2000

An interactive simulation environment for end-to-end digital imaging system design and fidelity analysis

Moira Joyce Turner
College of William & Mary - Arts & Sciences

Recommended Citation

Turner, Moira Joyce, "An interactive simulation environment for end-to-end digital imaging system design and fidelity analysis" (2000). *Dissertations, Theses, and Masters Projects.* William & Mary. Paper 1539623980.
https://dx.doi.org/doi:10.21220/s2-gj98-j145

AN INTERACTIVE SIMULATION ENVIRONMENT FOR END-TO-END DIGITAL IMAGING SYSTEM DESIGN AND FIDELITY ANALYSIS

A Dissertation

Presented to

The Faculty of the Department of Computer Science

The College of William and Mary in Virginia

In Partial Fulfillment

Of the Requirements for the Degree of

Doctor of Philosophy

by

Moira Joyce Turner

2000

UMI Number: 9982022

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

APPROVAL SHEET

This dissertation is submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

Moira J. Turner

Moira J. Turner

Approved, May 2000

Stephen K. Park

Stephen K. Park
Thesis Advisor

William L. Bynum

William L. Bynum

Richard H. Prosl

Richard H. Prosl

Zia-ur Rahman

Lawrence Leemis

Lawrence M. Leemis
Department of Mathematics

To

Sarah McIntyre Young Gibb

(b. 16 October, 1920, d. 12 July, 1987)

&

James Randall Duffy

(b. 25 July, 1915, d. 3 January, 1958)

♡♡

I wish they could have known ...

Table of Contents

ACKNOWLEDGMENTS

This research was made possible by the encouragement and support — both moral and financial — of my husband, John. It has also required sacrifice — sometimes willing, sometimes not — on the part of my children, Allison and Jonathan. I hope my family can forgive me for the times I wasn't there, whether it was the body or the soul that was absent.

This work would also have been impossible without the academic support of my mentor and advisor, Dr. Stephen K. Park, whose help continued to be forthcoming throughout times of extreme difficulty in his own personal life.

List of Tables

List of Figures

ABSTRACT

The detailed specification, implementation, and documentation of an interactive software environment based on a continuous/discrete/continuous imaging system model is presented. The purpose of the interactive environment is to support the design and performance analysis of end-to-end digital imaging systems. Development of the environment is based on the objectives of acceptable response time, large sampling grid capability, good graphical user interface design, independence from proprietary applications and portability among UNIX workstations. While one-dimensional variations of interactive design environments have been developed by the commercial active filter design community, there is little or no evidence that the increased complexity associated with the extension to two dimensions had been satisfactorily accomplished prior to the work in this dissertation. The computer time versus computer memory trade-off is discussed as it applies in this particular context. and the results of a systematic study of representation passband limits are presented. The object of the study was to determine the representation passband parameters beyond which any aliasing contribution from frequencies beyond the representation passband is invariably negligible. Validation of the environment is documented by an exhaustive consideration of simple input scenes comprised of a uniform square on a uniform background, in which the square can be arbitrarily small and arbitrarily located within the scene. The effects of sampling and the dependence of those effects on sample-scene phase are illustrated in 1-D, used as a predictor for the 2-D outcome, and then illustrated in 2-D for the purpose of comparing the projected and actual results.

AN INTERACTIVE SIMULATION ENVIRONMENT FOR END-TO-END DIGITAL IMAGING SYSTEM DESIGN AND FIDELITY ANALYSIS

Chapter 1

Introduction

1.1 Preface

This dissertation chronicles the development of an interactive simulation environment for the design and performance analysis of end-to-end digital imaging systems. Specifically. research focused on the detailed specification, implementation, and documentation of such an environment that would provide at least the following capabilities.

- Selecting an input scene from the environment's library.
- Displaying images in either the spatial domain or the frequency domain.
- Visualizing the input-output matrix representations[1] of the input scene as it is processed through each component of the end-to-end system (*qualitative* analysis).
- Analyzing performance based on the evaluation of fidelity metrics (*quantitative* analysis).

[1] "Matrix representation" is the term used in this dissertation to describe the generic 2-D data structures that represent the input scene as it is processed through each component of the end-to-end system.

- Interactively modifying the system filters.
- Interactively modifying the 2-D (two-dimensional) sampling density.
- Simulating image acquisition noise.
- Isolating and visualizing thc effccts of aliasing due to spatial sampling.
- Operating in a 1-D (one-dimensional) mode.

Development of the environment[2] was originally motivated by concurrent interests in the areas of graphical user interface design and digital image processing. A vehiclc for combining the two areas into a unified field of research was actively sought and identified as the result of a request to create a framework within which students could develop 1-D system models in a Discrete Linear Systems class offered by the Department of Computer Science. Generation of a graphical user interface to handle a 1-D problem paved the way for the extension to the much harder 2-D problem. A search for similar environments unearthed several, both 1-D and 2-D. Some of these, in particular the 1-D variety, exhibit sophisticated features, but research confirmed that none are based on the comprehensive 2-D continuous-input / discrete-processing / continuous-output (c/d/c) system model described in Chapter 2 [24]. The development of an environment which would support end-to-end digital imaging system design and performance analysis utilizing the c/d/c system model was established as the primary goal of the dissertation research.

[2] "Environment" is the term used consistently in the remainder of this dissertation to denote the suite of interactive software developed during the course of this research. The suite of programs facilitates the design and performance analysis of end-to-end digital imaging systems, utilizing a front-end graphical user interface.

In developing the specification for and the implementation of an environment based on the c/d/c system model, the objectives were:

- support for end-to-end digital imaging system design and performance analysis;
- acceptable response time;
- sampling grids at least as large as 512 × 512;
- good Graphical User Interface (GUI) design;
- independence from proprietary applications;
- portability among the three UNIX/Linux systems available for test purposes.

The environment was developed in ANSI C as an X Window System application for UNIX platforms utilizing version X11 Release 4 or later. The implementation incorporates raw X commands [13, 14] with no other software restrictions on the operation of the environment; it runs without software modification on Linux, Irix and Sun operating systems.

1.2 Introduction

As stated in Section 1.1, the environment is based on the c/d/c system model, which is described in Chapter 2. It has been demonstrated that this model is more comprehensive than other common, but incomplete, models [22, 24] and so is well suited for simulating end-to-end digital imaging systems. The environment based on the c/d/c system model facilitates imaging system design as well as both *qualitative* (visual) and *quantitative* (numerical) evaluation of imaging system performance.

1.2.1 Acceptable response times

The facility to visualize the matrix representation of the input scene at various processing stages within the c/d/c system model provides insight into the extent to which each system component contributes to effects and artifacts observed in the output image. However, if the matrix representation could not be displayed interactively within a reasonable period of time, the environment would not be very useful. One objective of this research, therefore, was to provide an environment with acceptable response times which would facilitate modification of the c/d/c system model parameters and allow convenient display of the matrix representation of the input scene at the input and output of each component of the c/d/c system model.

Two computational techniques that are naturally part of the c/d/c system model (depending on implementation details) are CPU-time intensive. These are the calculation of direct and inverse 2-D Fourier transforms and 2-D spatial domain convolutions. The use of these two computational techniques was considered carefully, therefore, to minimize thc response time experienced by the user. As a result, input scene data is *pre-processed* to ensure that Fourier representations (i.e., frequency domain representations) are the standard format for all the data supplied to the environment, and subsequent processing occurs exclusively in the frequency domain. This ensures that spatial domain convolutions can be implemented as multiplications in the frequency domain on a frequency-by-frequency basis and eliminates any delay which would otherwise result from transformation of the input scene to the frequency domain.

1.2.2 Two distinct modes of operation

Also in the interests of minimizing response time, although ultimately rejected, the operation of the environment in two distinct modes was investigated.

- The *process* mode was a 2-D mode in which the 2-D matrix representation of the input scene was processed through the c/d/c system model, allowing the user to visualize the matrix representation at the input and output of each c/d/c system model component and to display 2-D fidelity metrics.

- The *design* mode was a 1-D mode in which a 1-D vector representation of a typical row or column from a 2-D matrix representation was processed through the c/d/c system model, allowing the user to view a representation of the 1-D vector at the input and output of each c/d/c system model component and to display 1-D fidelity metrics.

In *design* mode only, the user had the capability to redesign the digital imaging system by modifying the c/d/c system model parameters. The purpose of restricting processing to 1-D in this case was to allow the user to view the characteristic effects of parameter changes without having to wait for 2-D matrix representation processing and display each time a parameter is modified. While the environment remained in *design* mode, changes to the system parameters did not affect the 2-D matrix representations. When *design* mode was exited, the environment reverted to *process* mode and the effects of the altered system parameters became visible as the 2-D input scene was processed through the c/d/c system model.

In practice, this two-mode implementation proved cumbersome and frustrating to the user and so was rejected. A redesign adapted the philosophy of this approach by replacing the *process* and *design* modes with 2-D and 1-D modes. This change added the design capabilities of 1-D mode to 2-D mode, thus allowing the user to trade off convenience and patience versus inconvenience and processing speed, as the merits of a particular scenario demand. The 1-D mode, of course, is a useful tool in its own right. It is not simply a means of speeding up response time when making filter or system parameter changes.

1.2.3 Memory requirements

In addition to response time, the amount of memory required by the environment was carefully considered. To speed processing time, it would be desirable to perform all calculations (including transformations to the spatial domain) once and store the results. Subsequently, any matrix representation in either domain would be instantaneously accessible. However, this would be feasible only when the sampling grid is small, and so the following steps are taken to minimize memory requirements.

- Possible values of the user-selected parameters, sampling grid size ($N_1 \times N_2$) and representation passbands (τ_1, τ_2), are restricted, thus imposing an upper limit on the size of the stored data structures.

- As input scenes are *pre-processed* for inclusion in the environment's library, scenes which exceed 8 times the maximum sampling grid size (1024×1024) in either dimension are cropped to a maximum of 8192×8192.

- Only frequency domain matrix representations are stored.

- Two-quadrant complex conjugacy is utilized, making it necessary to store only half of each frequency domain matrix representation.

- Display size is never allowed to exceed 512 × 512.

The environment can utilize a sampling grid size of 512 × 512 on contemporary systems.

1.3 C/D/C System Model Background Literature

The c/d/c system model described in Chapter 2 is the basis for the environment. In this section, sources for the c/d/c system model and related papers are listed. In each case. a brief description of the paper's content is included. Wherever appropriate, the description is a quote from the author(s).

Fidelity Analysis of Sampled Imaging Systems, S. K. Park and Z. Rahman (1999) [24]

"For those sampled imaging systems where the effects of digital image acquisition. digital filtering, and image reconstruction are significant, the modeling, simulation and performance analysis should be based on a ... comprehensive continuous-input, discrete-processing, continuous-output end-to-end model. This ... comprehensive model should properly account for the low-pass filtering effects of image acquisition prior to sampling. the potentially important noiselike effects of the aliasing caused by sampling, additive noise due to device electronics and quantization, the generally high-boost filtering effects of digital processing, and the low-pass filtering effects of image reconstruction .... Computable mean-square-based fidelity metrics are developed by which both component-level and system-level performance can be quantified. In addition, ... system performance can be assessed qualitatively by visualizing the output image as the sum of three component images. each of which relates to a corresponding fidelity metric."

Digital Image Restoration — Eliminating the Blur for a Clearer Picture, M. R. Banham and A. K. Katsagellos (1997) [1]

This article provides a review and analysis of modeling techniques for digital image restoration under the general headings "Where have we been?," "Where are we now?," and "Where are we going?" Under the latter heading, the authors affirm the c/d/c model. "Researchers are now attempting to improve the models used in identification and restoration by incorporating better prior knowledge into the problem. For example, it has been shown that digital restoration may fail when incomplete system models are used [22]. One example of an incomplete system model is that which excludes the image formation, or image gathering, process. In moving from the continuous to the discrete domain, the sampling and reconstruction procedures may introduce enough distortion in there [sic] own right to influence a poor digital restoration. By addressing the continuous to discrete step directly in the system/restoration model, the degradation is better modeled, and improvements can be made over the incomplete system approach. Though more complex than classical approaches, techniques of the future will likely make use of more complete models to provide better image restorations."

Information-Theoretic Assessment of Sampled Imaging Systems, F. O. Huck *et al.* (1999) [9]

"By rigorously extending modern communication theory to the assessment of sampled imaging systems, we develop the formulations that are required to optimize the performance of these systems within the critical constraints of image gathering, data transmission, and image display. The goal of this optimization is to produce images with the best possible visual quality for the wide range of statistical properties [that one normally encounters in] the radiance field of natural scenes ... Extensive computational results are presented to assess the performance of sampled imaging systems in terms of information rate, theoretical minimum data rate, and fidelity."

