orthogonal grid works adequately to load the volumetric character of a bounded volume while maintaining a *voxel approximation* of the original surface. Imagine dividing a three-dimensional block of space into a set of regular rectangular boxes called voxels. Each voxel has the same dimensions and represents the bounding box that surrounds an individual grid point. All properties of the voxel are stored by this grid point. Let a series of volumes be placed into this grid space, with the restriction that their volume must ultimately be represented by these voxels. Therefore, each grid point contained inside one of these volumes would be loaded with the characteristics of the bounded volume. Figure B.6 shows an example of a cone being interpolated into a three-dimensional grid. The final result is a series of voxels which have a different set of characteristics from the rest of the voxels, i.e., in figure B.6, they are opaque and not transparent.

In general, the characteristics stored in each voxel might include information such as indexes to other data sets, various gas or chemical concentrations, temperature, absorption, etc. All of the uninterpolated voxels would contain a default form of these characteristics. If an algorithm were to "drill" into or "ray trace" through the grid, it could quickly create a profile of these characteristics along the direction that is "drilled" or "traced." Such a situation is discussed in section B.5.

B.4.1 Intercepting a Cutting Plane with the Surface of a UPO-based Geometric Model

The "slice" portion of the algorithm requires a regularly spaced set of cutting planes which are perpendicular to the Z axis of the VGO. Each of these cutting planes is spaced so that it represents a single plane of voxels in the XY plane. In other words, each layer of voxels along the Z axis has its own associated cutting plane. Figure B.7 shows that the cutting planes pass through the center of each layer of voxels so that the planes coincide with the locations of the actual grid points. Once the cutting planes are defined, the bounded surface is intercepted with the plane. This plane-surface interception may create a single curve or a series of curves in the slicing plane. If the surface is bounded, these curves will be closed. If the surface is also non-intersecting (i.e., surface does not intersect itself), then the curves will also be non-intersecting. The closed curves can then be intercepted with the two-dimensional layer of voxels which lie in the cutting plane. Since the Universal Primitive Object is bounded and non-intersecting, it is ideal for use with this algorithm.

When intercepting a UPO-based surface with a cutting plane, one can see the advantage of the additional link between any edge and the two polygons that share the edge. Most methods of intercepting a plane and a polygon surface model involve cycling through each edge to determine which edges are intercepted. Although excellent for hidden surface removal or getting interception points, if you are interested in generating the intercepted curve, it is extremely difficult to do because the order

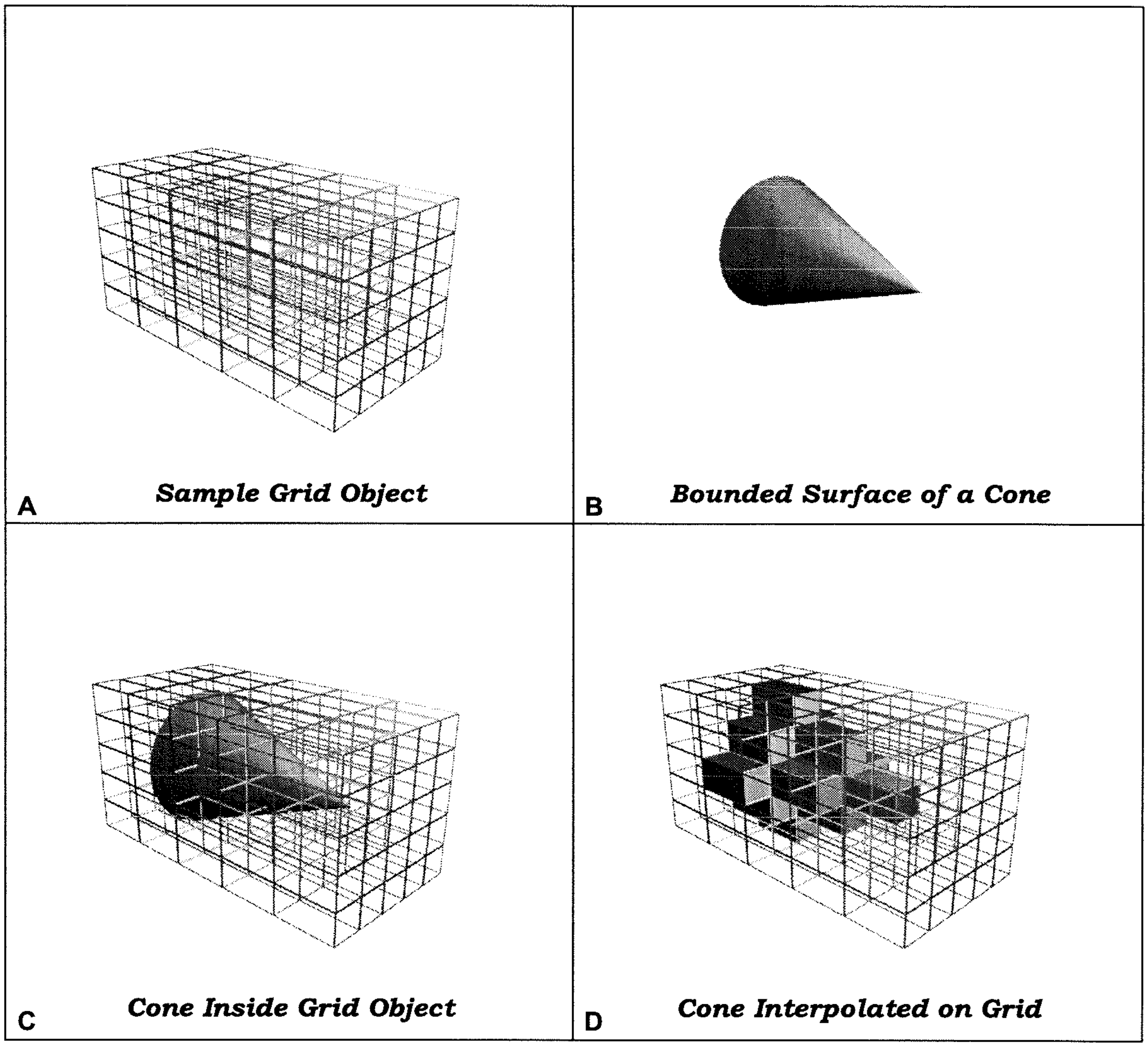


Figure B.6: Example of a UPO interpolated on three-dimensional VGO. (a) A wire-frame of a 5x5x5 VGO. (b) A cone as the bounded volume to be interpolated. (c) The cone placed inside the VGO. (d) The final interpolation of the cone onto the VGO.

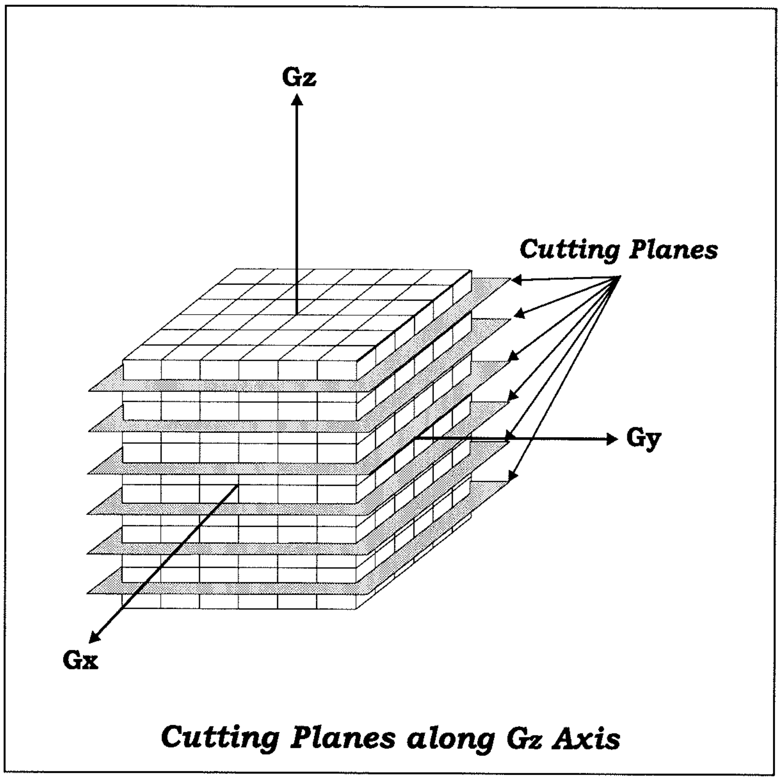


Figure B.7: Cutting planes going along Z axis of the VGO.

of the intercepted points is variable. Using a UPO-based surface eliminates these problems because of this additional link between each edge and the polygons that share it.

The UPO reduces a difficult problem to a simple methodology. Once the UPO is placed and oriented in the WCS, it can be transformed into the GCS of the proper VGO. Since the UPO's point array is the only place where the physical coordinates are stored, only this point array is affected by the transformation to the GCS. Once in the GCS, a plane-surface interception is reduced to cutting the UPO with a set of parallel XY planes. Furthermore, the UPO remains in the GCS and all of the interpolation algorithms are conducted in the VGO's GCS. Once the interpolation is complete, the UPO is returned to the WCS and may be reused again.

When the UPO-based surface is sliced by a cutting plane, the algorithm cycles through the edge array to find all of the edges intercepted by the plane. The GCS coordinates of the interception point where each edge and plane coincide are stored in a data structure. In addition, the two polygons that share the intercepted edge are also stored in the same data structure. Once all of the edges are found, the algorithm cycles through all of the intercepted points and uses the indexes to the stored polygons to determine how to order the interception points into a set of piece-wise continuous curves.

The algorithm begins with the first point in the array of intercepted points. It loops through the other points until it finds another point that shares one of the two polygons associated with the first point. Once the second point is found, then the algorithm uses the polygon not shared by the first two points to find the third point. At the same time, it stores the index to the other polygon associated with the first point as the *end polygon index.* The algorithm continues the cycle until it finds a point that has the same other polygon index as the end polygon index. This establishes that this point is the final point of the curve and that the curve is closed. Once the curve is closed, the points associated with the curve are stored in a separate array and removed from the intercepted point array. The algorithm continues until there are no more points left in the intercepted point array. The final result is one or more closed curves representing the cutting plane-UPO surface interception.

B.4.2 Ordering the Intercepted Closed Curves

As mentioned, an alogrithm for intercepting a closed curve with a two-dimensional grid has already been developed [9]. However, this algorithm deals strictly with unordered curves intercepted with a 2D grid. It does not consider any relationship between the curves intercepted on the grid. However, there may be times when the slicing operation may result in multiple closed curves being inside one another (see figure B.8). Not only must the curve be interpolated on the grid, but the order of the curves interpolated must also be considered. Otherwise, the final result might not be what is desired (see figure B.9). Therefore, the slicing algorithm includes a method for sorting the curves from the outer most curve to the inner most one. Then, the "dicing" algorithm can feed the curves in the proper order to the well-defined curve-grid interception algorithm.

There are various methods to sort a series of curves. Our algorithm uses the piece-wise continuous nature of the curves. The piece-wise nature allows us to define the following geometric property:

Given: *A 2D plane that contains a line segment and a series of non-intersecting, piece-wise continuous, closed curves.*

Result: *One can determine the number of curves that contain all or part of the line segment.*

Method: *Extend a ray to infinity along one of the two directions of the line segment. Count the number of times the ray strikes each curve. If the ray strikes a curve an odd number of times, then the line segment is inside the closed curve. If the ray strikes the curve an even number of times, then the curve does not contain the line segment. Figure B.10 graphically demonstrates this methodology.*

Given that all of the curves that come from the bounded surface-plane interception are piece-wise continuous and non-intersecting, then the above method can be used to sort the curves. Since each curve is made up of line segments and is non-intersecting, one can choose any line segment that makes up part of the curve. Using this method, the interpolation algorithm determines which curves contain the line segment and, subsequently, the curve. Given that none of the curves intersect, i.e., each curve either lies completely within or outside another curve, the algorithm orders each curve by the number of other curves that it lies within. For example, if a curve lies within two curves, it is a level two curve. If a curve does not lie within any curves, it is a level zero curve. Thus, the curves can be sorted from the outer most curves (level 0) to the inner most ones (level N).

B.4.3 Using Closed Curves to Store Data on VGO

Once the curves are sorted, the "dicing" portion of the algorithm begins. The algorithm we use for intercepting a piece-wise continuous, closed curve with a two-

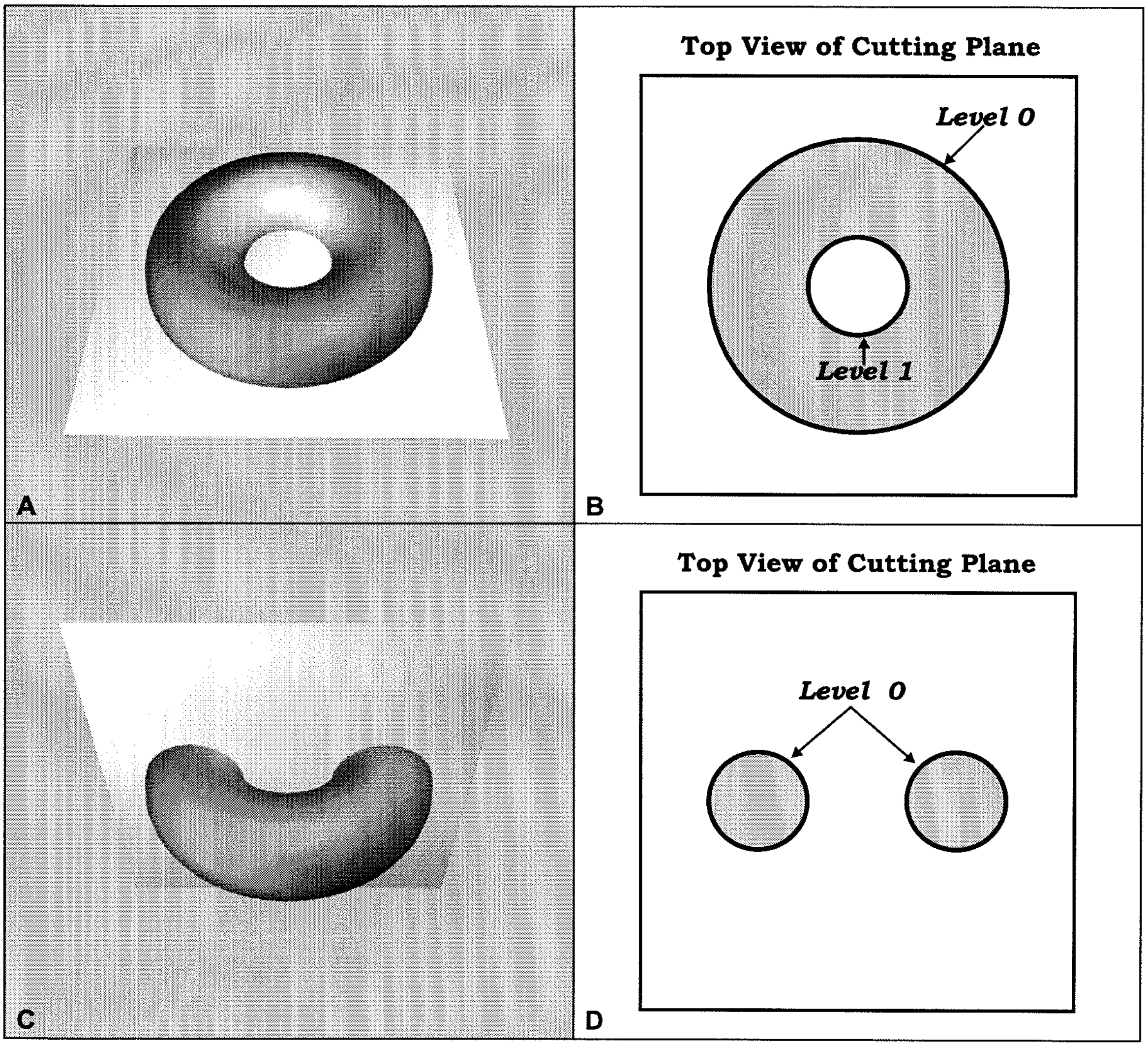


Figure B.8: Example of two types of multiple curve interpolation. (a) Cutting plane passing through Torus to form top and bottom. (b) Result of slice in **A** is one curve inside the other. (c) Cutting plane passing through torus to form two unconnected pieces. (d) Result of slice in (c) is two independent curves.

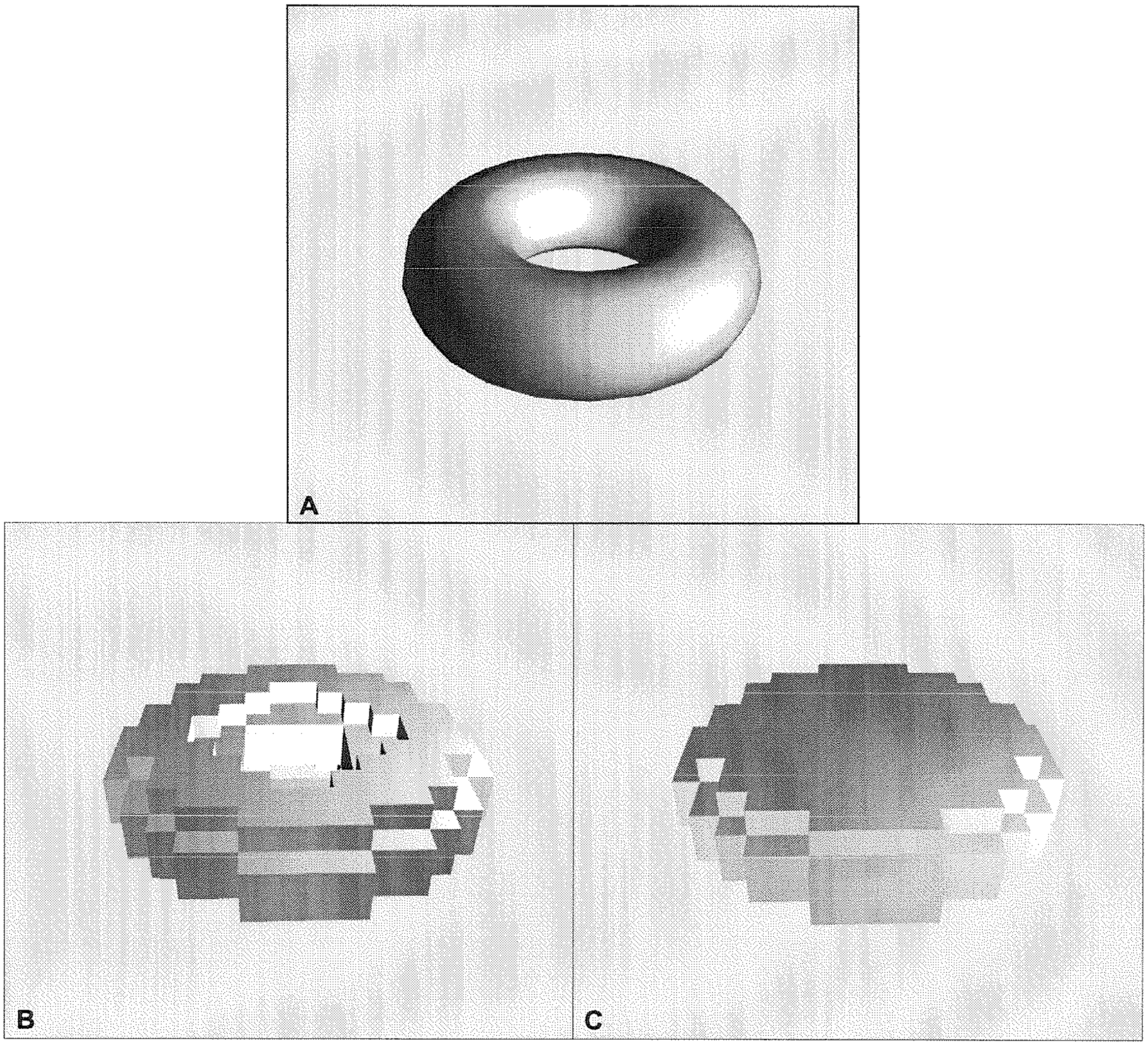
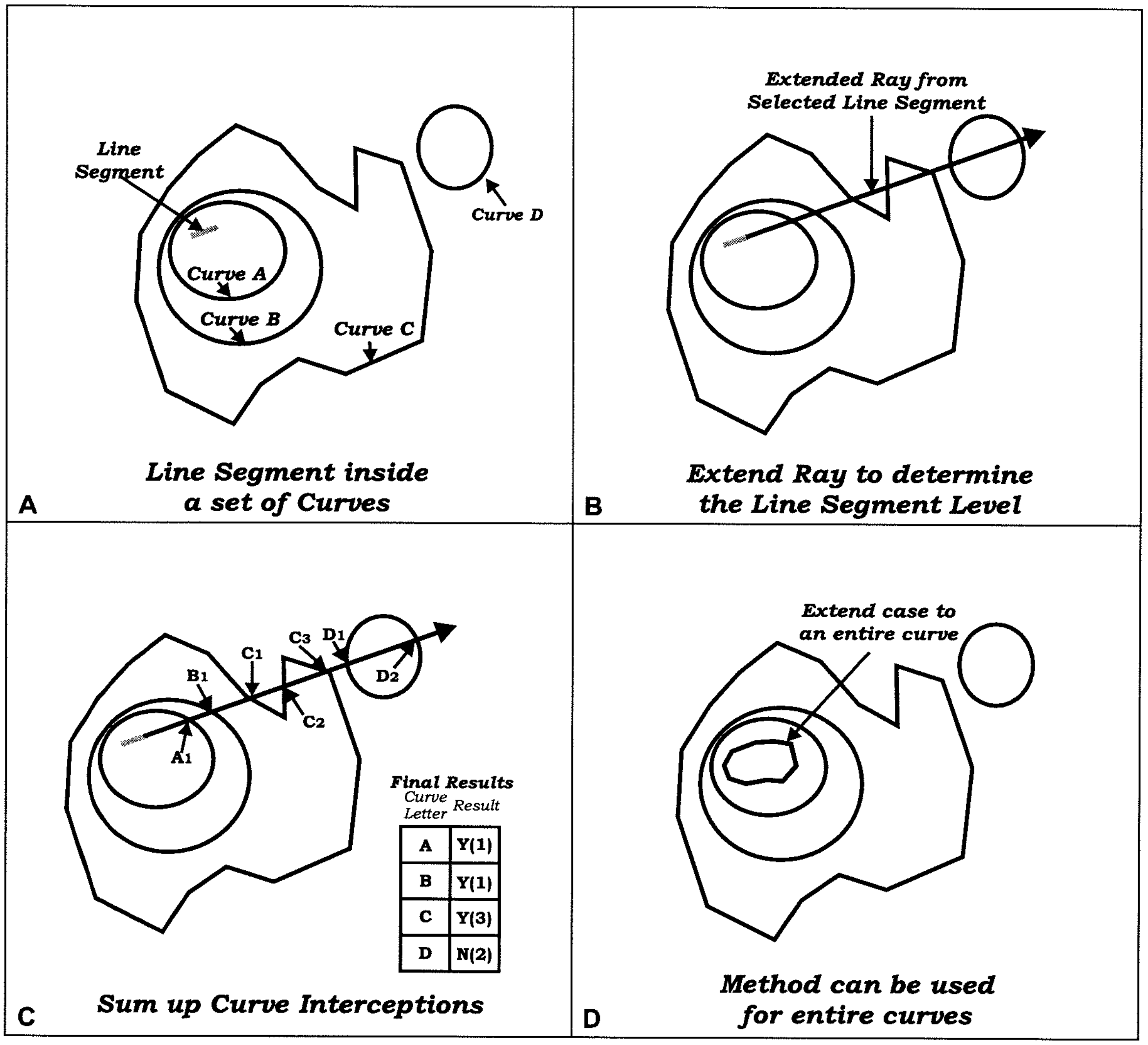


Figure B.9: Two examples of a Torus interpolation. (a) Bounded surface model of Torus. (b) A correctly interpolated Torus. (c) An incorrectly interpolated Torus (no internal ring).