Incomplete System Models Can Cause Image Restoration Failures, S. K. Park and R. Hazra (1995) [22]

A comparison of the qualitative restoration results obtained by using the c/d/c system model versus using a less comprehensive continuous-input / continuous-output (c/c) system model.

Oversampling Requirements for Pixelated-Imager Systems, O. Hadar and G. D. Boreman (1999) [6]

"The image quality resulting from a 2-D image-sampling process by an array of pixels is described. The description is based on a Fourier transformation of the Wigner-Seitz cell, which transforms a unit cell of the sampling lattice in the spatial domain into a bandwidth cell in the spatial-frequency domain. The area of the resulting bandwidth cell is a quantitative measure of the image fidelity of the sampling process. We compare the image-quality benefits of three different over-sampling geometries in terms of the modulation transfer function (MTF) as a function of the amount of oversampling used."

Influence of Sampling on Target Recognition and Identification, R. Vollmerhausen *et al.* (1999) [28]

"Two perception experiments are conducted to quantify the relationship between imager sampling artifacts and target recognition and identification performance using that imager. The results of these experiments show that in-band aliasing (aliasing that overlaps the base-band signal) does not degrade target identification performance, but out-of-band aliasing (such as visible display raster) degrades identification performance significantly.

Aliasing had less impact on the recognition task than the identification task, but both in-band and out-of-band aliasing moderately degrades recognition performance. Based on these experiments and other results reported in the literature, it appears that in-band aliasing has a strong effect on low-level discrimination tasks such as point (hot-spot) detection; out-of-band aliasing has only a minor impact on these tasks. For high-level discrimination tasks such as target identification, however, out-of-band aliasing has a significant impact on performance, whereas in-band aliasing has a minor effect. For intermediate-level discrimination tasks such as target-recognition, both in-band and out-of-band aliasing have a moderate impact on performance. ... The degraded performance due to undersampling is modeled as an effective increase in system blur ... "

Sampling Criteria for Sensor Simulation, E. Jacobs and T. C. Edwards (1999) [11]

"Electro-optic sensor simulation and sensor design require a common understanding of spatial sampling. As part of the development of a high fidelity sensor simulation, the U.S. Army Night Vision Electronic Sensors Directorate has developed a methodology that enables the simulation designer to minimize the spatial sampling rate of the input image based on the sensor parameters. Gabor information theory is used to define the limit imposed on the ability to simultaneously represent a spatial function and its Fourier transform. This is then coupled with the necessary consequences of sampling the image to develop limits that ensure minimal error. Resolution requirements for synthetic presensor imagery can be developed using this formalism. These requirements are not based on the instantaneous field of view (IFOV) or on the detector width alone but are based on the full sensor point spread function."

Artificial Scenes and Simulated Imaging, S. E. Reichenbach *et al.* (1991) [25]

"This paper describes a software simulation environment for controlled image processing research. The simulation is based on a comprehensive model of the end-to-end imaging process that accounts for statistical characteristics of the scene, image formation, sampling, noise, and display reconstruction."

Developing Operational Performance Metrics Using Image Comparison Metrics and the Concept of Degradation Space, C. E. Halford *et al.* (1999) [7]

"A technique for determining relative degradations from image metrics is presented along with a technique for predicting sensor performance from metrics. These techniques are illustrated with degradations of blur and noise in thermal imagery. These uses of metrics are depicted as mappings among a degradation space, an image quality metric space, and an operational performance space. This technique has utility in sampled imagery applications where input and output image comparison is possible, e.g. validation of an infrared scene projector (IRSP), testing image compression algorithms, image simulation, etc. Such applications have a known input image and a degraded output image. With the input image, one can characterize the output image in terms of its degradations relative to the input. The concept of a degradation space leads to developing an Operational Performance Metric (OPM) in terms of more traditional Image Quality Metrics (IQMs). The technique is illustrated using empirical results for human observers performing recognition tasks with thermal imagery in a degradation space of blur and noise."

1.4 Comparable Interactive Environments

The search for interactive simulation environments capable of facilitating the design and analysis of end-to-end systems unearthed a variety of applications aimed at both the UNIX and Microsoft Windows platforms. Although some of the environments were extremely impressive, end-to-end systems were hard to find: most displayed only input and output, and none were based on a comprehensive 2-D c/d/c system model. The primary purpose of the 2-D tools was usually image or data visualization. A representative sample of the applications that were found is described here.

1.4.1 Two-dimensional applications

System Image Analyzer from JCD Publishing [33]

System Image Analyzer (SIA) allows a user to design a linear imaging system and to visualize the output of that system. Like some other environments which provide linear system design capability, SIA displays a collection of icons which represent the prototype components of the linear system to be built. The user selects and combines these components in series to create an end-to-end system and then has the capability to change the parameters of these components. SIA has the functionality that most closely resembles the environment. It does not have the capability to implement the full c/d/c system model, however, because no component which simulates the effects of sampling is included. SIA is a Microsoft Windows-based system.

The Image Toolbox with Khoros Pro from Khoral Research Inc. [30]

Khoros Pro is software technology for data analysis and visualization, for modeling and simulation, and for software development and maintenance. It contains GUI creation capabilities. The Image Toolbox is a library of routines, designed for use with Khoros Pro, that provides image manipulation algorithms including convolution and various frequency filter design operators. Khoros Pro and the environment which is the subject of this dissertation are not directly analogous, but Khoros Pro with the Image Toolbox could possibly have been utilized to develop the environment, had the initial objectives not precluded the use of proprietary software. Khoros Pro 2001 was developed on UNIX-based operating systems. Khoros Pro 2000 also runs on Windows NT.

PV-WAVE Extreme Advantage from Visual Numerics [31]

PV-WAVE for UNIX, Windows 95, Windows NT, and OpenVMS is advanced visual data analysis software. PV-WAVE's interface allows the user to transform raw data into high-quality full-color graphics, images and maps. The system's visual exploration tools provide a GUI which is specifically designed for visual data analysis tasks. Executables (programs that can be run) prepared on one platform can be transferred to another without recompilation. The PV-WAVE: Image Processing Toolkit is a Toolbox designed for use with PV-WAVE and is incorporated into PV-WAVE Extreme Advantage. The toolkit not only provides a general purpose set of image display and image processing operations but also an extensive set of filters and transforms. Tools comparable to those which are incorporated in the environment include linear and other spatial domain filters, frequency domain filters, noise generation and noise removal. Again, PV-WAVE Extreme Advantage and the environment which is the subject of this dissertation are not directly analogous, but, like Khoros, PV-WAVE could possibly have been utilized to develop the environment, had the initial objectives not precluded the use of proprietary software.

xv — Interactive Image Display for the X Window System by John Bradley [3]

xv is an interactive image manipulation program for the X Window System. It can operate on many different image formats and on all X displays known to the *xv* author. The application incorporates image enhancement techniques and data processing algorithms, as well as the capability to resize and reformat images, but it is not an imaging system design tool. *xv* is a UNIX-based application and was much used as an auxiliary application in the development of the environment.

1.4.2 One-dimensional applications

SystemView from ELANIX Inc. [34]

SystemView incorporates high-speed design and evaluation processing embedded in an intuitive design environment. It runs under Microsoft Windows with minimal hardware requirements, and features comprehensive analog and digital design tools for use in such applications as DSP, communications, signal processing and control. SystemView provides an approach to analog-digital filter design and discrete time linear system design which is not dissimilar to the methodology used by SIA but with a feel that is much more sophisticated. The systems are designed using graphical templates, the parameters can be modified at will, and then processing is initiated. The system output can be graphically displayed in the time, frequency or phase domain.

AFDPLUS from Webb Laboratories [29]

AFDPLUS is an active filter design environment developed for the commercial active filter design community. It will analyze the completed design in the frequency and time domains. AFDPLUS features user-defined filter circuits, interactive time and frequency domain graphics, schematic display and print functions, and multiple filter combinations for the design and analysis of complex systems.

PV-WAVE: Signal Processing Toolkit from Visual Numerics [31]

PV-WAVE: Signal Processing Toolkit is another Toolbox designed for use with PV-WAVE. The Toolbox provides a broad selection of predefined and readily customized DSP functions. It includes filter analysis functions and both classical and advanced filter designs.

1.5 Overview

The remainder of this dissertation is arranged in the following manner. Chapter 2 describes the mathematical model used in the environment at both the conceptual and specification levels. The next chapter discusses this model at the implementation level with particular emphasis on the data structures utilized and the algorithms used to manipulate them. Chapter 4 addresses the time versus memory concerns which apply in this particular case. The means of optimizing both response time and memory usage are discussed. Model verification and validation are the topics covered in Chapter 5; extensive results are presented. In Chapter 6, a summary is given, and conclusions drawn from the research are presented. Because there is always the possibility for improvement, a final section suggests additions and changes which could increase the functionality of the environment.

Chapter 2

Mathematical Model

The environment is based on the continuous-input/discrete-processing/continuous-output (c/d/c) system model, which is well described in recent literature [9, 24, 27]. This comprehensive model has been shown to be better suited for simulating end-to-end digital imaging systems than its more common but less complete counterparts [1, 22] — the continuous-input/continuous-output (c/c) imaging system model and the discrete-input/discrete-output (d/d) imaging system model. As stated by Park and Rahman in their recent contribution to Optical Engineering [24], the first models implemented were c/c models, which modeled the image, at all stages of the end-to-end system, as a function. This allowed mathematical filter performance analysis, but the model was not applicable to sampled imaging systems. With the advent of commonly-available digital computing, the d/d model became popular. The image was now represented as a 2-D array at all stages and matrix theory was applied to analyze system performance. However, because the representation of the image started and ended life in digitized form, no simulation of continuous-to-discrete sampling effects or discrete-to-continuous reconstruction effects could be introduced into the model.

To comprehensively model a sampled imaging system it is necessary to utilize a c/d/c system model which incorporates continuous representations of the image at system input and output along with digitized (matrix) representations of the image after sampling, during filtering, and prior to reconstruction.

Development of the c/d/c system model can be considered to occur at three levels:

- the conceptual level;
- the specification or design level;
- the implementation level.

In this chapter, Section 2.1 describes the model at the conceptual level. Section 2.2 defines the end-to-end system components and parameters and develops the mathematical model at the specification level. In the next chapter, the c/d/c system model is described at the implementation level, relative to the environment.

2.1 The C/D/C System Model — Conceptual Level

The simulation of any digital imaging system should take into account the series of changes imposed on the input scene as it is processed in a real-world end-to-end system. These are:

- the blurring (low-pass filtering) caused by the OTF of the lens of the device used in the acquisition of the scene;
- the noise-like aliasing artifacts due to sampling (digitization);
- the additive random noise due to device electronics and quantization;

- the digital (high-boost) filtering used to compensate for acquisition blurring without enhancing the aliasing due to sampling or the artifacts due to noise;
- the effects of reconstruction filtering which attempt to faithfully reproduce an image of the input scene and suppress the aliasing.

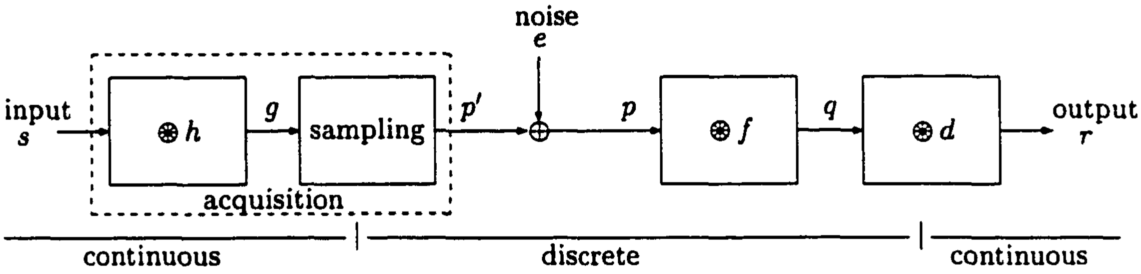


Figure 2.1: The continuous/discrete/continuous (c/d/c) system model.

Simulation can be accomplished using the continuous/discrete/continuous (c/d/c) system model shown in Figure 2.1 [19, 22, 24]. The model components are:

- input, s, a 2-D function;
- acquisition filter, h, a 2-D function;
- spatial sampling, a linear operation;
- additive noise, e, a 2-D array;
- digital filter, f, a 2-D array;
- reconstruction filter, d, a 2-D function;
- output, r, a 2-D function.

In Section 2.2, the mathematical representation of the c/d/c system model is presented. Two initial subsections identify (i) the system parameters and their interrelationships and (*ii*) modeling assumptions. Subsequent subsections develop the mathematical model for each of the model components listed above.

2.2 The C/D/C System Model — Specification Level

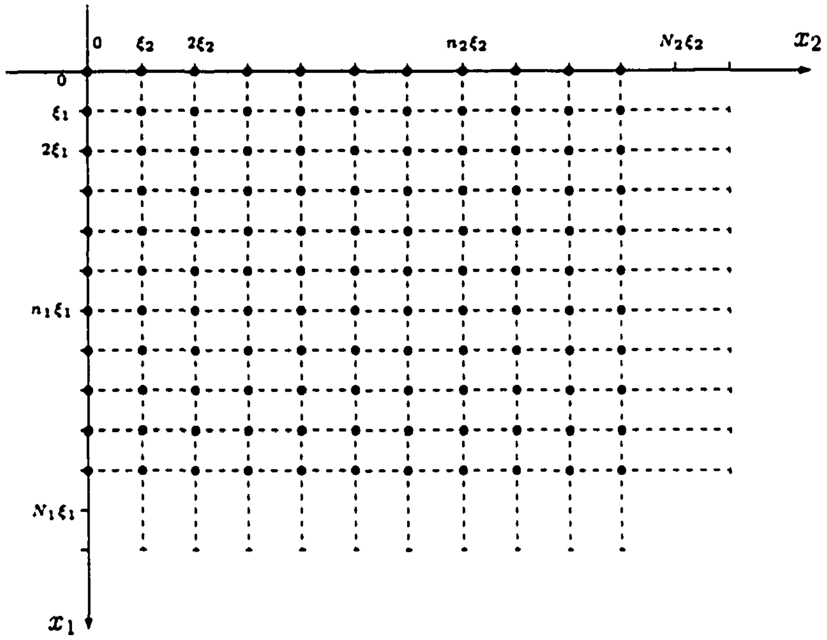


Figure 2.2: The 2-D (x_1, x_2) spatial coordinate system.

The spatial coordinate system used to reference the continuous and sampled representations of an input scene is the 2-D (x_1, x_2) system shown in Figure 2.2. Each grid point represents the center of a sensor in the acquisition (continuous-to-discrete) subsystem and the units of linear measure are the vertical and horizontal inter-sample distances ξ_1 and ξ_2.

2.2.1 System parameters

The parameters which define the c/d/c system model are:

- sampling grid size, $N_1 \times N_2$ (integer valued);

- vertical and horizontal inter-sample distances, ξ_1, ξ_2 (real valued);
- period (FOV) of the input scene, $P_1 \times P_2 = N_1\xi_1 \times N_2\xi_2$ (real valued);
- representation passband parameters, $\tau_1 \times \tau_2 = iN_1 \times iN_2$, where $i = 1, 2, 3, 4$ is user selected (integer valued);
- reconstruction passband parameters, $\tau_{r_1} \times \tau_{r_2} = kN_1 \times kN_2$, where $k = i, 4$ is user selected (integer valued);
- vertical and horizontal acquisition filter parameters, β_1, β_2 (real valued);
- vertical and horizontal digital filter parameters, λ_1, λ_2 (real valued);
- vertical and horizontal reconstruction filter parameters, α_1, α_2 (real valued);
- signal-to-noise ratio of the additive noise, SNR (real valued).

The $P_1 \times P_2$ field of view (FOV) is that part of the scene which is projected onto the sensors of the $N_1 \times N_2$ sampling grid. The model represents the scene within the FOV by a Fourier series (see Subsection 2.2.2) as a continuous, periodic function in the spatial domain and as a corresponding discrete, aperiodic matrix representation in the frequency domain. The spatial domain representation is periodic with period $P_1 \times P_2$. As illustrated in Figure 2.2, the $P_1 \times P_2$ period is defined in terms of the 2-D sampling grid size, $N_1 \times N_2$, and the inter-sample area, $\xi_1 \times \xi_2$, by $P_1 \times P_2 = N_1\xi_1 \times N_2\xi_2$. In the frequency domain representation, it is necessary (and has been shown valid — see Subsection 4.2.1) to assume that the representation is band-limited, i.e., that there are only a finite number of Fourier coefficients that differ from zero. The representation passband parameters, τ_1, τ_2, define the cut-off frequencies of the band-limited input scene. Because band-limiting the input scene should not be allowed to mask any significant effects of aliasing, τ_1 and τ_2 are chosen to be integer multiples of N_1 and N_2 respectively. In the environment, subject to the relationships specified above and some software constraints, each of the c/d/c system model parameters is under user control.

2.2.2 Modeling assumption (Fourier series, complex form)

The default modeling assumption used as the basis for the implementation of the c/d/c system model is that a real-valued function s in two variables measured over a finite area, $P_1 \times P_2$, can be represented by a band-limited Fourier series as

$$s(x_1, x_2) = \sum_{|\nu_1| \le \tau_1} \sum_{|\nu_2| \le \tau_2} \hat{S}[\nu_1, \nu_2] \exp(i2\pi\nu_1 x_1/P_1) \exp(i2\pi\nu_2 x_2/P_2)$$

for $(x_1, x_2) \in \Re \times \Re$, where P_1 is the height of the area and P_2 is the width of the area. The complex-valued Fourier coefficients that define the frequency domain representation of s at the frequency $[\nu_1/P_1, \nu_2/P_2]$ are

$$\hat{S}[\nu_1, \nu_2] = \frac{1}{P_1 P_2} \int_{P_1} \int_{P_2} s(x_1, x_2) \exp(-i2\pi\nu_1 x_1/P_1) \exp(-i2\pi\nu_2 x_2/P_2) dx_1 dx_2$$

for $[\nu_1, \nu_2] \in \mathcal{T}_1 \times \mathcal{T}_2$.[1] The representation passband parameters, τ_1, τ_2, are the cutoff frequency indices for the summation. The corresponding cut-off frequencies are τ_1/P_1, τ_2/P_2. Because the input function $s(x_1, x_2)$ is real-valued, it can be shown that $\hat{S}[-\nu_1, -\nu_2] = \hat{S}^*[\nu_1, \nu_2]$ and $\hat{S}[-\nu_1, \nu_2] = \hat{S}^*[\nu_1, -\nu_2]$, as illustrated in Figure 2.3.

[1] $\mathcal{T}_1 = \{0, \pm1, \pm2, \ldots, \pm\tau_1\}$; $\mathcal{T}_2 = \{0, \pm1, \pm2, \ldots, \pm\tau_2\}$.

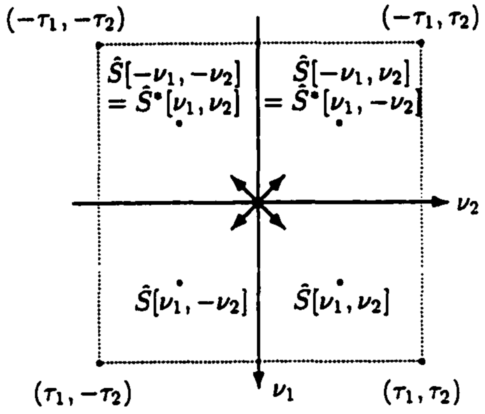


Figure 2.3: Complex conjugacy in the frequency domain.

2.2.3 Pre-processing and loading input

The environment offers two methods of input scene selection.

1. The user may choose a digitized (sampled) spatial domain input scene from a menu of input scenes available in the environment's library.

2. The user may choose to synthesize an input scene by defining it directly in the frequency domain. This is achieved by selection from a function menu.

2.2.3.1 digitized input scene

Super-resolution spatial domain images are defined by real brightness values specified at a grid of points within a 2-D (x_1, x_2) spatial coordinate system that is finer than the $N_1 \times N_2$ sampling grid shown in Figure 2.2. Such digitized input scenes must be *pre-processed* before being included in the environment's library. If necessary, an input scene is clipped to the maximum size acceptable to the environment and then, if either input scene dimension is not a power of 2, padding, cropping or resampling is performed as specified by the user.

The output from these two pre-processing stages is an $M_1 \times M_2$ matrix of real brightness values $s'[m_1, m_2]$ overlaid on the $P_1 \times P_2$ FOV of the c/d/c system model. The matrix is assumed to be periodic with period $M_1 \times M_2$.

The Discrete Fourier Transform (DFT) matrix $\hat{s}'$, periodic with period $M_1 \times M_2$, corresponding to the digitized input scene s', is defined as

$$\hat{s}'[\nu_1, \nu_2] = \frac{1}{M_1 M_2} \sum_{m_1=0}^{M_1-1} \sum_{m_2=0}^{M_2-1} s'[m_1, m_2] \exp(-i2\pi\nu_1 m_1/M_1) \exp(-i2\pi\nu_2 m_2/M_2)$$

for $[\nu_1, \nu_2] \in \mathcal{T}_1 \times \mathcal{T}_2$. The $\hat{s}'[\nu_1, \nu_2]$ coefficients are computed using a 2-D Fast Fourier Transform (FFT) algorithm, as described in Section 2.2.9. Then, the complex-valued Fourier coefficient matrix representation $\hat{S}'$ corresponding to the band-limited input scene s' is constructed from the $\hat{s}'$ coefficients as indicated.

$$\hat{S}'[\nu_1, \nu_2] = \begin{cases} \hat{s}'[\nu_1, \nu_2] & \nu_1 = 0, \pm 1, \pm 2, \ldots, \pm(M_1/2 - 1), \\ & \nu_2 = 0, \pm 1, \pm 2, \ldots, \pm(M_2/2 - 1); \\ \hat{s}'[\nu_1, \nu_2]/2 & \nu_1 = 0, \pm 1, \pm 2, \ldots, \pm(M_1/2 - 1), \\ & \nu_2 = \pm M_2/2; \\ \hat{s}'[\nu_1, \nu_2]/2 & \nu_1 = \pm M_1/2, \\ & \nu_2 = 0, \pm 1, \pm 2, \ldots, \pm(M_2/2 - 1); \\ \hat{s}'[\nu_1, \nu_2]/4 & \nu_1 = \pm M_1/2, \\ & \nu_2 = \pm M_2/2. \end{cases}$$

Because the input function $s'(x_1, x_2)$ is real-valued, $\hat{S}'[\nu_1, \nu_2]$ has the conjugate symmetry illustrated in Figure 2.3. Therefore, to conserve memory, only $\hat{S}'[\nu_1, \nu_2]$ coefficients in the range $\nu_1 = 0, 1, \ldots, M_1/2$; $\nu_2 = 0, \pm 1, \ldots, \pm M_2/2$ are stored in the environment's library.

Pre-processing occurs off-line and once only. When the Fourier domain representation of a digitized input scene has been successfully stored in the environment's library, it is thereafter readily available for use at run time.

2.2.3.2 synthesized input scene

Synthesized input scenes are directly defined in the frequency domain. At run time, when a synthesized input scene is selected by the user, $\hat{S}'[\nu_1, \nu_2]$ coefficients in the range $\nu_1 = 0, 1, \ldots, \tau_1$; $\nu_2 \in \mathcal{T}_2$ are calculated and stored in memory, where τ_1 and τ_2 are respectively the vertical and horizontal input scene passband parameters selected by the user. Because the function $s'(x_1, x_2)$ is real-valued, $\hat{S}'[\nu_1, \nu_2]$ coefficients in the range $\nu_1 = -1, -2, \ldots, -\tau_1$; $\nu_2 \in \mathcal{T}_2$ can be generated by conjugate symmetry when required for computation, as discussed previously.

<u>pulse train function</u>

This synthesized input scene is described in Section 5.1.1.

<u>sharkstooth function</u>

This synthesized input scene is defined as

$$\hat{S}'[\nu_1, \nu_2] = \begin{cases} \frac{b}{2} \cdot \frac{b}{2} & \nu_1 = \nu_2 = 0 \\ \frac{b}{2} \cdot \frac{-b}{\pi^2 \nu_2^2} \left(1 - (-1)^{\nu_2}\right) & \nu_1 = 0; \nu_2 = 1, 2, \ldots, \tau_2 \\ \frac{-b}{\pi^2 \nu_1^2} \left(1 - (-1)^{\nu_1}\right) \cdot \frac{b}{2} & \nu_1 = 1, 2, \ldots, \tau_1; \nu_2 = 0 \\ \frac{-b}{\pi^2 \nu_1^2} \left(1 - (-1)^{\nu_1}\right) \frac{-b}{\pi^2 \nu_2^2} \left(1 - (-1)^{\nu_2}\right) & \nu_1 = 1, 2, \ldots, \tau_1; \nu_2 = 1, 2, \ldots, \tau_2 \end{cases}$$

This is a separable function and so can be computed as the product of two coefficient vectors.