Line Segment
Curve A
Curve B
Curve C
Curve D
A
Line Segment inside a set of Curves
Extended Ray from Selected Line Segment
B
Extend Ray to determine the Line Segment Level
C3
D1
C1
B1
D2
C2
A1
C
Sum up Curve Interceptions
Extend case to an entire curve
D
Method can be used for entire curves

Final Results

Curve Letter	Result
A	Y(1)
B	Y(1)
C	Y(3)
D	N(2)

Figure B.10: Method for determining how many curves contain a line segment or curve. (a) Example of a line segment inside a series of non-intersecting curves. (b) Extend ray along one direction of line segment. (c) Count the number of intersections. If odd number, then line segment is inside curve. If even, the line segment is not inside curve.. (d) A curve containing the line segment also is inside the same number of curves if all of the curves are non-intersecting.

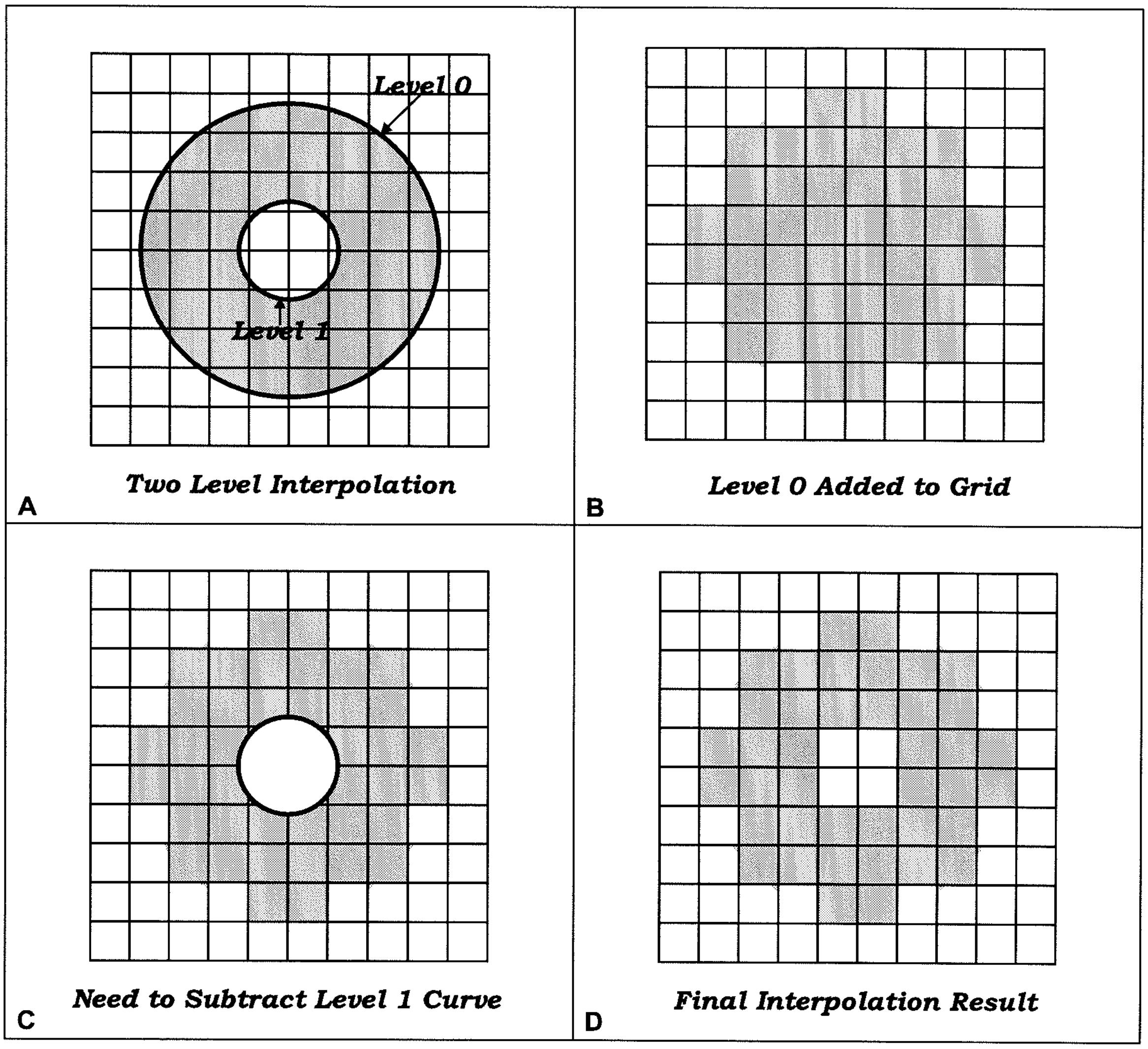


Figure B.11: Example of how two curves are interpolated on a 2D grid (Curves from figure 8b). (a) Two curves overlaid on grid. (b) Outer curve interpolated first places volume data in the grid (An even level # is additive). (c) Second curve overlayed on interpolated grid demonstrates what grid points should be returned to default value. (d) Both curves interpolated on the Grid.

dimensional grid determines the area of each grid rectangle that is contained inside this curve. If the area is greater than 50%, then the associated voxel is marked as being inside the object. Otherwise, the voxel remains unchanged. If a solid is intercepted and it creates multiple curves on the cutting plane, then these curves may represent holes or parts that are not connected to the object in the cutting plane. The level number of the curves determines which curves will encompass voxels and which curves will return changed voxels to their original state. For instance in figure B.11, the outer most curve, *level 0*, encompasses every grid rectangle inside the curve. However, the *level 1* curves define holes in the solid and, consequently, subtract the effected area from those grid boxes contained within both level 1 and level 0. Furthermore, a *level 2* curve would define areas in these holes which are again part of the solid. In summary, all of the even numbered levels are treated normally by the algorithm and any area of the grid rectangles inside these curves is added to the effected area of each of these grid rectangles. All of the odd number levels return the area inside the curve to normal by subtracting the effected area from each effected grid rectangle. In this way, the curves are "diced" and volumetric data is interpolated on the VGO. Once an entire UPO-based surface is interpolated on the VGO, it is no longer needed and can be discarded or returned to the WCS for use again.

B.4.4 The Slice and Dice Algorithm

1. Transform UPO Surface into Grid Coordinate System (GCS).
    (a) Create Geometric Model as a UPO and transform it to WCS.
    (b) Transform point array of UPO into GCS.
2. Compute bounding box for object in GCS.
    (a) Cycle through point array to get absolute minimum and maximum coordinates.
    (b) Use these two points as the basis for the bounding box.

3. Use bounding box to determine what cutting planes to use.
    (a) Use bounding box Z coordinates to define first and last cutting planes.
    (b) Define a loop to create cutting planes to use in interpolation.
4. Cycle through cutting planes to intercept volume with voxel layer.
    (a) Create cutting plane object using Z value (i.e., parallel with XY plane).
    (b) Pass the UPO surface model to Plane program object to determine the interception curves.
        i. Cycle through edge lines of polygons to find all of the intercepted edges.
            - Store intercepted point.
            - Store indexes to two polygons shared by edge.
        ii. Cycle through intercepted points to determine all of the curves intercepted.
        iii. Sort curves into levels.
    (c) Starting with level 0, intercept curves onto the voxel layer.
        i. Create a temporary two-dimensional grid.
            - Same grid point placement and size as the original grid in XY plane.
        ii. Intercept curves onto the VGO using polygon-grid interception routine.
        iii. Store intercepted areas for each grid rectangle in temporary grid.
            - Odd numbered levels subtract from intercepted areas.
            - Even numbered levels add to intercepted areas.
        iv. Use intercepted areas of temporary grid to load the voxel layer of the VGO.
            A. If at least 50% of the area of a grid rectangle is inside the UPO:

- Pass a unique object index to VGO grid point.
- Pass any other values associated with UPO surface model to VGO grid point.

B. If less than 50% of the area is inside the UPO, skip the grid rectangle.

(d) Repeat process for each cutting plane until entire volume is intercepted.

B.5 Application of the VGO and the UPO in the IRIMAGE Simulator

One application of the VGO is modeling a background atmosphere in the infrared (IR). Polluted atmospheres generate specific IR signatures which may be detected using IR detectors and focal plane arrays (FPA). The modeling of such atmospheres and imaging systems should make it possible to determine what conditions are necessary to passively image various gaseous pollutants in the atmosphere. In order to model an atmosphere, one must be able to define the conditions everywhere in the viewable area. One way is to use a three-dimensional grid structure which can approximate these conditions by cutting up space into smaller boxes filled with a homogeneous gas.

The IR system simulator, *IRIMAGE*, uses a VGO and a set of UPO-based volumes to approximate a clean atmosphere that has regions of higher pollution concentration. In the simulator, all or part of the atmosphere is represented by a three-dimensional VGO. Any polluted region is represented by a bounded volume in the form of a UPO. Different atmospheric conditions are stored in the form of two databases: gas concentration models and aerosol models. Therefore, each UPO is associated with a particular atmospheric model through the database indexes of a gas model and an aerosol model. In addition to these two index values, the UPO also stores its own object index. Subsequently, the VGO initializes the OIA to store the object index and the FDA to store two integer values for each voxel. Initially, the VGO loads each voxel with a pair of default gas and aerosol model indexes which represent the initial atmospheric conditions. As the simulation proceeds, various UPO's are interpolated on the atmosphere VGO using the interpolation method specified in section 4. IRIMAGE then ray traces through the VGO to determine an atmospheric profile along the ray. The profile is passed along with a value for the background scene temperature and emissivity to the atmospheric modeling algorithm, MODTRAN [10]. In MODTRAN, the final radiance for the background scene and atmosphere is computed and returned to the imaging system. Since most atmospheric systems are dynamic, the simulation utilizes a "key frame" animation extension to the UPO. The rest of the section details how IRIMAGE takes advantage of the VGO and UPO when computing the incident radiance on a detector system.