2.2.3.3 input scene filtering

The *convolution* $s = t \circledast s'$ which results from *post-processing* the scene function s', periodic with period $P_1 \times P_2$, through the 2-D scene filter t to obtain the band-limited scene function

s, also periodic with period $P_1 \times P_2$, is defined in the spatial domain as

$$s(x_1, x_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} t(x_1' - x_1, x_2' - x_2) s'(x_1', x_2') dx_1' dx_2'$$

for $(x_1, x_2) \in \Re \times \Re$. From the *Convolution Theorem*, $s = t \circledast s'$ if and only if

$$\hat{S}[\nu_1, \nu_2] = \hat{T}\left(\nu_1/P_1, \nu_2/P_2\right) \hat{S}'[\nu_1, \nu_2]$$

for $[\nu_1, \nu_2] \in \mathcal{Z} \times \mathcal{Z}$,[2] where

$$\hat{T}(\omega_1, \omega_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} t(x_1, x_2) \exp(-i2\pi\omega_1 x_1) \exp(-i2\pi\omega_2 x_2) dx_1 dx_2$$

for $(\omega_1, \omega_2) \in \Re \times \Re$. Therefore, *convolution* can be implemented in the frequency domain as multiplication on a frequency-by-frequency basis. The Fourier transform function $\hat{T}$ is sampled in the frequency domain at the frequencies $\omega_1 = \nu_1/P_1$ for $\nu_1 \in \mathcal{T}_1$ and $\omega_2 = \nu_2/P_2$ for $\nu_2 \in \mathcal{T}_2$ and the Fourier representation of $\hat{S}$ is generated using complex arithmetic.

For all scene filters, post-processing results in a matrix representation of $\hat{S}[\nu_1, \nu_2]$ coefficients in the range $\nu_1 = 0, 1, \ldots, \tau_1$; $\nu_2 \in \mathcal{T}_2$. $\hat{S}[\nu_1, \nu_2]$ coefficients in the range $\nu_1 = -1, -2, \ldots, -\tau_1$; $\nu_2 \in \mathcal{T}_2$ are generated by conjugate symmetry when required for computation. Although this could indicate a lack of understanding on the user's part and would result in unnecessary processing, if a digitized input scene is selected and $\tau_1 > M_1/2$ and/or $\tau_2 > M_2/2$, then the Fourier representation matrices are zero-padded.

[2] $\mathcal{Z} = \{0, \pm 1, \pm 2, \ldots\}$.

2.2.3.4 default implementation

The environment offers a menu of scene filters, each with user-controlled, normalized parameters, ψ_1, ψ_2:

- Cesaro;
- Dirichlet;
- Lanczos.

The input scene is automatically post-processed at run time, when the user selects an input scene. The default post-processor is a Dirichlet filter which simply applies the product of the input scene filter parameters and user-selected representation passband parameters to band-limit the input scene. Alternatively, the user may elect to use Lanczos or Cesaro filtering to smooth the band-limited Fourier representation of the input scene and thereby reduce ringing effects.

2.2.4 Acquisition filter

The *convolution* $g = h \circledast s$ which results from passing the scene function s, periodic with period $P_1 \times P_2$, through the 2-D acquisition filter h to obtain the image function g, also periodic with period $P_1 \times P_2$, is defined in the spatial domain as

$$g(x_1, x_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x_1' - x_1, x_2' - x_2) s(x_1', x_2') dx_1' dx_2'$$

for $(x_1, x_2) \in \Re \times \Re$. From the *Convolution Theorem*, $g = h \circledast s$ if and only if

$$\hat{G}[\nu_1, \nu_2] = \hat{H}(\nu_1/P_1, \nu_2/P_2) \hat{S}[\nu_1, \nu_2] \tag{2.1}$$

for $[\nu_1, \nu_2] \in \mathcal{Z} \times \mathcal{Z}$, where

$$\hat{H}(\omega_1, \omega_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x_1, x_2) \exp(-i2\pi\omega_1 x_1) \exp(-i2\pi\omega_2 x_2) dx_1 dx_2$$

for $(\omega_1, \omega_2) \in \Re \times \Re$. Therefore, *convolution* can be implemented in the frequency domain as multiplication on a frequency-by-frequency basis. The Fourier transform function $\hat{H}$ is sampled in the frequency domain at the frequencies $\omega_1 = \nu_1/P_1$ for $\nu_1 \in \mathcal{T}_1$ and $\omega_2 = \nu_2/P_2$ for $\nu_2 \in \mathcal{T}_2$ and the Fourier representation of $\hat{G}$ is generated using complex arithmetic.

Because the function $g(x_1, x_2)$ is real-valued and is band-limited by the input function representation passband parameters τ_1, τ_2, only $\hat{G}[\nu_1, \nu_2]$ coefficients in the range $\nu_1 = 0, 1, \ldots, \tau_1$; $\nu_2 \in \mathcal{T}_2$ are computed and stored. $\hat{G}[\nu_1, \nu_2]$ coefficients in the range $\nu_1 = -1, -2, \ldots, -\tau_1$; $\nu_2 \in \mathcal{T}_2$ are generated by complex conjugacy, as illustrated in Figure 2.3.

2.2.4.1 default implementation

The environment offers a menu of acquisition filters, each with user-controlled parameters:

- box;
- Gaussian;
- composite.

The default acquisition filter is Gaussian, defined in the spatial domain as the function

$$h(x_1, x_2) = \frac{1}{\beta_1 \beta_2} \exp\left(\frac{-\pi x_1^2}{\beta_1^2}\right) \exp\left(\frac{-\pi x_2^2}{\beta_2^2}\right) \tag{2.2}$$

for $(x_1, x_2) \in \Re \times \Re$. The real-valued parameters β_1 and β_2 are independently controlled and allow the user to adjust the level of blurring. The continuous aperiodic Fourier transform

$\hat{H}$ corresponding to the Gaussian filter h is

$$\hat{H}(\omega_1, \omega_2) = \exp(-\pi\beta_1^2\omega_1^2)\exp(-\pi\beta_2^2\omega_2^2) \tag{2.3}$$

for $(\omega_1, \omega_2) \in \Re \times \Re$. In this case, because the Fourier transform corresponding to the Gaussian filter is a separable product, i.e., $\hat{H}(\omega_1, \omega_2) = \hat{H}_1(\omega_1)\hat{H}_2(\omega_2)$, it is only necessary to compute and store $\hat{H}_1(\omega_1) = \exp\left(-\pi\beta_1^2\omega_1^2\right)$ at $\omega_1 = \nu_1/P_1$, where $\nu_1 = 0, 1, 2, \ldots, \tau_1$, and $\hat{H}_2(\omega_2) = \exp\left(-\pi\beta_2^2\omega_2^2\right)$ at $\omega_2 = \nu_2/P_2$, where $\nu_2 = 0, 1, 2, \ldots, \tau_2$. The stored matrix corresponding to the horizontal component is extended using complex conjugacy to include $\hat{H}_2$ values sampled at the frequencies $\omega_2 = \nu_2/P_2$, where $\nu_2 = -1, -2, \ldots, -\tau_2$.

2.2.5 Spatial sampling

Given a real periodic band-limited image function g defined by

$$g(x_1, x_2) = \sum_{|\nu_1| \le \tau_1} \sum_{|\nu_2| \le \tau_2} \hat{G}[\nu_1, \nu_2] \exp(i2\pi\nu_1 x_1/P_1)\exp(i2\pi\nu_2 x_2/P_2)$$

for $(x_1, x_2) \in \Re \times \Re$, the periodic $N_1 \times N_2$ matrix

$$p'[n_1, n_2] = g(n_1\xi_1, n_2\xi_2)$$

is generated by sampling the function N_1 times per vertical period and N_2 times per horizontal period with inter-sample distance ξ_1, ξ_2. The periodic discrete Fourier transform matrix $\hat{p}'$ corresponding to p', is periodic with period $N_1 \times N_2$.

Frequency folding, which occurs when a continuous function is sampled, can result in image degradation due to aliasing. The effect of this phenomenon is defined by the *Periodic*

Sampling Theorem which states that

$$\hat{p}'[\nu_1, \nu_2] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \hat{G}[\nu_1 - k_1 N_1, \nu_2 - k_2 N_2] \tag{2.4}$$

for $[\nu_1, \nu_2] \in \mathcal{Z} \times \mathcal{Z}$. Because g is band-limited by the input matrix representation passband parameters, τ_1 and τ_2, for any $[\nu_1, \nu_2]$ the infinite series becomes finite.

2.2.6 Additive noise

Stochastic noise is generated using a random variate generator to create a (pseudo-)random noise matrix, e, periodic with period $N_1 \times N_2$. The discrete Fourier transform matrix $\hat{e}$, periodic with period $N_1 \times N_2$, is generated by taking the DFT of e. That is,

$$\hat{e}[\nu_1, \nu_2] = \frac{1}{N_1 N_2} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} e[n_1, n_2] \exp\left(-i2\pi\left(\frac{\nu_1 n_1}{N_1}\right)\right) \exp\left(-i2\pi\left(\frac{\nu_2 n_2}{N_2}\right)\right)$$

for $[\nu_1, \nu_2] \in \mathcal{N}_1 \times \mathcal{N}_2$.[3] The discrete Fourier transform matrix $\hat{p}$, periodic with period $N_1 \times N_2$, is then generated by adding transform coefficients in the frequency domain, on a frequency-by-frequency basis. That is,

$$\hat{p}[\nu_1, \nu_2] = \hat{p}'[\nu_1, \nu_2] + \hat{e}[\nu_1, \nu_2]$$

for $[\nu_1, \nu_2] \in \mathcal{N}_1 \times \mathcal{N}_2$.

2.2.7 Digital filter

The *convolution* $q = f \circledast p$ which results from passing the matrix p, periodic with period $N_1 \times N_2$, through the 2-D digital filter f to obtain the matrix q, also periodic with period

[3] $\mathcal{N}_1 = \{0, 1, 2, \ldots, N_1 - 1\}$; $\mathcal{N}_2 = \{0, 1, 2, \ldots, N_2 - 1\}$.

$N_1 \times N_2$, is defined in the spatial domain as

$$q[n_1,n_2] = \sum_{n_1'=-\infty}^{\infty} \sum_{n_2'=-\infty}^{\infty} f[n_1' - n_1, n_2' - n_2] p[n_1', n_2']$$

for $[n_1,n_2] \in \mathcal{N}_1 \times \mathcal{N}_2$. From the *Convolution Theorem*, $q = f \circledast p$ if and only if

$$\hat{q}[\nu_1,\nu_2] = \hat{f}\,(\nu_1/N_1, \nu_2/N_2)\,\hat{p}[\nu_1,\nu_2] \tag{2.5}$$

for $[\nu_1,\nu_2] \in \mathcal{N}_1 \times \mathcal{N}_2$, where

$$\hat{f}(\omega_1,\omega_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \exp(-i2\pi\omega_1 n_1) \exp(-i2\pi\omega_2 n_2)$$

for $(\omega_1,\omega_2) \in \Re \times \Re$. Therefore, *convolution* can be implemented in the frequency domain as multiplication on a frequency-by-frequency basis. The Fourier transform $\hat{f}$, periodic with period 1 × 1, is sampled in the frequency domain at the frequencies $\omega_1 = \nu_1/N_1$ for $\nu_1 \in \mathcal{N}_1$, and $\omega_2 = \nu_2/N_2$ for $\nu_2 \in \mathcal{N}_2$.

The environment offers a menu of digital filters, most with user-controlled parameters, all defined directly in the frequency domain:

- all-pass;
- Constrained Least Squares (CLS) [8, 10, 17];
- inverse;
- modified inverse.