B.5.1 Generating an Atmospheric Profile in IRIMAGE

IRIMAGE utilizes a backward ray tracing approach to compute the atmospheric profile observed by each element of the simulation's Focal Plane Array (FPA). The advantage of using a ray tracing technique with the VGO is the relative ease of intercepting a ray with a three-dimensional, orthogonal grid and generating a gas and aerosol model profile along that ray. In addition, the ray also intercepts a two-dimensional background scene (a 2D imaging VGO behind the atmosphere VGO) to get the temperature and emissivity of the background. Most ray tracing applications utilize a ray tracer to incorporate accurate reflective and refractive effects to an image. However, the voxel nature of the VGO prohibits using this method to approximate any reflective or refractive effects. Overall, the greatest advantage of the Ray-VGO interception methodology is the fact that the VGO remains fixed over time and just the contents of the VGO change in time. Therefore, each ray only needs to be traced once if the ray stores each intercepted voxel's grid point index in an ordered stack of indexes. In addition, it must also store the path length of the ray through each voxel. With this information stored in an ordered stack, the ray generates a profile by accessing each intercepted voxel and retrieving its current state. If the state of the voxels changes because a new UPO has been interpolated on the VGO, the ray simply rebuilds the profile using the new state of the voxels along its path.

As mentioned, the rays store the grid point indexes of the intercepted voxels and not the actual gas and aerosol models stored by each grid point. Therefore, rays can be built and intercepted with the VGO before loading the UPO-based bounded volumes onto the VGO. Each ray is defined by a starting point, a direction vector and an end point (which is where the ray strikes the imaging VGO of the background scene). Using this information, each ray is intercepted with the VGO. The interception method uses a simple plane-line intersection algorithm that takes advantage of the uniform box nature of the voxels in the VGO. Each voxel is treated as the interception of six planes. Thus, the entire VGO's grid structure may be defined by extending planes that are perpendicular to each axis and are spaced according to the scaling vector, $\vec{G}$, of the VGO. Once the VGO is defined in this manner, then the ray-VGO interception reduces to a series of ray-plane interceptions. Each ray-plane interception point is stored temporarily, but in the correct spatial order in which the ray strikes each plane. Once the ray either passes entirely through the atmosphere VGO or strikes the imaging VGO, the algorithm cycles through the ray-plane interception points to determine which voxels were intercepted. As it cycles, it computes the midpoint between adjacent interception points. It passes the coordinate of the midpoint to the VGO which returns the nearest grid point index. This establishes that this portion of the ray passes through the grid point's voxel. Using these same two interception points, the algorithm computes the distance between adjacent points and stores that as the length of the ray in that particular voxel. Once all of the grid point indexes and path lengths are determined, the ray is capable of generating an atmospheric profile (see figure B.12).

Once the rays have been generated and intercepted with the VGO, the various UPO-based volumes can be intercepted with the VGO. The position, orientation and scale of the volumes in the WCS are controlled by the KeyFrame program object. This program object stores a set of "key frames" which define specific spatial conditions of the UPO for a specific set of frames. All the frames between the key frames are

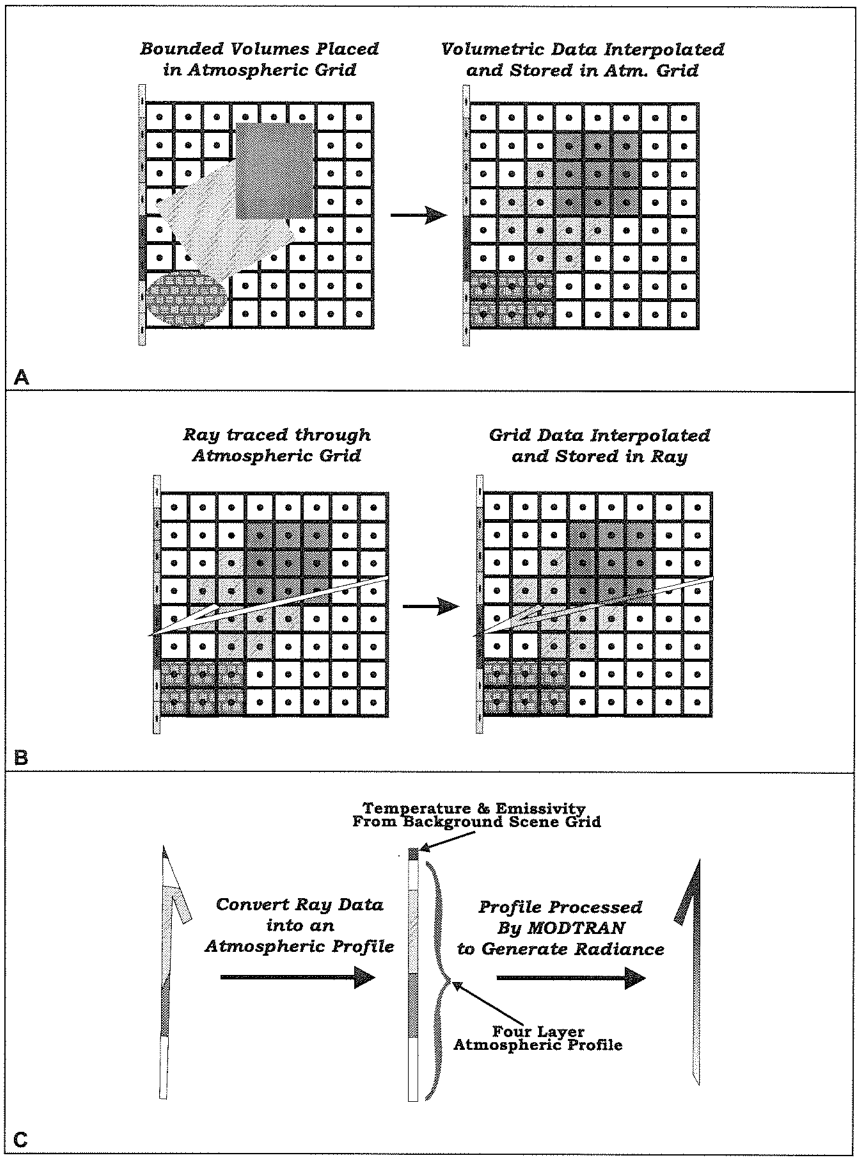

Bounded Volumes Placed in Atmospheric Grid
Volumetric Data Interpolated and Stored in Atm. Grid
A
Ray traced through Atmospheric Grid
Grid Data Interpolated and Stored in Ray
B
Temperature & Emissivity From Background Scene Grid
Convert Ray Data into an Atmospheric Profile
Profile Processed By MODTRAN to Generate Radiance
Four Layer Atmospheric Profile
C

Figure B.12: A 2D example of a how an atmospheric profile is generated. (a) Various atmospheres are interpolated onto the VGO. (b) Ray passes through the VGO and generates the stack of grid points that it intercepts. Then it retrieves the current atmospheric indexes for the profile. (c) IRIMAGE converts stack of values into an atmospheric profile. MODTRAN converts this atmospheric profile into an array of radiance values to be input back in the imaging system.

interpolated linearly. So instead of having to define the conditions for each frame, the user only needs to determine size, location and orientation of the UPO for certain frames. In addition to the animation of individual UPOs, the order in which the UPO's are loaded is also important. The layout order determines which UPO's take precedence on the VGO. Since there is the possibility of loading two UPO's in the same space on the VGO, the layout order determines which UPO should be loaded last (i.e., be the last to adjust the VGO). Once the UPO-based volumes are loaded on the VGO for a particular frame, each ray can compute the atmospheric profile along the ray.

Once the rays have been intercepted and the UPO's have been loaded, IRIMAGE can generate the atmospheric profile for any ray. IRIMAGE uses the stack of grid point indexes stored by each ray and retrieves the gas and aerosol model database indexes for each grid point. These indexes are placed in an array along with the path length through the associated voxel. IRIMAGE processes this new array until it has reduced the voxel stack to a smaller set of atmospheric layers which have different gas and/or aerosol models. Each layer is the combination of several adjacent voxels that share the same gas and aerosol index. IRIMAGE just combines these voxels into a single layer whose path length is the sum of all the affected voxels. Once the profile is generated, it is passed onto MODTRAN which computes the radiance along the ray's path. From this radiance, IRIMAGE is able to determine the output signal from a well-defined imaging system.

B.5.2 Example of Various UPOs Used in IRIMAGE

In figure B.13, a polluted region was modeled using four different UPO-based geometric models. The VGO was intially set to be a standard 1976 U.S. Atmosphere at 293K and 760 torr. Each UPO model used the same standard atmosphere parameters as the default conditions loaded on the VGO except that the temperature was 350K and the concentration of Carbon Monoxide (CO) was raised to 50,000 part per million (ppm) or 5% by volume. The standard atmosphere of the VGO had a temperature of

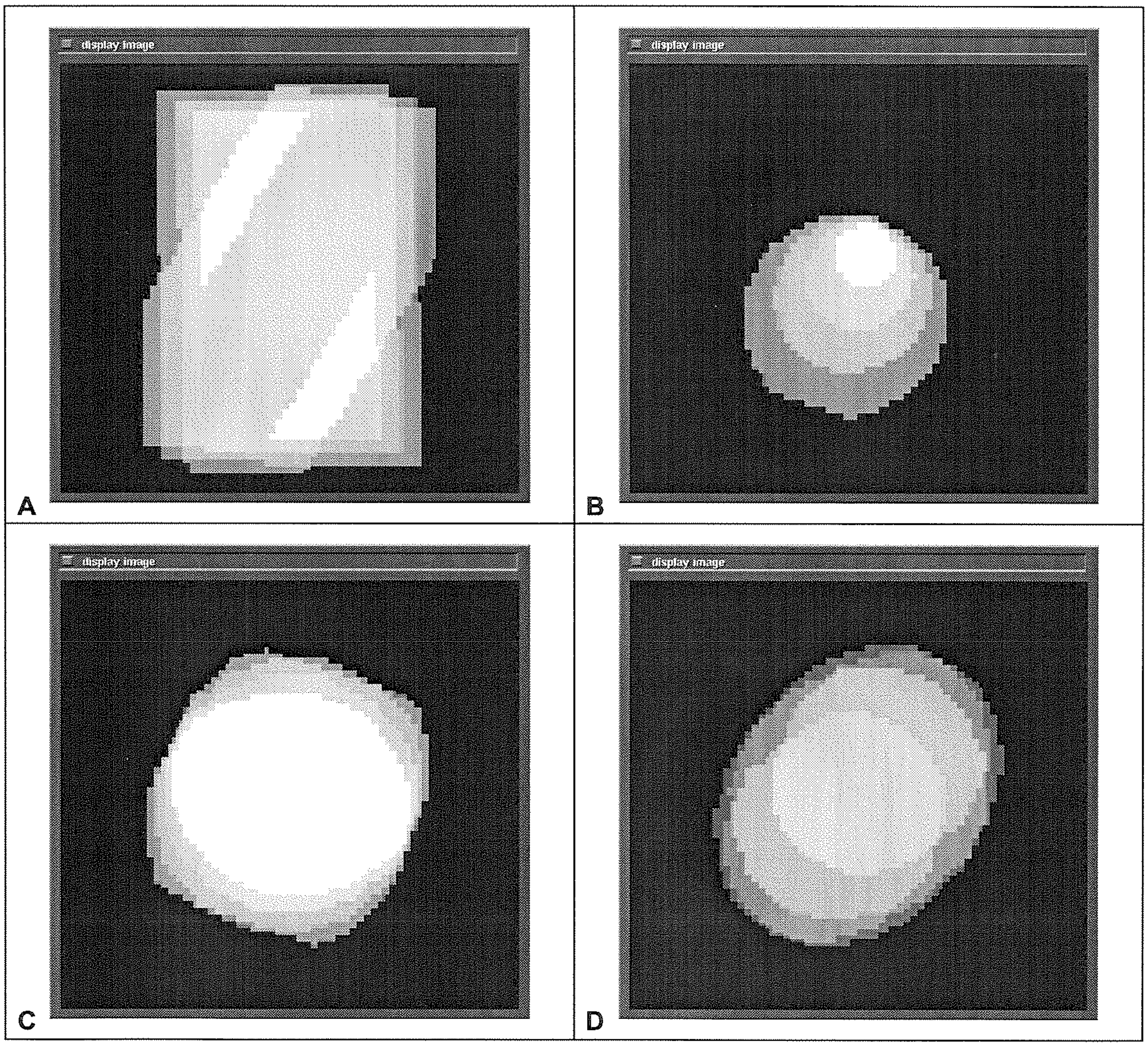


Figure B.13: Various UPO-based bounded volumes representing a polluted region of 5% carbon monoxide at 350K in a standard 1976 U.S. atmosphere at 293K. (a) Box, (b) Cone, (c) Cylinder, (d) Ellipsoid.