2.2.7.1 default implementation

The default digital filter is a modified inverse filter corresponding to the combination of the acquisition and reconstruction filters. The modified inverse filter is defined in the frequency domain as

$$\hat{f}(\omega_1,\omega_2) = \frac{\hat{H}^*(\omega_1 - \lfloor\omega_1\rfloor)\hat{D}^*(\omega_1 - \lfloor\omega_1\rfloor)}{(|\hat{H}(\omega_1 - \lfloor\omega_1\rfloor)\hat{D}(\omega_1 - \lfloor\omega_1\rfloor)|^2 + \lambda_1^2)} \frac{\hat{H}^*(\omega_2 - \lfloor\omega_2\rfloor)\hat{D}^*(\omega_2 - \lfloor\omega_2\rfloor)}{(|\hat{H}(\omega_2 - \lfloor\omega_2\rfloor)\hat{D}(\omega_2 - \lfloor\omega_2\rfloor)|^2 + \lambda_2^2)} \quad (2.6)$$

for $(\omega_1,\omega_2) \in \Re \times \Re$. The real-valued parameters λ_1 and λ_2 are independently controlled and allow the user to adjust the vertical and horizontal frequency responses of the separable digital filter.

2.2.8 Reconstruction filter

Given an $N_1 \times N_2$ matrix q, periodic with period $N_1 \times N_2$, and an aperiodic reconstruction kernel function d, then the corresponding reconstructed function $r = d \circledast q$ is periodic with period $P_1 = N_1\xi_1 \times P_2 = N_2\xi_2$. The *convolution* $r = d \circledast q$ is defined in the spatial domain as

$$r(x_1,x_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} d(x_1 - n_1\xi_1, x_2 - n_2\xi_2) q[n_1,n_2]$$

for $(x_1,x_2) \in \Re \times \Re$. From the *Convolution Theorem*, $r = d \circledast q$ if and only if

$$\hat{R}[\nu_1,\nu_2] = \frac{1}{\xi_1\xi_2}\hat{D}\left(\nu_1/P_1, \nu_2/P_2\right)\hat{q}[\nu_1,\nu_2] \quad (2.7)$$

for $[\nu_1,\nu_2] \in \mathcal{Z} \times \mathcal{Z}$, where

$$\hat{D}(\omega_1,\omega_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} d(x_1,x_2)\exp(-i2\pi\omega_1 x_1)\exp(-i2\pi\omega_2 x_2)dx_1 dx_2$$

for all (ω_1, ω_2).

Therefore, *convolution* can be implemented in the frequency domain as multiplication on a frequency-by-frequency basis. Because of the periodicity of $\hat{q}$, the matrix representation of the reconstructed image is not necessarily band-limited by the representation passband parameters τ_1, τ_2. The Fourier transform $\hat{D}$ is sampled in the frequency domain at the frequencies $\omega_1 = \nu_1/P_1$ with $\nu_1 \in \mathcal{T}_{r_1}$,[4] and $\omega_2 = \nu_2/P_2$ with $\nu_2 \in \mathcal{T}_{r_2}$.[5] The reconstruction passband parameters τ_{r_1}, τ_{r_2} are the cutoff indices for the frequency-by-frequency multiplication. The corresponding cut-off frequencies are τ_{r_1}/P_1, τ_{r_2}/P_2.

Because the function $r(x_1, x_2)$ is real-valued, only $\hat{R}[\nu_1, \nu_2]$ coefficients in the range $\nu_1 = 0, 1, \ldots, \tau_{r_1}$; $\nu_2 \in \mathcal{T}_{r_2}$ are computed and stored. $\hat{R}[\nu_1, \nu_2]$ coefficients in the range $\nu_1 = -1, -2, \ldots, -\tau_{r_1}$; $\nu_2 \in \mathcal{T}_{r_2}$ are generated by complex conjugacy, as illustrated in Figure 2.3.

2.2.8.1 default implementation

The environment offers a menu of reconstruction filters, each with user-controlled parameters:

- parametric cubic;
- triangle function.

The default reconstruction filter is parametric cubic defined in the spatial domain as

$$d(x_1, x_2) = d_{\alpha_1}\left(\frac{x_1}{\xi_1}\right) d_{\alpha_2}\left(\frac{x_2}{\xi_2}\right) \tag{2.8}$$

[4] $\mathcal{T}_{r_1} = \{0, \pm 1, \pm 2, \ldots, \pm \tau_{r_1}\}$.

[5] $\mathcal{T}_{r_2} = \{0, \pm 1, \pm 2, \ldots, \pm \tau_{r_2}\}$.

where

$$d_\alpha(x) = \begin{cases} (\alpha+2)|x|^3 - (\alpha+3)|x|^2 + 1 & |x| < 1 \\ \alpha|x|^3 - 5\alpha|x|^2 + 8\alpha|x| - 4\alpha & 1 \leq |x| < 2 \\ 0 & \text{otherwise.} \end{cases}$$

The real-valued parameters α_1 and α_2 are independently controlled and allow the user to vary the frequency responses of the vertical and horizontal components of the reconstruction filter.

It can be shown that the aperiodic Fourier transform corresponding to the PCC reconstruction kernel is

$$\hat{D}(\omega_1, \omega_2) = \hat{D}_{\alpha_1}(\omega_1)\hat{D}_{\alpha_2}(\omega_2) \tag{2.9}$$

where[6]

$$\hat{D}_\alpha(\omega) = \frac{3}{(\pi\omega)^2}\left(\text{sinc}^2(\omega) - \text{sinc}(2\omega)\right) + \frac{2\alpha}{(\pi\omega)^2}\left(3\text{sinc}^2(2\omega) - 2\text{sinc}(2\omega) - \text{sinc}(4\omega)\right)$$

for $\omega \in \Re$. In this case, because the Fourier transform corresponding to the filter is a separable product, it is only necessary to compute and store $\hat{D}_{\alpha_1}(\omega_1)$ at $\omega_1 = \nu_1/P_1$ with $\nu_1 = 0, 1, 2, \ldots, \tau_{r_1}$ and $\hat{D}_{\alpha_2}(\omega_2)$ at $\omega_2 = \nu_2/P_2$ with $\nu_2 = 0, 1, 2, \ldots, \tau_{r_2}$. The stored values corresponding to the horizontal component are extended using complex conjugacy to include $\hat{D}_{\alpha_2}$ values sampled at the frequencies $\omega_2 = \nu_2/P_2$, where $\nu_2 = -1, -2, \ldots, -\tau_{r_2}$.

2.2.9 Conversion from frequency to spatial domain

To avoid undersampling when s, g and r are displayed in the typical case $\tau_1 \times \tau_2 = 2N_1 \times 2N_2$, the default display size is set to $4N_1 \times 4N_2$. However, the user has the capability to set

[6] $\text{sinc}(x) = \sin(x)/x$

$\tau_1 \times \tau_2$ to $jN_1 \times jN_2$, where $j = 1, 2, 3, 4$, and/or to set the display size to $kN_1 \times kN_2$, where $k = 1/2, 1, 2, 4, 8$. *Frequency folding* is applied to the $2jN_1 \times 2jN_2$ Fourier coefficient matrices to map them to the $kN_1 \times kN_2$ display size. Spatial domain equivalents are computed by generating the inverse Fourier transform of the matrix representations for $\hat{S}$, $\hat{G}$, and $\hat{R}$.

Given the aperiodic matrix of Fourier coefficients $\hat{S}[\nu_1, \nu_2]$, for $[\nu_1, \nu_2] \in \mathcal{T}_1 \times \mathcal{T}_2$, then the corresponding band-limited function

$$s(x_1, x_2) = \sum_{|\nu_1| \leq \tau_1} \sum_{|\nu_2| \leq \tau_2} \hat{S}[\nu_1, \nu_2] \exp(i2\pi\nu_1 x_1/P_1) \exp(i2\pi\nu_2 x_2/P_2)$$

is periodic with period $P_1 \times P_2$ and is displayed by sampling the function at the points $x_1 = 2jn_1\xi_1/k$, $x_2 = 2jn_2\xi_2/k$

$$s(2jn_1\xi_1/k, 2jn_2\xi_2/k) = \sum_{\nu_1=0}^{2\tau_1-1} \sum_{\nu_2=0}^{2\tau_2-1} \hat{S}[\nu_1, \nu_2] \exp\left(i2\pi\left(\frac{\nu_1 n_1}{2\tau_1} + \frac{\nu_2 n_2}{2\tau_2}\right)\right)$$

for $n_1 = 0, 1, 2, \ldots, 2\tau_1 - 1$ and $n_2 = 0, 1, 2, \ldots, 2\tau_2 - 1$. The displayed image will not be undersampled provided $k \geq 2j$. The inverse Fourier transforms of the matrix representations of $\hat{G}$ and $\hat{R}$ are similarly defined.

Given the periodic matrix of Fourier coefficients $\hat{p}'[\nu_1, \nu_2]$, defined for $[\nu_1, \nu_2] \in \mathcal{N}_1 \times \mathcal{N}_2$. for the purposes of displaying a spatial domain representation, the coefficient matrix is clipped or zero-padded to the display size, as appropriate. Then the $kN_1 \times kN_2$ spatial domain representation is the inverse Fourier transform

$$p'[n_1, n_2] = \sum_{\nu_1=0}^{kN_1-1} \sum_{\nu_2=0}^{kN_2-1} \hat{p}'[\nu_1, \nu_2] \exp\left(i2\pi\left(\frac{\nu_1 n_1}{kN_1} + \frac{\nu_2 n_2}{kN_2}\right)\right)$$

for $n_1 = 0, 1, 2, \ldots, kN_1 - 1$ and $n_2 = 0, 1, 2, \ldots, kN_2 - 1$. The inverse Fourier transforms of the matrix representations of $\hat{p}$ and $\hat{q}$ are similarly computed.

2.2.10 Fidelity analysis

The purpose of the fidelity metrics shown in Figure 2.4 is to serve as an evaluation tool for changes made to system parameters in the c/d/c system model. The calculations are based on *Parseval's Fourier Series Theorems.* Given a band-limited function

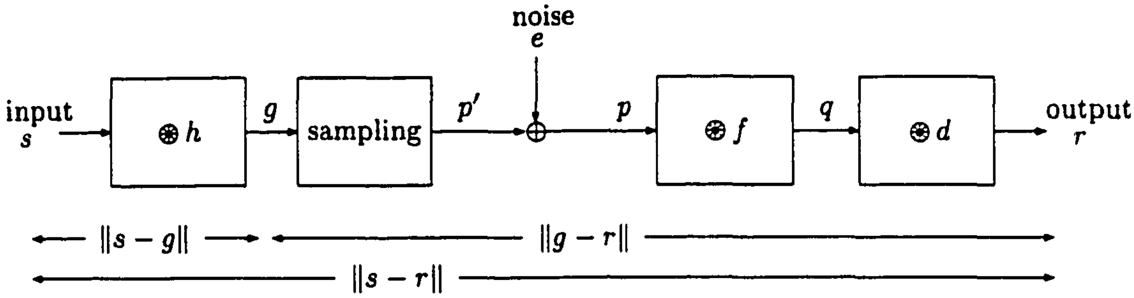


Figure 2.4: Fidelity metrics.

$$s(x_1, x_2) = \sum_{|\nu_1| \leq \tau_1} \sum_{|\nu_2| \leq \tau_2} \hat{S}[\nu_1, \nu_2] \exp\left(i2\pi\left(\frac{\nu_1 x_1}{P_1} + \frac{\nu_2 x_2}{P_2}\right)\right)$$

which is periodic with period $P_1 \times P_2$, the mean-square (MS) value of s is

$$\|s\|^2 = \frac{1}{P_1 P_2} \int_{P_1} \int_{P_2} |s(x_1, x_2)|^2 dx_1 dx_2 = \sum_{|\nu_1| \leq \tau_1} \sum_{|\nu_2| \leq \tau_2} |\hat{S}[\nu_1, \nu_2]|^2.$$

Similarly, given the band-limited function r, also periodic with period $P_1 \times P_2$, the MS difference between s and r is

$$\|s - r\|^2 = \frac{1}{P_1 P_2} \int_{P_1} \int_{P_2} |s(x_1, x_2) - r(x_1, x_2)|^2 dx_1 dx_2$$

or, equivalently,

$$\|s-r\| = \sqrt{\sum_{|\nu_1|\leq\max(\tau_1,\tau_{r_1})}\ \sum_{|\nu_2|\leq\max(\tau_2,\tau_{r_2})} |\hat{S}[\nu_1,\nu_2]-\hat{R}[\nu_1,\nu_2]|^2}. \tag{2.10}$$

In Equation (2.10), if $\max(\tau_1,\tau_{r_1}) = \tau_{r_1}$, then the $\hat{S}$ coefficients in which $\nu_1 = \tau_1+1, \tau_1+2, \ldots, \tau_{r_1}$ are known to be zero. Similarly, if $\max(\tau_2,\tau_{r_2}) = \tau_{r_2}$, then the $\hat{S}$ coefficients in which $\nu_2 = \tau_2+1, \tau_2+2, \ldots, \tau_{r_2}$ are known to be zero. In the cases where $\max(\tau_1,\tau_{r_1}) = \tau_1$ and/or $\max(\tau_2,\tau_{r_2}) = \tau_2$, the $\hat{R}$ coefficients between the limiting frequency indices of the reconstruction and representation passband parameters are known to be zero.

β	0.001	0.25	0.5	0.75	1.0	1.25
$\|s-g\|$	0.0	2.2	5.6	8.0	9.7	10.9
$\|g-r\|$	13.3	11.3	7.9	5.2	3.8	3.5
$\|s-r\|$	13.3	12.7	11.7	11.2	11.1	11.1

β	1.5	1.75	2.0	2.25	2.5	2.75
$\|s-g\|$	11.8	12.4	12.9	13.3	13.7	14.0
$\|g-r\|$	3.6	3.8	4.2	4.5	4.8	5.1
$\|s-r\|$	11.2	11.5	11.9	12.3	12.6	12.8

Table 2.1: Center row of image "aerial" with $N_2 = 64$

As an example of the utility of fidelity metrics, Table 2.1 shows 1-D results generated by varying the Gaussian acquisition filter parameter, β, while the CLS restoration filter parameter is fixed at $\lambda = 0.001$ and the parametric cubic reconstruction filter parameter is fixed at $\alpha = -0.5$, i.e., the filters are defined by

$$\hat{H}(\omega) = \exp(-\pi\beta^2\omega^2)$$

$$\hat{f}(\omega) = \frac{\xi\hat{H}^*(\omega-\lfloor\omega\rfloor)\hat{D}^*(\omega-\lfloor\omega\rfloor)}{|\hat{H}(\omega-\lfloor\omega\rfloor)\hat{D}(\omega-\lfloor\omega\rfloor)|^2+\lambda|\hat{c}(\omega-\lfloor\omega\rfloor)\hat{D}(\omega-\lfloor\omega\rfloor)|^2}$$

where $\hat{c}(\omega) = 4\sin(\pi\omega)^2$ is the CLS stabilizing functional and

$$\hat{D}_\alpha(\omega) \;=\; \frac{3}{(\pi\omega)^2}\left(\mathrm{sinc}^2(\omega) - \mathrm{sinc}(2\omega)\right) + \frac{2\alpha}{(\pi\omega)^2}\left(3\mathrm{sinc}^2(2\omega) - 2\mathrm{sinc}(2\omega) - \mathrm{sinc}(4\omega)\right)$$

for $\omega \in \Re$. Figure 2.5 is a graphical representation of Table 2.1. It is apparent from this figure that the parameter value $\beta \approx 1.125$ is optimal in the sense that it minimizes the end-to-end fidelity metric, $\|s - r\|$. A trade-off has to be achieved in selecting a value for β. If β is large, blurring becomes the dominant error factor and so $\|s - r\|$ approaches $\|s - g\|$. If β is small, the error due to aliasing becomes the dominant factor and so $\|s - r\|$ approaches $\|g - r\|$.

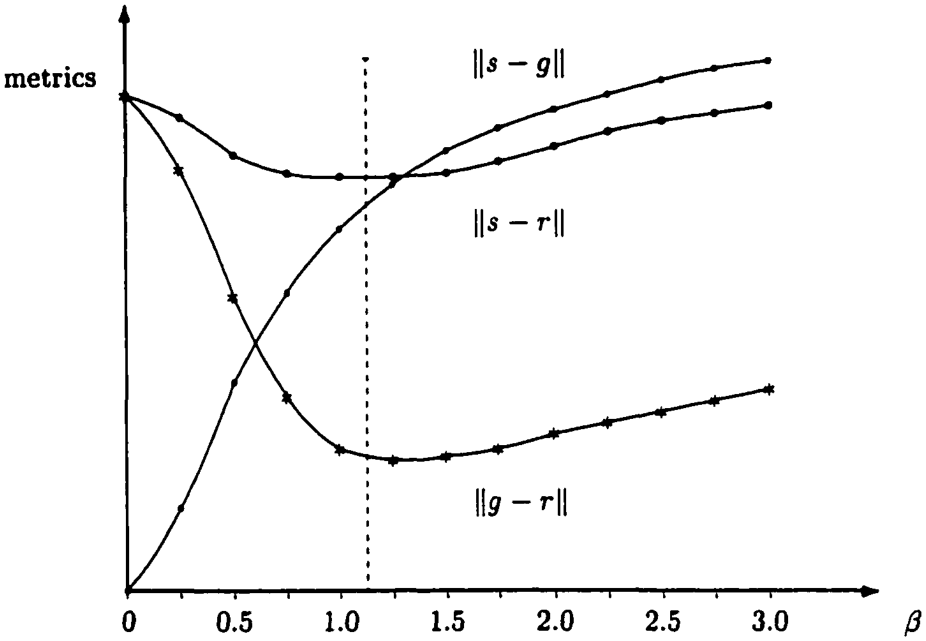


Figure 2.5: Varying β in filter definition $\hat{H}$

This trade-off is still more evident if the restoration filter is removed from the system model because restoration is an attempt to compensate for the blurring effects caused by acquisition. Table 2.2 shows 1-D results generated by varying the Gaussian acquisition filter parameter, β, as in Table 2.1, but with the CLS restoration filter replaced by an all-pass filter, i.e., a filter with the response $\hat{f}(\omega) = 1$ for all ω. In this case, the parameter value

β	.001	0.25	0.5	0.75	1.0	1.25
$\|s-g\|$	0.0	2.2	5.6	8.0	9.7	10.9
$\|g-r\|$	13.4	11.3	7.8	5.0	3.1	1.9
$\|s-r\|$	13.4	12.6	11.7	11.4	11.6	12.0
β	1.5	1.75	2.0	2.25	2.5	2.75
$\|s-g\|$	11.8	12.4	12.9	13.3	13.7	14.0
$\|g-r\|$	1.2	0.8	0.7	0.6	0.5	0.5
$\|s-r\|$	12.4	12.8	13.2	13.5	13.8	14.1

Table 2.2: Center row of image "aerial" with $N_2 = 64$ and $\hat{f}(\omega) = 1$

$\beta \approx 0.75$ is optimal, as illustrated in Figure 2.6. Moreover, Figure 2.6 shows that, without restoration filtering, $\|s-r\|$ becomes coincident with $\|s-g\|$ as the blurring (β) increases.

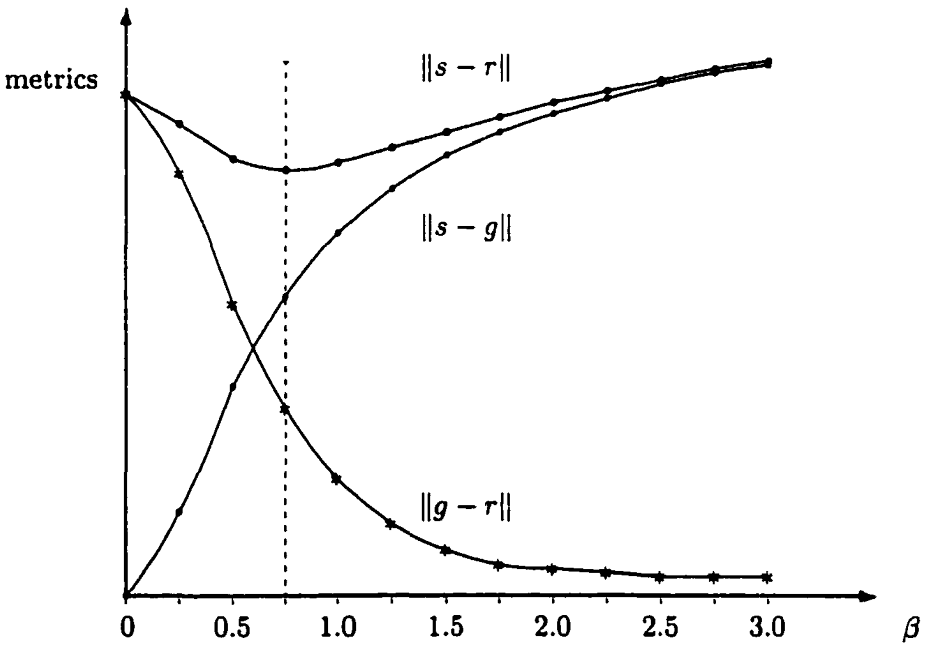


Figure 2.6: Varying β in filter definition $\hat{H}$ with $\hat{f}(\omega) = 1$

Fidelity metrics are considered useful to the operation of the environment because, to realistically employ the environment as a design tool, it is envisaged that it could be used in conjunction with an off-line, front-end process that would generate desirable ranges of system parameters. The environment would then be used to test parameters within the identified ranges both qualitatively and quantitatively by displaying both the numerical fidelity of the results and the changes to the image of the input scene at each stage of processing throughout the c/d/c system model.

2.2.11 Reconstructed output as 3 separate components

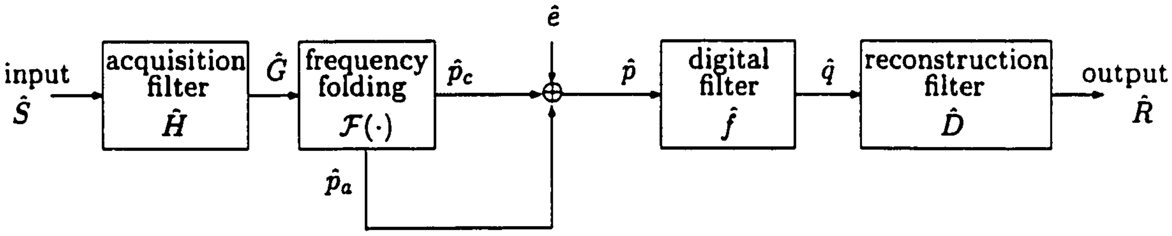


Figure 2.7: Split Output from Sampling Box

Figure 2.7 illustrates that, in the frequency domain, the output from the sampling box can be separated into two components:

1. the cascaded component, $\hat{p}_c[\nu_1, \nu_2] = \hat{G}[\nu_1, \nu_2]$;

2. the error attributable to aliasing,

$$\hat{p}_a[\nu_1, \nu_2] = \sum\sum_{[k_1,k_2]\neq[0,0]} \hat{G}[\nu_1 - k_1N_1,\ \nu_2 - k_2N_2].$$

This is true because, referring back to Figure 2.1,

$$\begin{aligned}\hat{p}'[\nu_1,\nu_2] &= \sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} \hat{G}[\nu_1 - k_1N_1,\, \nu_2 - k_2N_2] \\ &= \hat{G}[\nu_1,\nu_2] + \sum\sum_{[k_1,k_2]\neq[0,0]} \hat{G}[\nu_1 - k_1N_1,\, \nu_2 - k_2N_2]\end{aligned}$$

The aperiodic aliased component, $\hat{p}_a$, can be calculated in the frequency domain by subtracting the cascaded component, $\hat{p}_c$, which is analogous to $\hat{G}$, from the composite image, $\hat{p}'$, which is periodic with period $N_1 \times N_2$.

$$\begin{aligned}\hat{p}_a[\nu_1,\nu_2] &= \hat{p}'[\nu_1,\nu_2] - \hat{p}_c[\nu_1,\nu_2] \\ &= \sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} \hat{G}[\nu_1 - k_1N_1,\, \nu_2 - k_2N_2] - \hat{G}[\nu_1,\nu_2]\end{aligned}$$

Although $\hat{p}_a$ is aperiodic, the representation passband parameters that band-limit $\hat{p}_c$, similarly band-limit the coefficients of $\hat{p}_a$ that are affected by $\hat{p}_c$. Beyond the representation passband parameters, $\hat{p}_a[\nu_1,\nu_2] = \hat{p}'[\nu_1,\nu_2]$. During reconstruction, the infinite representation is band-limited by the reconstruction passband parameters. The components $\hat{p}_c$ and $\hat{p}_a$ can be summed with the stochastic noise to obtain $\hat{p}$, as illustrated in Figure 2.7.