293K and had standard levels of CO. The levels of CO in a standard atmosphere are between 0.05 to .2 ppm according to Seinfeld [11]. In addition, Seinfeld shows that the heightened CO level we used for the bounded volumes is typical for the output of an internal combustion engine prior to the exhaust entering the catalytic converter. The background imaging VGO had a fixed temperature of 293K and an emissivity of .9 (90% of a blackbody). The various images generated clearly demonstrate how a bounded volume based on the UPO construct can be loaded onto the VGO. In addition, one can see how the ray-VGO intersection algorithm creates a reasonable approximation of a polluted atmosphere's profile.

B.6 Conclusions

The object-oriented design of the VGO and UPO makes these objects very flexible. Although major or minor modifications can be made to either object, these changes won't effect how other routines use the VGO or the UPO as long as the modifications don't effect how their routines are called. This reduces the amount of recoding required to make a change or improvement. For instance, the VGO has no direct relationship to the various objects (image, UPO, etc.) placed upon it. Therefore, new geometric models and interpolation algorithms can be implemented which do not affect the internal structure of the VGO. At the same time, different algorithms for accessing the FDA or the OIA can be implemented which add new capabilities to the VGO and do not affect its current routines. Besides modifying the functionality of the VGO and the UPO, their object-oriented design provides the user and/or programmer with the flexibility to define new primitive surface models or custom forms of the VGO. A programmer can tailor certain aspects of the VGO based on the requirements of the application, the speed of the machine and the memory available for the program. The user may add new types of primitives to the UPO library by developing ways to define these primitives using the UPO's methods of connecting contours. Another advantage to an object-oriented approach is the multiple reuse of the same object in the same program or another program. For example, IRIMAGE uses three different implementations of the VGO: a 2D background imaging VGO, a 3D Atmosphere VGO, and a 2D FPA layout VGO. Using a piece of code multiple times not only cuts down on how much code is written, but also reduces the amount of memory used by the program to store the various routines. Overall, these factors make the VGO and the UPO good examples of useful tools which can be maintained and improved without affecting their functionality.

Despite all of its advantages and benefits, the VGO still has some drawbacks. The ray tracing method is effective at determining an approximate profile, but the voxel nature prevents the reflective and refractive advantages of a ray trace from being exploited. This limits the VGO's usefulness to applications which do not require these properties. In addition, the regular spacing of grid points can create unnatural effects for complex and slow moving objects (jerkiness, "flashing pixels," etc.) These problems are inherent with any representation of space which uses a voxel approximation. However, most of these problems can be avoided or minimized with good program design and careful planning of how to use the VGO.

The algorithm for interpolating UPO-based bounded volumes onto the VGO works quite well. It sufficiently flexible to handle any type of geometry that can be built using the UPO as a basis. However, the algorithm still has room for improvement. Currently, the scale and number of grid points determines the accuracy of the interpolation of the bounded volume. Large numbers of grid points spaced close together give excellent representations of the volumes but at the cost of huge amounts of memory. Smaller numbers of grid points spaced far apart will severely limit the accuracy of the model for most cases, but it limits the amount of memory used and reduces the time required to compute the interpolation. Unfortunately, it is very hard to determine the perfect balance. The accuracy is also limited by the fact that a single cutting plane determines how an entire plane of grid points is interpolated. Multiple planes could be implemented that would more accurately determine what perentage of the *volume* of the voxel is contained inside the bounded volume instead of using a single *area* value. Finally, the current interpolation algorithm does provide a method of combining data sets that overlap in the same voxel. This is all right for solid objects, but inadequate for gaseous and liquid objects which can intermix. Again, the user can attempt to approximate this property by loading "mixed" data sets and placing them at these boundries.

In general, the VGO, UPO and the interpolation algorithm provide the basis for a new approach to modeling three-dimensional scenes. Although there are certain drawbacks, the object-oriented design and their more generalized nature make it possible to improve their capabilities over time. The additional techniques of ray tracing through the VGO, and the simulation of temporal behavior exploited by IRIMAGE, demonstrates that this methodology is effective in simulating behavior that is not heavily dependent upon surface representations.

Bibliography

[1] J. Foley, A. van Dam, S. Feiner, J. Hughes, and R. Phillips. *Introduction to Computer Graphics.* (Addison-Wesley, Reading, MA, 1990), pp. 377-81.

[2] Discussion of Constructive Solid Geometry: J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Priciples and Practice, Second Edition.* (Addison-Wesley, Reading, MA, 1990), pp. 557, 712-714.

[3] Excellent review of octrees: H. Samet. *Design and Analysis of Spatial Data Structures.* (Addison-Wesley, Reading, MA, 1990).

[4] R. Lohner and P. Parikh, "Three-dimensional grid generation by the advancing front method," *Int. J. Numer. Methods in Fluids*, 8, 1988, 1135-1149.

[5] P.L. George and E. Serveno, "The Advancing-Front Mesh Generation Method Revisted," *Int. J. Numer. Methods in Engineering.* 37, 1990, 3605-3619.

[6] J. Foley, A. van Dam, S. Feiner, J. Hughes, and R. Phillips. *Introduction to Computer Graphics.* (Addison-Wesley, Reading, MA, 1990), pp. 382-83.

[7] J. Foley, A. van Dam, S. Feiner, J. Hughes, and R. Phillips. *Introduction to Computer Graphics.* (Addison-Wesley, Reading, MA, 1990), pp. 180-182.

[8] D.A.P. Mitchel, "Fast Algorithms for 3D Graphics," *Ph.D. Thesis*, University of Sheffield, July 1990.

[9] C. Deutsch, "A FORTRAN 77 Subroutine for Determining the Fractional Area of Rectangular Grid Blocks within a Polygon," *Computers & Geosciences*, 16(3), 1990, 379-384.

[10] F.X. Kneizys, L.W. Abreu, G.P. Anderson, J.H. Chetwynd, E.P. Shettle, A. Berk, L.S. Bernstein, D.C. Robertson, P.K. Acharya, L.S. Rothman, J.E.A. Selby,

W.O. Gallery, and S.A. Clough, “The MODTRAN 2/3 Report and LOWTRAN 7 Model,” *prepared for PL/GPOS*, 1996.

[11] J.H. Seinfeld, *Atmospheric Chemistry and Physics of Air Pollution*, (John Wiley & Sons, New York, 1986).

Appendix C Example of Parameter Files Used by IRIMAGE

This part of the appendix contains an example of the seven parameter files used by IRIMAGE to run a simulation. In this case, these are the parameter files for the Car simulation discussed in chapter 2.5. There are seven files, a single general file and one for each of the major objects of IRIMAGE. You should notice that the format of the files makes it reasonably easy to determine what each of the parameters are.

C.1 The General Parameter File

```
#
# IRIMAGE General Parameter File
#
# Simulation Name: Car
# Created on Date: 9/3/97
#

$ IRIMAGE General Parameters
 Memory Model           = 0
 Total Number of Frames= 90
 Starting Frame Number = 1
 Scene Parameter File  = car_sim/script/car.scn
 Atmosphere Param File = car_sim/script/car.atm
 Detector Param File   = car_sim/script/car.det
 FPA Parameter File    = car_sim/script/car.fpa
 Optics Parameter File = car_sim/script/car.opt
 Output Parameter File = car_sim/script/car.out
 Num of Imaging Systems= 1
 Imaging System Index  = 1
 Output Info to Screen = 0
 Output Info to File   = 0
 Monitor Filename      = car_mon
```

C.2 The Scene Object Parameter File

```
#
# Scene/Image Object (SCN) Parameter File
#
# Simulation Name: Car
# Created on Date: 9/3/97
#


#
# Scene Loading (scene.cxx)
#
$ Scene Parameters
 Origin of Scene        = 0.0, 0.0, 5.0
 Orientation of Scene   = 180.0, 0.0, 0.0
 Scene Grid Spacing     = 2.0e-3, 2.0e-3
 Scene Grid Size        = 400, 400
 Interpolation Method   = 0
 Background Temperature= 293.15
 Background Emissivity = 0.90
 Background Mask        = 0.0
 Use Animation Param.   = 1
 First or Only Frame    = 1



#
# Image Database Loading (image.cxx)
#
$ Image DB Parameters
 Number of images       = 3
 Image File Name        = bars_t50.ascii
 Type of Image File     = 1
 Image Index            = 1
 Corner Location        = 1
 Grid Point:X,Y Spacing= 4.0e-3, 4.0e-3
 Grid Point:X,Y Sizing = 200, 200
 Pos. of Temperature   = 0
 Pos. of Emissivity    = -1
 Pos. of Mask Values   = -1
 ASCII:# of Header Line= 1
 ASCII:X, Y Coords     = 0
 Image File Name       = car_bw.pgmb
 Type of Image File    = 3
 Image Index           = 2
 Corner Location       = 1
 Grid Point:X,Y Spacing= 2.0e-3, 2.0e-3
 Grid Point:X,Y Sizing = 400, 300
 Pos. of Temperature   = 0
 Pos. of Emissivity    = -1
 Pos. of Mask Values   = -1
 Binary:Type of File   = 1
 Binary:Temper. Mult   = -0.1
 Binary:Temper. Add    = 305
 Binary:Emissivity Mult= .0001
 Binary:Emissivity Add = 0.86
 Binary:Mask Mult      = 1.0
 Binary:Mask Add       = 0.0
 Image File Name       = car_mat.ascii
 Type of Image File    = 1
 Image Index           = 3
 Corner Location       = 1
 Grid Point:X,Y Spacing= 2.0e-3, 2.0e-3
 Grid Point:X,Y Sizing = 400, 300
 Pos. of Temperature   = -1
 Pos. of Emissivity    = -1
 Pos. of Mask Values   = 0
 ASCII:# of Header Line= 0
 ASCII:X, Y Coords     = 0


#
# Virtual Image Object Loading (image.cxx)
```

```
#
$ VIO Parameters
 Number of VIOs        = 3
 VIO Object Name       = Bars_Image
 VIO Object Index      = 1
 Image Index           = 1
 Load Temperature Pos  = 0
 Load Emissivity Pos   = -1
 Load Mask Values Pos  = -1
 Interpolate using Mask= 0
 VIO Object Name       = Car
 VIO Object Index      = 2
 Image Index           = 2
 Load Temperature Pos  = 0
 Load Emissivity Pos   = -1
 Load Mask Values Pos  = -1
 Interpolate using Mask= 1
 Minimum value of Mask = 0.0
 Transparency at Min.  = 1.0
 Maximum value of Mask = 100.0
 Transparency at Max.  = 0.0
 VIO Object Name       = Car_Matte
 VIO Object Index      = 3
 Image Index           = 3
 Load Temperature Pos  = -1
 Load Emissivity Pos   = -1
 Load Mask Values Pos  = 0
 Interpolate using Mask= 0


#
# Load Animation Parameters (keyfrm.cxx)
#
# Animation Parameters for Each VIO
#
$ Key Frame Param: Bars_Image
```

```
 Number of Key Frames  = 1
 Interpolation Method  = 0
 Frame Number          = 1
 Object Active         = 1
 Origin Location (WCS) = 0.0, 0.0, 0.0
 Orient. about Origin  = 0.0, 0.0, 180.0
 Scale about Origin    = 1.0, 1.0, 0.0
 # of Interp. Coeff.   = 0
 # of Integer Data Pts = 0
 # of FP Data Pts      = 0
$ Key Frame Param: Car
 Number of Key Frames  = 2
 Interpolation Method  = 0
 Frame Number          = 1
 Object Active         = 1
 Origin Location (WCS) = -.125, -0.05, 0.0
 Orient. about Origin  = 0.0, 0.0, 0.0
 Scale about Origin    = 1.0, 1.0, 0.0
 # of Interp. Coeff.   = 0
 # of Integer Data Pts = 0
 # of FP Data Pts      = 0
 Frame Number          = 90
 Object Active         = 1
 Origin Location (WCS) = -0.375, -0.05, 0.0
 Orient. about Origin  = 0.0, 0.0, 0.0
 Scale about Origin    = 1.0, 1.0, 0.0
 # of Interp. Coeff.   = 0
 # of Integer Data Pts = 0
 # of FP Data Pts      = 0
$ Key Frame Param: Car_Matte
 Number of Key Frames  = 2
 Interpolation Method  = 0
 Frame Number          = 1
 Object Active         = 1
 Origin Location (WCS) = -.125, -0.05, 0.0
 Orient. about Origin  = 0.0, 0.0, 0.0
 Scale about Origin     = 1.0, 1.0, 0.0
 # of Interp. Coeff.    = 0
 # of Integer Data Pts = 0
 # of FP Data Pts       = 0
 Frame Number           = 90
 Object Active          = 1
 Origin Location (WCS) = -0.375, -0.05, 0.0
 Orient. about Origin  = 0.0, 0.0, 0.0
 Scale about Origin     = 1.0, 1.0, 0.0
 # of Interp. Coeff.    = 0
 # of Integer Data Pts = 0
 # of FP Data Pts       = 0

#
# Load Layout Parameters (keyfrm.cxx)
#
# Layout DB Loads VIO Index and Type in Order of Layout
#
$ Layout DB Param: Scene Object Layout
 Number of Layout Obj  = 1
 Layout Key Frame Num. = 1
 Num of Objects in Fram= 3
 Object at Pos #0001    = 1
 Object Type Pos #0001 = 0
 Object at Pos #0001    = 3
 Object Type Pos #0001 = 0
 Object at Pos #0001    = 2
 Object Type Pos #0001 = 0
```