Figure 2.8 indicates that the same result will be obtained whether the three components are added and then passed through $\hat{f}$ and $\hat{D}$ or whether each component is first passed through $\hat{f}$ and $\hat{D}$ and the results then summed. The Fourier matrix representation of the reconstructed image is the sum of the three image components $\hat{R}_c$, $\hat{R}_e$ and $\hat{R}_a$, each of which can be calculated and analyzed independently, i.e.,

$$\hat{R}[\nu_1,\nu_2] = \hat{R}_c[\nu_1,\nu_2] + \hat{R}_e[\nu_1,\nu_2] + \hat{R}_a[\nu_1,\nu_2]$$

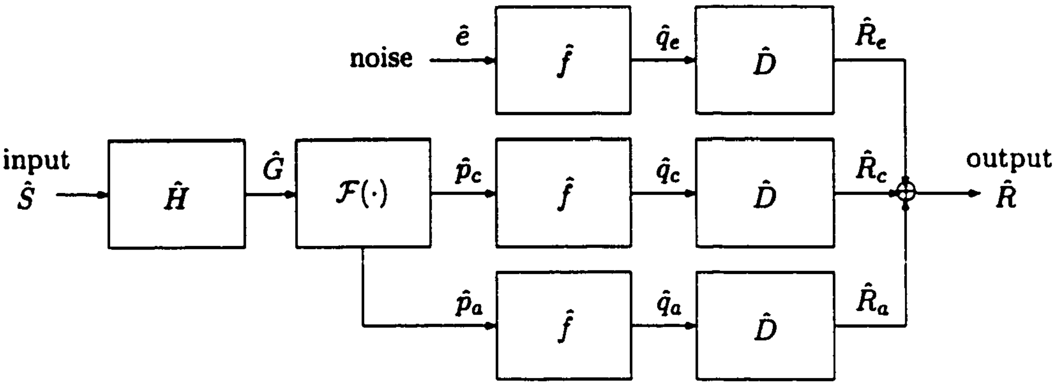


Figure 2.8: Cascaded, aliased and noise components.

where the *cascaded* component is

$$\hat{R}_c[\nu_1,\nu_2] = \hat{S}[\nu_1,\nu_2]\hat{H}(\nu_1/P_1,\nu_2/P_2)\hat{f}(\nu_1/N_1,\nu_2/N_2)\hat{D}(\nu_1/P_1,\nu_2/P_2),$$

the *noise* component is

$$\hat{R}_e[\nu_1,\nu_2] = \hat{e}[\nu_1,\nu_2]\hat{f}(\nu_1/N_1,\nu_2/N_2)\hat{D}(\nu_1/P_1,\nu_2/P_2),$$

and the *aliased* component is

$$\begin{aligned}\hat{R}_a[\nu_1,\nu_2] &= \hat{f}(\nu_1/N_1,\nu_2/N_2)\hat{D}(\nu_1/P_1,\nu_2/P_2) \\ &\quad \sum\sum_{[k_1,k_2]\neq[0,0]} \left(\hat{S}[\nu_1 - k_1N_1,\nu_2 - k_2N_2]\hat{H}(\nu_1/P_1 - k_1,\nu_2/P_2 - k_2)\right).\end{aligned}$$

Options in the environment allow the user to select any one of the three separate reconstructed components for display and analysis in the environment's main window as an alternative to the composite reconstructed output. These options enable the user to analyze qualitatively and quantitatively the *noise*, *cascaded* and *aliased* components, which sum to form the system output. It is possible to visualize the matrix representations of the individual components in either domain.

Chapter 3

The C/D/C Model — Implementation Level

In this chapter, Section 3.1 specifies the size and type of stored data structures. The four possible component modes available for selection by the user are each considered: composite, cascaded component, aliased component, and noise component. Section 3.2 discusses the algorithms required to process and display the matrix representations of an image for each component mode in both the spatial and frequency domains. Each algorithm is described in some detail and pseudo-code is provided.

3.1 Data Structures

The matrix representation of an input scene as it is processed through the c/d/c model can be displayed at each of the six model stages shown in Figure 3.1:

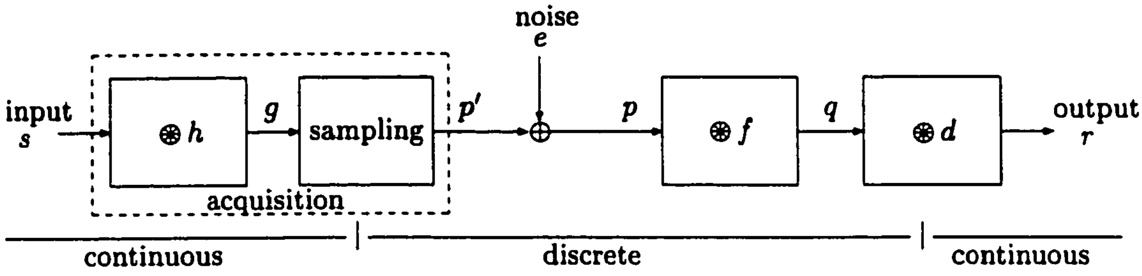


Figure 3.1: The six stages at which the image can be displayed.

1. input to acquisition filter (s);

2. output from acquisition filter/input to sampling box (g);

3. output from sampling box (p');

4. input to digital filter (p);

5. output from digital filter/input to reconstruction filter (q);

6. output from reconstruction filter (r).

As discussed in Chapter 2, to speed processing, only the Fourier transform associated with the matrix representation at each stage is stored.

3.1.1 Storage of displayed image

Recall that N_1 and N_2 are user-controlled system parameters which specify the number of samples per vertical and horizontal period respectively. The convention adopted in the environment is that the size of a displayed image can be user-controlled up to a maximum size of $8N_1 \times 8N_2$. The 8-bit brightness values used to define a digitized image are stored in a dynamically allocated array of characters, and the array is updated on demand for the purpose of image display. Dynamic storage allocation occurs in the function **image_display** on a call from either **get_raw_pic** or **manage_display_size_change**, and the dynamically allocated storage is freed by either **get_raw_pic** or **manage_display_size_change** before a reallocation occurs.

3.1.2 Storage of Fourier transforms

The dimensions of the frequency domain matrix representation of an input scene stored in the environment's library are $(M_1/2+1) \times M_2$, where $M_1, M_2 \in \{2^m : m = 4, 5, \ldots, 13\}$. Figure 3.2 (a) shows the dimensions and frequency indices of the full-size Fourier transform matrix representation which can be generated by complex conjugacy from the coefficient array stored in the environment's library; in Figure 3.2 (b), the representation passband parameters τ_1, τ_2 have been applied and the coefficients from the first array have been packed into a smaller array. Figure 3.2 illustrates the case $M_1/2 \geq \tau_1$ and $M_2/2 \geq \tau_2$. In

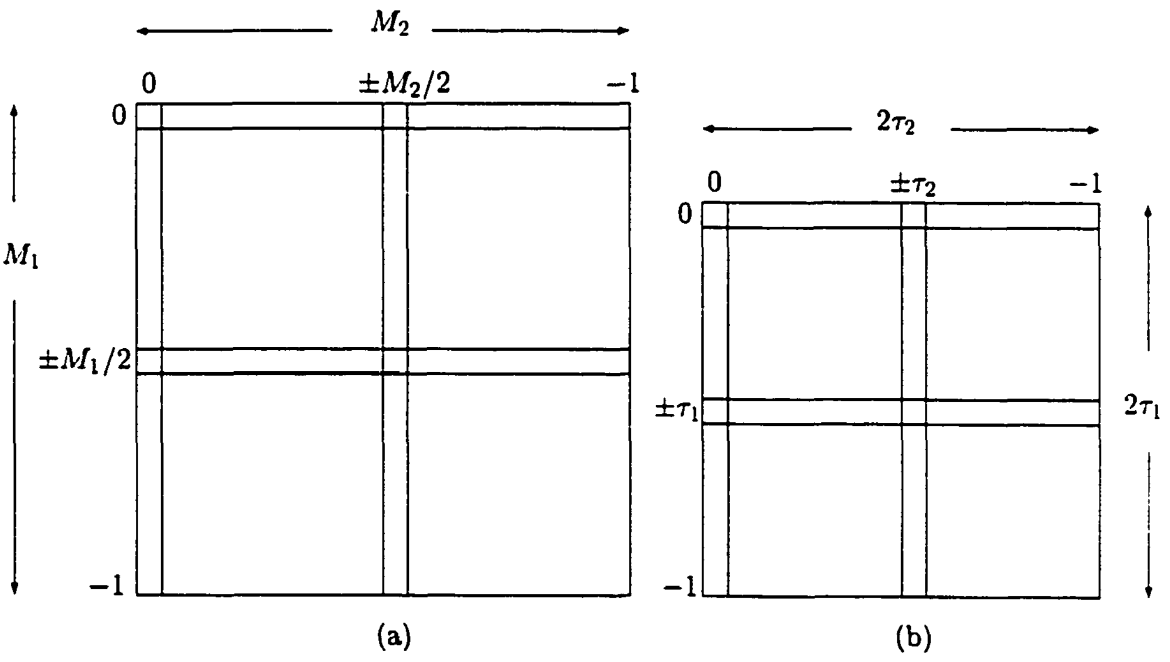


Figure 3.2: The packed transform data structure.

the case $M_1/2 < \tau_1$ and $M_2/2 < \tau_2$, all elements of the full-size Fourier transform coefficient array are transferred into the corners of a larger array which is zero-padded, as shown in Figure 3.3. Combinations of the two illustrated cases are also possible. The following should be noted.

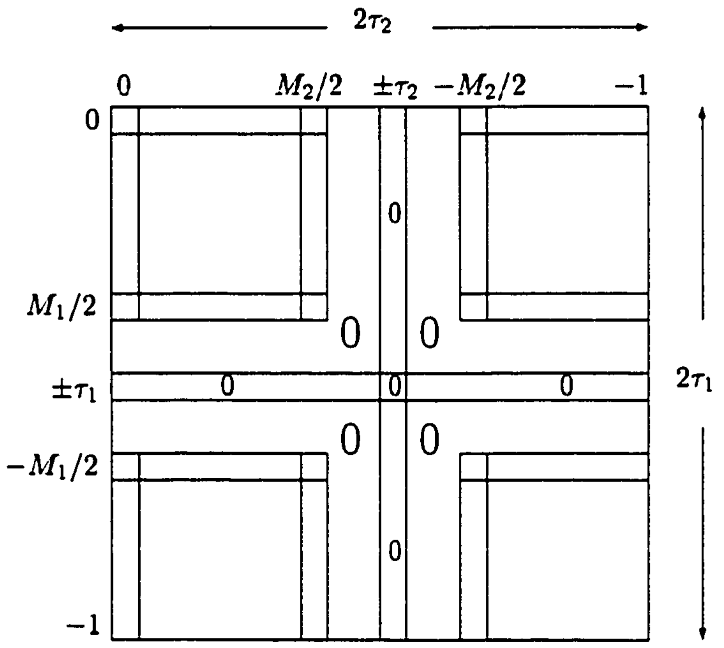


Figure 3.3: The filled transform data structure.

1. The stored data structure is not origin-centered.

2. Only the upper half of the data structure ($\nu_1 = 0, 1, \ldots, \tau_1$) is actually stored.

3. In Figure 3.2(b), the frequency index denoted as $\pm\tau_1$ contains the sum of the transform coefficients at $+\tau_1$ and $-\tau_1$ from Figure 3.2(a). If $M_1/2 = \tau_1$, the frequency index denoted as $\pm\tau_1$ contains the transform coefficients from the frequency index denoted as $\pm M_1/2$ in Figure 3.2(a).

4. In Figure 3.2(b), the frequency index denoted as $\pm\tau_2$ contains the sum of the transform coefficients at $+\tau_2$ and $-\tau_2$ from Figure 3.2(a). If $M_2/2 = \tau_2$, the frequency index denoted as $\pm\tau_2$ contains the transform coefficients from the frequency index denoted as $\pm M_2/2$ in Figure 3.2(a).

5. In Figure 3.2(b), each element of row $\pm\tau_1$ and each element of column $\pm\tau_2$ is divided by 2 after storage. In this way, the $[\pm\tau_1, \pm\tau_2]$ element is divided by 4.

6. In Figure 3.3, each element of rows $M_1/2$ and $-M_1/2$ and each element of columns $M_2/2$ and $-M_2/2$ is divided by 2 after storage. In this way, elements $[M_1/2. M_2/2]$, $[M_1/2, -M_2/2]$, $[-M_1/2, M_2/2]$, and $[-M_1/2, -M_2/2]$ are divided by 4.