C.3 The Atmosphere Object Parameter File

```
#
# Atmospheric Object (ATM) Parameter File
#
# Simulation Name: Car
# Created on Date: 9/3/97
#

#
# General Atm Object Parameters (atm.cxx)
#
$ Atm Class Parameters
 Origin of Atm Grid     = 0.05, 0.0, 2.5
 Orientation - Atm Grid= 0.0, 0.0, 0.0
 Atm Grid Spacing       = 1.0e-3, 1.0e-3, 1.0e-3
 Atm Grid Size          = 200, 200, 150
 Use Geom Objects       = 1
 Background Gas Model   = 1
 Background Aerosol Mod= -1
 Type of Standard File = 1


#
# Gas Model Database (modtran.cxx)
#
$ Atmosphere: GasDB
 Num. of Models in DB   = 2
 Model Index: 1
 Pressure  Value/Unit   = 7.60e+02, C
 Temperature Value/Unit= 2.931e+02, A
 H2O Density/Unit       = 0.0, 6
 O2 Density/Unit        = 0.0, 6
 O3 Density/Unit        = 0.0, 6
 CO Density/Unit        = 0.0, 6
 CO2 Density/Unit       = 0.0, 6
 CH4 Density/Unit      = 0.0, 6
 NO Density/Unit       = 0.0, 6
 NO2 Density/Unit      = 0.0, 6
 N2O Density/Unit      = 0.0, 6
 NH3 Density/Unit      = 0.0, 6
 HNO3 Density/Unit     = 0.0, 6
 SO2 Density/Unit      = 0.0, 6
 Model Index: 2
 Pressure  Value/Unit  = 7.61e+02, C
 Temperature Value/Unit= 3.232e+02, A
 H2O Density/Unit      = 0.0, 6
 O2 Density/Unit       = 0.0, 6
 O3 Density/Unit       = 0.0, 6
 CO Density/Unit       = 5.00e+03, A
 CO2 Density/Unit      = 0.0, 6
 CH4 Density/Unit      = 0.0, 6
 NO Density/Unit       = 0.0, 6
 NO2 Density/Unit      = 0.0, 6
 N2O Density/Unit      = 0.0, 6
 NH3 Density/Unit      = 0.0, 6
 HNO3 Density/Unit     = 0.0, 6
 SO2 Density/Unit      = 0.0, 6


#
# Aerosol Model Database (modtran.cxx)
#
$ Atmosphere: ArsolDB
 Num. of Models in DB  = 1
 Model Index: 1
 Aerosol Number Density= 1.0
 Equiv. Liq H2O Content= 0.0
 Rain Rate             = 0.0
 Aerosol Model         = 0
 Cloud Model Type      = 0
 Upper Atm Aerosol Mod.= 0
 Seasonal Modifications= 0

 Change Profile Region = 0



#

# Modtran Cards (modcard.cxx)

#

$ Modtran: Card 1.

 Model Type             = 7

 Type of Path           = 2

 Execution Mode         = 1

 Type of Scattering     = 0

 Default T/P Model      = 0

 Default H2O Model      = 0

 Default O3 Model       = 0

 Default CH4 Model      = 0

 Default N2O Model      = 0

 Default CO Model       = 0

 Default Trace Gas Mdl  = 0

 Data Loading Method    = 1

 Report Generation Type= 0

 Background Temperature= 293.150

 Background Albedo      = 0.1


$ Modtran: Card 2.

 Aerosol Attenuation    = 1

 Season Aerosol Profile= 0

 Stratospheric Profile  = 1

 Mass character-Navy    = 3

 Cloud Model Type       = 0

 Army Vert. Struct. Alg= 0

 Surface visibility km  = 0.0

 Wind Speed-Navy/Desert= 0.0

 24 hr Ave Wind Speed   = 0.0

 Rain Rate (mm/hr)      = 0.0

 Surface Altitude       = 0.0
```

```
$ Modtran: Card 2c.
 Number of Atm Layers  = 4
 User Defined Gas Dens.= 1
 User Defined Aerosols = 0
 Layer Top Boundry Alt.= 0.0
 Layer Pressure/Unit   = 7.60e+02, C
 Layer Temperature/Unit= 2.931e+02, A
 H20 Density/Unit      = 0.0, 6
 C02 Density/Unit      = 0.0, 6
 03 Density/Unit       = 0.0, 6
 N20 Density/Unit      = 0.0, 6
 CO Density/Unit       = 0.0, 6
 CH4 Density/Unit      = 0.0, 6
 02 Density/Unit       = 0.0, 6
 NO Density/Unit       = 0.0, 6
 S02 Density/Unit      = 0.0, 6
 N02 Density/Unit      = 0.0, 6
 NH3 Density/Unit      = 0.0, 6
 HN03 Density/Unit     = 0.0, 6
 Layer Top Boundry Alt.= 5.0e-3
 Layer Pressure/Unit   = 7.61e+02, C
 Layer Temperature/Unit= 2.931e+02, A
 H20 Density/Unit      = 0.0, 6
 C02 Density/Unit      = 0.0, 6
 03 Density/Unit       = 0.0, 6
 N20 Density/Unit      = 0.0, 6
 CO Density/Unit       = 0.0, 6
 CH4 Density/Unit      = 0.0, 6
 02 Density/Unit       = 0.0, 6
 NO Density/Unit       = 0.0, 6
 S02 Density/Unit      = 0.0, 6
 N02 Density/Unit      = 0.0, 6
 NH3 Density/Unit      = 0.0, 6
 HN03 Density/Unit     = 0.0, 6
 Layer Top Boundry Alt.= 6.0e-3
 Layer Pressure/Unit   = 7.60e+02, C
 Layer Temperature/Unit= 2.931e+02, A
 H2O Density/Unit      = 0.0, 6
 CO2 Density/Unit      = 0.0, 6
 O3 Density/Unit       = 0.0, 6
 N2O Density/Unit      = 0.0, 6
 CO Density/Unit       = 0.0, 6
 CH4 Density/Unit      = 0.0, 6
 O2 Density/Unit       = 0.0, 6
 NO Density/Unit       = 0.0, 6
 SO2 Density/Unit      = 0.0, 6
 NO2 Density/Unit      = 0.0, 6
 NH3 Density/Unit      = 0.0, 6
 HNO3 Density/Unit     = 0.0, 6
 Layer Top Boundry Alt.= 5.0e-2
 Layer Pressure/Unit   = 0.0, 6
 Layer Temperature/Unit= 0.0, 6
 H2O Density/Unit      = 0.0, 6
 CO2 Density/Unit      = 0.0, 6
 O3 Density/Unit       = 0.0, 6
 N2O Density/Unit      = 0.0, 6
 CO Density/Unit       = 0.0, 6
 CH4 Density/Unit      = 0.0, 6
 O2 Density/Unit       = 0.0, 6
 NO Density/Unit       = 0.0, 6
 SO2 Density/Unit      = 0.0, 6
 NO2 Density/Unit      = 0.0, 6
 NH3 Density/Unit      = 0.0, 6
 HNO3 Density/Unit     = 0.0, 6

$ Modtran: Card 3.
 Initial Altitude      = 1.0e-2
 Final Altitude        = 0.0
 Initial Zenith Angle  = 179.0
 Path Length           = 0.0
 Earth Center Angle    = 0.0
 Radius of Earth       = 0.0
 Type of Path          = 1

$ Modtran: Card 4.
 Initial Frequency WN  = 2000
 Final Frequency WN    = 2300
 Frequency Interval WN = 2
 Fullwidth at Half Max = 2

#
# Geom Values (geom.cxx)
#
$ Geom Class Parameters
 Box Type Index        = 0
 Ellipse Type Index    = 1
 Cone Type Index       = 2
 Cylinder Type Index   = 3



#
# Box Object Parameters (box.cxx)
#
$ Geom Model: Box DB
 Number of Box Models  = 0



#
# Cone Object Parameters (cone.cxx)
#
$ Geom Model: Cone DB
 Number of Cone Models = 1
 Cone Model Name       = Cone_1
 Cone Model Index      = 1
 Height along Z axis   = 0.4
 Start. Angle of 1st Pt= 0.0
 Radius of Base        = .075
 Number of Pnts in Base= 16
 X Offset of Top       = 0.0
 Y Offset of Top       = 0.0

#
# Cylinder Object Parameters (cylinder,cxx)
#
$ Geom Model: Cylinder DB
 Num of Cylinder Models= 0

#
# Ellipsoid Object Parameters (ellipse.cxx)
#
$ Geom Model: Ellipse DB
 Num. of Ellipse Models= 0



#
# Load Animation Parameters (keyfrm.cxx)
#
# Animation Parameters for Each Object
#
# Integer Data Value #1 = Gas Model
# Integer Data Value #2 = Aerosol Model
#
$ Key Frame Param: Cone_1
 Number of Key Frames   = 2
 Interpolation Method   = 0
 Frame Number           = 1
 Object Active          = 1
 Origin Location (WCS) = 0.268, -0.005, 2.5
 Orient. about Origin   = 0.0, 90.0, 0.0
 Scale about Origin     = 0.75, 0.75, 1.0
 # of Interp. Coeff.    = 0
 # of Integer Data Pts = 2
 Integer Data Value #01  = 2
 Integer Data Value #02  = 1
 # of FP Data Pts       = 0
 Frame Number           = 90
 Object Active          = 1
 Origin Location (WCS) = .149, -0.005, 2.5
 Orient. about Origin  = 0.0, 90.0, 0.0
 Scale about Origin    = 1.0, 1.0, 1.0
 # of Interp. Coeff.   = 0
 # of Integer Data Pts = 2
 Integer Data Value #01  = 2
 Integer Data Value #02  = 1
 # of FP Data Pts       = 0



#
# Load Layout Parameters (keyfrm.cxx)
#
# Layout DB Loads Object Index and Type in Order of Layout
#
$ Layout DB Param: Geometry Object Layout
 Number of Layout Obj  = 1
 Layout Key Frame Num. = 1
 Num of Objects in Fram= 1
 Object at Pos #0001   = 1
 Object Type Pos #0001 = 2
```

C.4 The FPA Object Parameter File

```
#
# Focal Plane Array Object (FPA) Parameter File
#
# Simulation Name: Car
# Created on Date: 9/3/97
#


#
# FPA Parameters (fpa.cxx)
#
# Load specified index number for FPA
$ FPA Parameters # 1
 WCS Center Pnt of FPA = 0.0, 0.0, -0.17
 WCS Orientation of FPA= 0.0, 0.0, 0.0
 Detector Element Pitch= 3.8e-5, 3.8e-5
 Num Of Elements (X,Y) = 256, 256
 Ray Distrib. Method   = 1
 # of Mesh Points (X,Y)= 1, 1
 Optics Object Index   = 1
 Output Object Index   = 1
 Unit Cell:# of Det(X,Y)= 1, 1
 Cell Loc. in Unit Cell= 0, 0
 # of Detectors in Cell= 1
 Pct Size of Element   = 100.0, 100.0
 Detector ID Index     = 1
 Min Wavenumber to Calc= 2040
 # of Lines to Calculat= 100
 Bandwidth (wavenumber)= 2
 Full Width - Half Max = 2
```

C.5 The Detector Object Parameter File

```
#
# Detector Database (DET) Parameter File
#
# Simulation Name: Car
# Created on Date: 9/3/97
#


#
# General Detector Parameters (detector.cxx)
#
$ Detector Object
 Number of PV Detect   = 1
 Number of PC Detect   = 0
 Number of Pyro Detect = 0
 Number of Bolo Detect = 0
 Detector Type    = 0
 Detector ID Index= 1



#
# Detector Type Information (detector.cxx)
#
# Each detector has this type of loading parameters
#
# (Integration Time = (1/Frame Rate)*(Amber Int Time/128)
#
$ Param-Detector #  1
 Det. Cold Shield FOV  = 22.2
 Fill Factor           = 0.954
 Quantum Efficiency    = 0.65
 Load Resistance       = 1.66e9
 Integration Time      = 0.005208
 Detector Temperature  = 77.0
 FPA Output Units      = 0
 Noise Prob Function   = 0
 Sigma Width           = 2.0
 Location of curve     = 0
 Curve Index           = 1



#
# Photovoltaic Detector Specific Parameters (PV) (detector.cxx)
#
$ PV Detector Param #  1



#
# Responsivity/D* Curves of Detector(s) (detector.cxx)
#
$ Detector Response Param #  1
 Type of Curve         = 0
 Output Signal Units   = 0
 Input Power Units     = 4
 Curve Spectral Units  = 0
 Measured Noise Voltage= 0.624e-3
 Meas. Cold Shield FOV = 22.2
 Measured Noise Bandwth= 548.57
 Measured Int. Time    = .000911
 # of Points on Curve  = 4
 Spectral Unit         = 1818.0
 Responsivity/D* Value = 1.0e-20
 Spectral Unit         = 1850.0
 Responsivity/D* Value = 1.90e-7
 Spectral Unit         = 2000.0
 Responsivity/D* Value = 1.90e-7
 Spectral Unit         = 3400.0
 Responsivity/D* Value = 1.90e-7
```

C.6 The Optics Object Parameter File

```
#
# Optics Object (OPT) Parameter File
#
# Simulation Name: Car
# Created on Date: 9/3/97
#


#
# Optics Parameters (optics.cxx)
#
# Load Specific Optical Index (id) Right Justify
$ Optical Param # 1
 Parameter Loading Meth= 0
 Front Origin (WCS)     = 0.0, 0.0, 0.0
 Front OA Vector (WCS) = 0.0, 0.0, -1.0
 1st Focal Point(WCS)  = 0.0, 0.0, 0.07
 1st Principal Pnt(WCS)= 0.0, 0.0, -0.03
 1st Nodal Point(WCS)  = 0.0, 0.0, -0.03
 Back Origin (WCS)     = 0.0, 0.0, -0.1
 Back OA Vector (WCS)  = 0.0, 0.0, -1.0
 2nd Focal Point(WCS)  = 0.0, 0.0, -0.17
 2nd Principal Pnt(WCS)= 0.0, 0.0, -0.07
 2nd Nodal Point(WCS)  = 0.0, 0.0, -0.07
 Nodal Phase Angle     = 0.0
 Effective F Number    = 2.5489
 Transmission Coeff Loc= 2
 Number of Trans Curves= 2
 Transmission Coeff Ind= 1
 Transmission Coeff Ind= 14


#
# Transmission Coefficients for Compound Optics (optics.cxx)
#
```

```
#
# CaF Window
#
$ Transmission Coeff # 1
 # of Transmission Pnts= 12
 Frequency (wavenumber)= 1800
 Transmission Coeff.   = 0.935
 Frequency (wavenumber)= 2000
 Transmission Coeff.   = 0.935
 Frequency (wavenumber)= 2083
 Transmission Coeff.   = 0.930
 Frequency (wavenumber)= 2174
 Transmission Coeff.   = 0.920
 Frequency (wavenumber)= 2273
 Transmission Coeff.   = 0.920
 Frequency (wavenumber)= 2381
 Transmission Coeff.   = 0.925
 Frequency (wavenumber)= 2500
 Transmission Coeff.   = 0.930
 Frequency (wavenumber)= 2632
 Transmission Coeff.   = 0.945
 Frequency (wavenumber)= 2778
 Transmission Coeff.   = 0.930
 Frequency (wavenumber)= 2941
 Transmission Coeff.   = 0.910
 Frequency (wavenumber)= 3125
 Transmission Coeff.   = 0.910
 Frequency (wavenumber)= 3333
 Transmission Coeff.   = 0.920


#
# Filter - N03288-8 (Cen = 3.288, FWHM=.070)
#
$ Transmission Coeff # 2
```

```
# of Transmission Pnts= 17

Frequency (wavenumber)= 2000

Transmission Coeff.   = 0.0

Frequency (wavenumber)= 2973

Transmission Coeff.   = 0.0

Frequency (wavenumber)= 2993

Transmission Coeff.   = 0.050

Frequency (wavenumber)= 2999

Transmission Coeff.   = 0.100

Frequency (wavenumber)= 3005

Transmission Coeff.   = 0.200

Frequency (wavenumber)= 3014

Transmission Coeff.   = 0.700

Frequency (wavenumber)= 3028

Transmission Coeff.   = 0.775

Frequency (wavenumber)= 3044

Transmission Coeff.   = 0.730

Frequency (wavenumber)= 3053

Transmission Coeff.   = 0.680

Frequency (wavenumber)= 3065

Transmission Coeff.   = 0.710

Frequency (wavenumber)= 3067

Transmission Coeff.   = 0.700

Frequency (wavenumber)= 3071

Transmission Coeff.   = 0.600

Frequency (wavenumber)= 3082

Transmission Coeff.   = 0.200

Frequency (wavenumber)= 3086

Transmission Coeff.   = 0.100

Frequency (wavenumber)= 3091

Transmission Coeff.   = 0.050

Frequency (wavenumber)= 3113

Transmission Coeff.   = 0.0

Frequency (wavenumber)= 3333

Transmission Coeff.   = 0.0
```

```
#
# Filter - N03306-6 (Cen = 3.3306, FWHM=.36)
#
$ Transmission Coeff # 3
 # of Transmission Pnts= 19
 Frequency (wavenumber)= 2000
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2833
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2861
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2886
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2899
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2915
 Transmission Coeff.   = 0.850
 Frequency (wavenumber)= 2928
 Transmission Coeff.   = 0.901
 Frequency (wavenumber)= 2941
 Transmission Coeff.   = 0.893
 Frequency (wavenumber)= 2967
 Transmission Coeff.   = 0.901
 Frequency (wavenumber)= 3030
 Transmission Coeff.   = 0.885
 Frequency (wavenumber)= 3110
 Transmission Coeff.   = 0.915
 Frequency (wavenumber)= 3210
 Transmission Coeff.   = 0.890
 Frequency (wavenumber)= 3231
 Transmission Coeff.   = 0.850
 Frequency (wavenumber)= 3263
 Transmission Coeff.   = 0.200
 Frequency (wavenumber)= 3279
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 3295
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 3311
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 3344
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 4000
 Transmission Coeff.   = 0.0


#
# Filter - N03322-8 (Cen = 3.322, FWHM=.071)
#
$ Transmission Coeff # 4
 # of Transmission Pnts= 18
 Frequency (wavenumber)= 2000
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2949
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2965
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2970
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2974
 Transmission Coeff.   = 0.200
 Frequency (wavenumber)= 2990
 Transmission Coeff.   = 0.800
 Frequency (wavenumber)= 2999
 Transmission Coeff.   = 0.862
 Frequency (wavenumber)= 3013
 Transmission Coeff.   = 0.862
 Frequency (wavenumber)= 3018
 Transmission Coeff.   = 0.825
 Frequency (wavenumber)= 3028
 Transmission Coeff.   = 0.790
 Frequency (wavenumber)= 3033
 Transmission Coeff.   = 0.795
 Frequency (wavenumber)= 3040
 Transmission Coeff.   = 0.740
 Frequency (wavenumber)= 3053
 Transmission Coeff.   = 0.200
 Frequency (wavenumber)= 3058
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 3064
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 3071
 Transmission Coeff.   = 0.020
 Frequency (wavenumber)= 3085
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 3333
 Transmission Coeff.   = 0.0


#
# Filter - N03411-6  (Cen = 3.411 , FWHM = .046 )
#
$ Transmission Coeff # 5
 # of Transmission Pnts=12
 Frequency (wavenumber)= 2000
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2880
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2896
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2903
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2923
 Transmission Coeff.   = 0.750
 Frequency (wavenumber)= 2929
 Transmission Coeff.   = 0.788
 Frequency (wavenumber)= 2942
 Transmission Coeff.   = 0.750
 Frequency (wavenumber)= 2949
 Transmission Coeff.   = 0.700
 Frequency (wavenumber)= 2966
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2972
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2985
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 3333
 Transmission Coeff.   = 0.0


#
# Filter - N03689-4  (Cen = 3.689 , FWHM = .046 )
#
$ Transmission Coeff # 6
 # of Transmission Pnts= 18
 Frequency (wavenumber)= 2000
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2663
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2667
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2670
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2676
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2682
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2687
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2692
 Transmission Coeff.   = 0.200
 Frequency (wavenumber)= 2703
 Transmission Coeff.   = 0.665
 Frequency (wavenumber)= 2718
 Transmission Coeff.   = 0.727
 Frequency (wavenumber)= 2721
 Transmission Coeff.   = 0.700
 Frequency (wavenumber)= 2726
 Transmission Coeff.   = 0.600
 Frequency (wavenumber)= 2736
 Transmission Coeff.   = 0.200
 Frequency (wavenumber)= 2741
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2745
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2752
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2762
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 3333
 Transmission Coeff.   = 0.0


#
# Filter - N03888-4 (Cen = 3.888, FWHM = .047)
#
$ Transmission Coeff # 7
 # of Transmission Pnts= 15
 Frequency (wavenumber)= 2000
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2532
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2539
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2545
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2550
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2556
 Transmission Coeff.   = 0.200
 Frequency (wavenumber)= 2576
 Transmission Coeff.   = 0.785
 Frequency (wavenumber)= 2579
 Transmission Coeff.   = 0.796
 Frequency (wavenumber)= 2587
 Transmission Coeff.   = 0.760
 Frequency (wavenumber)= 2597
 Transmission Coeff.   = 0.200
 Frequency (wavenumber)= 2602
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2605
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2612
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2620
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 3333
 Transmission Coeff.   = 0.0


#
# Filter - N03990-4  (Cen = 3.990, FWHM = .190)
#
$ Transmission Coeff # 8
 # of Transmission Pnts= 17
 Frequency (wavenumber)= 2000
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2381
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2433
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2451
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2463
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2503
 Transmission Coeff.   = 0.800
 Frequency (wavenumber)= 2509
 Transmission Coeff.   = 0.872
 Frequency (wavenumber)= 2522
 Transmission Coeff.   = 0.898
 Frequency (wavenumber)= 2545
 Transmission Coeff.   = 0.870
 Frequency (wavenumber)= 2567
 Transmission Coeff.   = 0.885
 Frequency (wavenumber)= 2581
 Transmission Coeff.   = 0.870
 Frequency (wavenumber)= 2594
 Transmission Coeff.   = 0.800
 Frequency (wavenumber)= 2635
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2653
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2667
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2740
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 3333
 Transmission Coeff.   = 0.0


#
# Filter - N04227-4  (Cen = 4.227, FWHM = .055)
#
$ Transmission Coeff # 9
 # of Transmission Pnts= 15
 Frequency (wavenumber)= 2000
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2326
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2335
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2339
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2345
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2360
 Transmission Coeff.   = 0.700
 Frequency (wavenumber)= 2369
 Transmission Coeff.   = 0.750
 Frequency (wavenumber)= 2372
 Transmission Coeff.   = 0.754
 Frequency (wavenumber)= 2375
 Transmission Coeff.   = 0.750
 Frequency (wavenumber)= 2378
 Transmission Coeff.   = 0.700
 Frequency (wavenumber)= 2392
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2396
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2404
 Transmission Coeff.   = 0.020
 Frequency (wavenumber)= 2410
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 3333
 Transmission Coeff.   = 0.0


#
# Filter - N04235-4 (Cen = 4.235, FWHM = .181)
#
$ Transmission Coeff #10
 # of Transmission Pnts= 29
 Frequency (wavenumber)= 2000
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2283
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2288
```

```
Transmission Coeff.    = 0.010
Frequency (wavenumber)= 2294
Transmission Coeff.    = 0.020
Frequency (wavenumber)= 2296
Transmission Coeff.    = 0.040
Frequency (wavenumber)= 2299
Transmission Coeff.    = 0.080
Frequency (wavenumber)= 2304
Transmission Coeff.    = 0.200
Frequency (wavenumber)= 2320
Transmission Coeff.    = 0.770
Frequency (wavenumber)= 2326
Transmission Coeff.    = 0.840
Frequency (wavenumber)= 2336
Transmission Coeff.    = 0.870
Frequency (wavenumber)= 2342
Transmission Coeff.    = 0.860
Frequency (wavenumber)= 2347
Transmission Coeff.    = 0.830
Frequency (wavenumber)= 2353
Transmission Coeff.    = 0.810
Frequency (wavenumber)= 2364
Transmission Coeff.    = 0.840
Frequency (wavenumber)= 2370
Transmission Coeff.    = 0.850
Frequency (wavenumber)= 2375
Transmission Coeff.    = 0.820
Frequency (wavenumber)= 2387
Transmission Coeff.    = 0.670
Frequency (wavenumber)= 2389
Transmission Coeff.    = 0.650
Frequency (wavenumber)= 2392
Transmission Coeff.    = 0.660
Frequency (wavenumber)= 2398
Transmission Coeff.    = 0.740
Frequency (wavenumber)= 2401
```

```
 Transmission Coeff.   = 0.774
 Frequency (wavenumber)= 2404
 Transmission Coeff.   = 0.760
 Frequency (wavenumber)= 2415
 Transmission Coeff.   = 0.250
 Frequency (wavenumber)= 2421
 Transmission Coeff.   = 0.060
 Frequency (wavenumber)= 2427
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2433
 Transmission Coeff.   = 0.010
 Frequency (wavenumber)= 2439
 Transmission Coeff.   = 0.004
 Frequency (wavenumber)= 2451
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 3333
 Transmission Coeff.   = 0.0


#
# Filter - N04461-8 (Cen = 4.461, FWHM = .052)
#
$ Transmission Coeff #11
 # of Transmission Pnts= 21
 Frequency (wavenumber)= 2000
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2193
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2200
 Transmission Coeff.   = 0.010
 Frequency (wavenumber)= 2212
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2217
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2221
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2225
 Transmission Coeff.   = 0.200
 Frequency (wavenumber)= 2228
 Transmission Coeff.   = 0.300
 Frequency (wavenumber)= 2235
 Transmission Coeff.   = 0.800
 Frequency (wavenumber)= 2237
 Transmission Coeff.   = 0.830
 Frequency (wavenumber)= 2239
 Transmission Coeff.   = 0.857
 Frequency (wavenumber)= 2247
 Transmission Coeff.   = 0.830
 Frequency (wavenumber)= 2250
 Transmission Coeff.   = 0.800
 Frequency (wavenumber)= 2260
 Transmission Coeff.   = 0.300
 Frequency (wavenumber)= 2262
 Transmission Coeff.   = 0.200
 Frequency (wavenumber)= 2263
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2270
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2275
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2283
 Transmission Coeff.   = 0.010
 Frequency (wavenumber)= 2294
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 3333
 Transmission Coeff.   = 0.0


#
# Filter - N04577-4 (Cen = 4.577, FWHM = .260)
#
$ Transmission Coeff #12
```

```
# of Transmission Pnts= 17
Frequency (wavenumber)= 2000
Transmission Coeff.   = 0.0
Frequency (wavenumber)= 2128
Transmission Coeff.   = 0.0
Frequency (wavenumber)= 2167
Transmission Coeff.   = 0.025
Frequency (wavenumber)= 2176
Transmission Coeff.   = 0.050
Frequency (wavenumber)= 2181
Transmission Coeff.   = 0.100
Frequency (wavenumber)= 2195
Transmission Coeff.   = 0.595
Frequency (wavenumber)= 2210
Transmission Coeff.   = 0.795
Frequency (wavenumber)= 2220
Transmission Coeff.   = 0.815
Frequency (wavenumber)= 2227
Transmission Coeff.   = 0.795
Frequency (wavenumber)= 2250
Transmission Coeff.   = 0.595
Frequency (wavenumber)= 2270
Transmission Coeff.   = 0.795
Frequency (wavenumber)= 2275
Transmission Coeff.   = 0.595
Frequency (wavenumber)= 2288
Transmission Coeff.   = 0.100
Frequency (wavenumber)= 2291
Transmission Coeff.   = 0.050
Frequency (wavenumber)= 2298
Transmission Coeff.   = 0.025
Frequency (wavenumber)= 2326
Transmission Coeff.   = 0.0
Frequency (wavenumber)= 3333
Transmission Coeff.   = 0.0
```

```
#
# Filter - W04684-4 (Cen = 4.684, FWHM = .600 )
#
$ Transmission Coeff #13
 # of Transmission Pnts= 17
 Frequency (wavenumber)= 1800
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 1951
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 1992
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2010
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2058
 Transmission Coeff.   = 0.800
 Frequency (wavenumber)= 2073
 Transmission Coeff.   = 0.850
 Frequency (wavenumber)= 2083
 Transmission Coeff.   = 0.840
 Frequency (wavenumber)= 2141
 Transmission Coeff.   = 0.875
 Frequency (wavenumber)= 2183
 Transmission Coeff.   = 0.900
 Frequency (wavenumber)= 2227
 Transmission Coeff.   = 0.9222
 Frequency (wavenumber)= 2270
 Transmission Coeff.   = 0.900
 Frequency (wavenumber)= 2281
 Transmission Coeff.   = 0.800
 Frequency (wavenumber)= 2370
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2378
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2392
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2439
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 3333
 Transmission Coeff.   = 0.0


#
# Filter - N04693-8 (Cen = 4.693, FWHM = .167)
#
$ Transmission Coeff #14
 # of Transmission Pnts= 23
 Frequency (wavenumber)= 2000
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2054
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 2059
 Transmission Coeff.   = 0.010
 Frequency (wavenumber)= 2071
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2076
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2084
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2089
 Transmission Coeff.   = 0.200
 Frequency (wavenumber)= 2104
 Transmission Coeff.   = 0.700
 Frequency (wavenumber)= 2107
 Transmission Coeff.   = 0.750
 Frequency (wavenumber)= 2112
 Transmission Coeff.   = 0.780
 Frequency (wavenumber)= 2116
 Transmission Coeff.   = 0.770
 Frequency (wavenumber)= 2125
 Transmission Coeff.   = 0.785
 Frequency (wavenumber)= 2133
 Transmission Coeff.   = 0.775
 Frequency (wavenumber)= 2139
 Transmission Coeff.   = 0.730
 Frequency (wavenumber)= 2145
 Transmission Coeff.   = 0.720
 Frequency (wavenumber)= 2156
 Transmission Coeff.   = 0.750
 Frequency (wavenumber)= 2162
 Transmission Coeff.   = 0.725
 Frequency (wavenumber)= 2177
 Transmission Coeff.   = 0.200
 Frequency (wavenumber)= 2182
 Transmission Coeff.   = 0.100
 Frequency (wavenumber)= 2184
 Transmission Coeff.   = 0.050
 Frequency (wavenumber)= 2191
 Transmission Coeff.   = 0.025
 Frequency (wavenumber)= 2203
 Transmission Coeff.   = 0.0
 Frequency (wavenumber)= 3333
 Transmission Coeff.   = 0.0


#
# Filter -  (Cen = , FWHM = )
#
$ Transmission Coeff #
 # of Transmission Pnts=
 Frequency (wavenumber)=
 Transmission Coeff.   =
```

C.7 The Output Object Parameter File

```
#
# Output Object (OUT) Parameter File
#
# Simulation Name: Car
# Created on Date: 9/3/97
#

#
# Output Parameters (General Information) (output.cxx)
#
# Load General Output Parameters for Object labeled by id.
$ Output Parameters # 1
 Coordinate Output Type= 1
 Output File Index     = 1


#
# Output File Parameters
#
# Load Specific Output file PArameters (based on index)
# Most answers are TRUE-1, FALSE-0
$ Output File Param # 1
 Output Filename Head   = car_sim/data/car_move
 Output Detector Type   = 0
 Output Coordinates     = 1
 Output Incident Power = 1
 Output Photon Rate     = 1
 Pwr/Pht Rate in Log10 = 1
 Output Detector Signal= 1
 Output Detector Noise = 1
 Output Detector Total = 1
 Det. Output in Log10   = 0
```