3.1.2.1 input to acquisition filter

The user specifies the vertical (τ_1) and horizontal (τ_2) passband parameters of the matrix representation $\hat{S}$ corresponding to the input scene. Therefore, only $\hat{S}[\nu_1, \nu_2]$ coefficients for $[\nu_1, \nu_2] \in \mathcal{T}_1 \times \mathcal{T}_2$ are required for computation. However, because $\hat{S}[\nu_1, \nu_2]$ coefficients in the range $-1, -2, \ldots, -\tau_1$; $\nu_2 \in \mathcal{T}_2$ can be generated by conjugate symmetry. only $\hat{S}[\nu_1. \nu_2]$ coefficients in the range $0, 1, \ldots, \tau_1$; $\nu_2 \in \mathcal{T}_2$ are stored. The coefficients of $\hat{S}$ are stored in a dynamically allocated array of $(\tau_1 + 1) \times 2\tau_2$ variables of COMPLEX type.

3.1.2.2 acquisition filter

It should be noted that the acquisition filter is implemented in the environment as a separable filter. This requirement is not inherent to the c/d/c model or to the environment's design. The filter is so implemented to conserve memory (only a single row and column of the 2-D filter need be stored) and to facilitate the independent operation of the 1-D system.

The coefficients of the sampled Fourier transform $\hat{H}$ corresponding to the acquisition filter h are temporarily stored in a dynamically allocated array of $(\tau_1 + 1) \times 2\tau_2$ variables of COMPLEX type. The use of a larger data structure would be redundant due to the passband parameters imposed on the input scene. The matrix representation of $\hat{H}$ is generated by multiplying the stored values $\hat{H}_1(\omega_1)$ at $\omega_1 = \nu_1/P_1$, where $\nu_1 = 0, 1, 2, \ldots, \tau_1$, by the stored values $\hat{H}_2(\omega_2)$ at $\omega_2 = \nu_2/P_2$, where $\nu_2 \in \mathcal{T}_2$. The matrix representation of $\hat{H}$ is freed as soon as $\hat{G}$ has been computed.

3.1.2.3 output from acquisition filter/input to sampling box

The representation passband parameters limit the size of the matrix representation $\hat{G}$ which corresponds to the image at the output from the acquisition filter. Only $\hat{G}[\nu_1, \nu_2]$ coefficients for $[\nu_1, \nu_2] \in \mathcal{T}_1 \times \mathcal{T}_2$ are, therefore, required for computation. However, as for $\hat{S}$, only $\hat{G}[\nu_1, \nu_2]$ coefficients in the range $\nu_1 = 0, 1, \ldots, \tau_1$; $\nu_2 \in \mathcal{T}_2$ need be stored. The coefficients of $\hat{G}$ are stored in a dynamically allocated array of $(\tau_1 + 1) \times 2\tau_2$ variables of COMPLEX type.

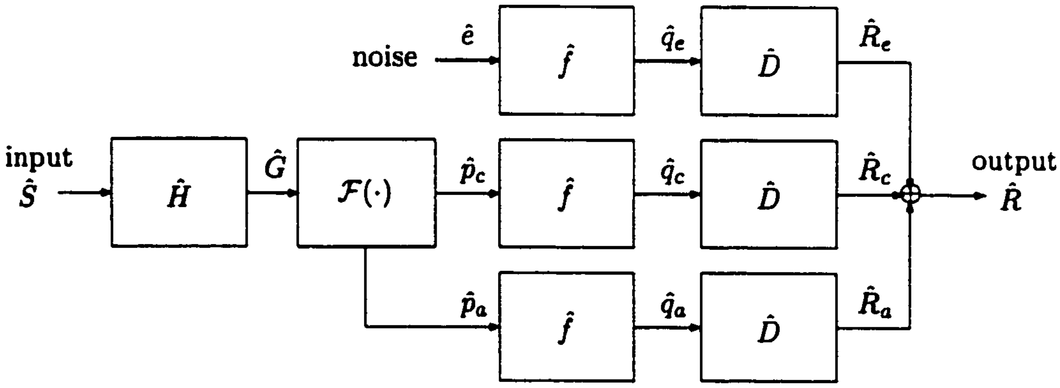


Figure 3.4: Cascaded, aliased and noise components.

3.1.2.4 output from sampling box

The matrix representation of the output from the sampling box is stored in one of three possible forms.

composite output

The matrix representation $\hat{p}'$, corresponding to the composite output from the sampling box shown in Figure 3.1, is periodic with vertical period N_1 and horizontal period N_2. ($\hat{p}' = \hat{p}_c + \hat{p}_a$, as shown in Figure 3.4, and $\hat{p}' + \hat{e} = \hat{p}$, as shown in Figure 3.1.) Therefore, only $\hat{p}'[\nu_1, \nu_2]$ coefficients for $[\nu_1, \nu_2] \in \mathcal{N}_1 \times \mathcal{N}_2$ are required for computation. The coefficients of $\hat{p}'$ are stored in a dynamically allocated array of $N_1 \times N_2$ variables of COMPLEX type.

cascaded component

The matrix representation $\hat{p}_c$, corresponding to the cascaded component from the sampling box shown in Figure 3.4, is analogous to $\hat{G}$. The coefficients of $\hat{p}_c$ are stored in a dynamically allocated array of $(\tau_1 + 1) \times 2\tau_2$ variables of COMPLEX type.

aliased component

The matrix representation $\hat{p}_a$, corresponding to the aliased component from the sampling box shown in Figure 3.4, represents the error attributable to frequency folding. See Section 2.2.11. The coefficients of $\hat{p}_a$ that cannot be generated by replicating the $N_1 \times N_2$ composite output from the sampling box ($\hat{p}'$) are stored in a dynamically allocated array of $(\tau_1 + 1) \times 2\tau_2$ variables of COMPLEX type. Any subsequent increase in matrix representation size required for processing (e.g., a reconstruction passband parameter greater than the corresponding representation passband parameter) can be achieved by padding with replications of the $N_1 \times N_2$ matrix representation of the composite output from the sampling box at the output from the digital filter, because all cascaded coefficients beyond the representation passband parameters must be zero.

The notation for the composite output from the sampling box differs from that used for the cascaded and aliased components because the composite output from the sampling box is modified by additive noise before input to the digital filter whereas, in the case of the components, the output from the sampling box is the input to the digital filter.

3.1.2.5 additive noise

The matrix representation $\hat{e}$, corresponding to the additive noise, is periodic with vertical period N_1 and horizontal period N_2. Coefficients in the range $[\nu_1, \nu_2] \in \mathcal{N}_1 \times \mathcal{N}_2$ are required for computation, but there is no need to store the data structure as the (pseudo-)random noise matrix representation is generated each time noise subsystem processing occurs. The coefficients of $\hat{e}$ are temporarily stored in a dynamically allocated array of $N_1 \times N_2$ variables of COMPLEX type.

3.1.2.6 input to digital filter

The matrix representation of the input to the digital filter is stored in one of four possible forms.

composite output

The matrix representation $\hat{p}$, corresponding to the input to the digital filter, is periodic with vertical period N_1 and horizontal period N_2. Coefficients of $\hat{p}[\nu_1, \nu_2]$ for $[\nu_1, \nu_2] \in \mathcal{N}_1 \times \mathcal{N}_2$ are required for computation and are consequently stored. The coefficients of $\hat{p}$ are stored in a dynamically allocated array of $N_1 \times N_2$ variables of COMPLEX type.

cascaded component

The matrix representation $\hat{p}_c$, the cascaded component input to the digital filter, is analogous to the cascaded component output from the sampling box. The coefficients of $\hat{p}_c$ are stored in a dynamically allocated array of $(\tau_1 + 1) \times 2\tau_2$ variables of COMPLEX type.

aliased component

The matrix representation $\hat{p}_a$, the aliased component input to the digital filter, is analogous to the aliased component output from the sampling box. The coefficients of $\hat{p}_a$ are stored in a dynamically allocated array of $(\tau_1 + 1) \times 2\tau_2$ variables of COMPLEX type.

noise component

The matrix representation $\hat{e}$ is the additive noise component input to the digital filter. The coefficients of $\hat{e}$ are stored in a dynamically allocated array of $N_1 \times N_2$ variables of COMPLEX type.

3.1.2.7 digital filter

It should be noted that the digital filter is implemented in the environment as a separable filter. This requirement is not inherent to the c/d/c model or to the environment's design. The filter is so implemented to conserve memory (only a single row and column of the 2-D filter need be stored) and to facilitate the independent operation of the 1-D system.

The coefficients of the sampled Fourier transform $\hat{f}$, periodic with period $N_1 \times N_2$. corresponding to the digital filter f are temporarily stored in a dynamically allocated array of $N_1 \times N_2$ variables of COMPLEX type. The matrix representation of $\hat{f}$ is generated by multiplying the stored values $\hat{f}_1(\omega_1)$ at $\omega_1 = \nu_1/N_1$, where $\nu_1 \in \mathcal{N}_1$, by the stored values

$\hat{f}_2(\omega_2)$ at $\omega_2 = \nu_2/N_2$, where $\nu_2 \in \mathcal{N}_2$. The matrix representation of $\hat{f}$ is freed as soon as $\hat{q}$ has been computed.

3.1.2.8 output from digital filter/input to reconstruction filter

The matrix representation of the input to the reconstruction filter is stored in one of four possible forms.

composite output from the digital filter

The matrix representation $\hat{q}$, corresponding to the output from the digital filter, is periodic with period $N_1 \times N_2$. Coefficients of $\hat{q}[\nu_1, \nu_2]$ for $[\nu_1 . \nu_2] \in \mathcal{N}_1 \times \mathcal{N}_2$ are required for computation and are consequently stored. The coefficients of $\hat{q}$ are stored in a dynamically allocated array of $N_1 \times N_2$ variables of COMPLEX type.

cascaded component

The matrix representation $\hat{q}_c$ corresponds to the cascaded component output from the digital filter. The coefficients of $\hat{q}_c$ are stored in a dynamically allocated array of $(\tau_1 + 1) \times 2\tau_2$ variables of COMPLEX type.

aliased component

The matrix representation $\hat{q}_a$ corresponds to the aliased component output from the digital filter. The coefficients of $\hat{q}_a$ are stored in a dynamically allocated array of $(\tau_1 + 1) \times 2\tau_2$ variables of COMPLEX type.

noise component

The matrix representation $\hat{q}_e$, corresponding to the noise component output from the digital filter, is periodic with vertical period N_1 and horizontal period N_2. Coefficients of $\hat{q}_e[\nu_1, \nu_2]$ for $[\nu_1, \nu_2] \in \mathcal{N}_1 \times \mathcal{N}_2$ are required for computation and are consequently stored. The coefficients of $\hat{q}_e$ are stored in a dynamically allocated array of $N_1 \times N_2$ variables of COMPLEX type.

3.1.2.9 reconstruction filter

It should be noted that the reconstruction filter is implemented in the environment as a separable filter. This requirement is not inherent to the c/d/c model or to the environment's design. The filter is so implemented to conserve memory (only a single row and column of the 2-D filter need be stored) and to facilitate the independent operation of the 1-D system.

Reconstruction is affected by the vertical (τ_{r_1}) and horizontal (τ_{r_2}) reconstruction passband parameters applied to generate the matrix representation $\hat{R}$ corresponding to the reconstructed image. The coefficients of the sampled Fourier transform $\hat{D}$ corresponding to the reconstruction filter d are temporarily stored in a dynamically allocated array of $(\tau_{r_1} + 1) \times 2\tau_{r_2}$ variables of COMPLEX type. The use of a larger data structure would be redundant due to the passband parameters imposed on the reconstructed image. The matrix representation of $\hat{D}$ is generated by multiplying the stored values $\hat{D}_1(\omega_1)$ at $\omega_1 = \nu_1/P_1$, where $\nu_1 = 0, 1, 2, \ldots, \tau_{r_1}$, by the stored values $\hat{D}_2(\omega_2)$ at $\omega_2 = \nu_2/P_2$, where $\nu_2 \in \mathcal{T}_{r_2}$. The matrix representation of $\hat{D}$ is freed as soon as $\hat{R}$ has been computed.

3.1.2.10 output from reconstruction filter

The matrix representation $\hat{R}$ corresponds to the output from the reconstruction filter. Only $\hat{R}[\nu_1, \nu_2]$ coefficients for $[\nu_1, \nu_2] \in \mathcal{T}_{r_1} \times \mathcal{T}_{r_2}$ are required for computation. However, because $\hat{R}$ is analogous to $\hat{S}$ and $\hat{G}$, only $\hat{R}[\nu_1, \nu_2]$ coefficients in the range $\nu_1 = 0, 1, \ldots, \tau_{r_1}$; $\nu_2 \in \mathcal{T}_{r_2}$ need be stored. The coefficients of $\hat{R}$ are stored in a dynamically allocated array of

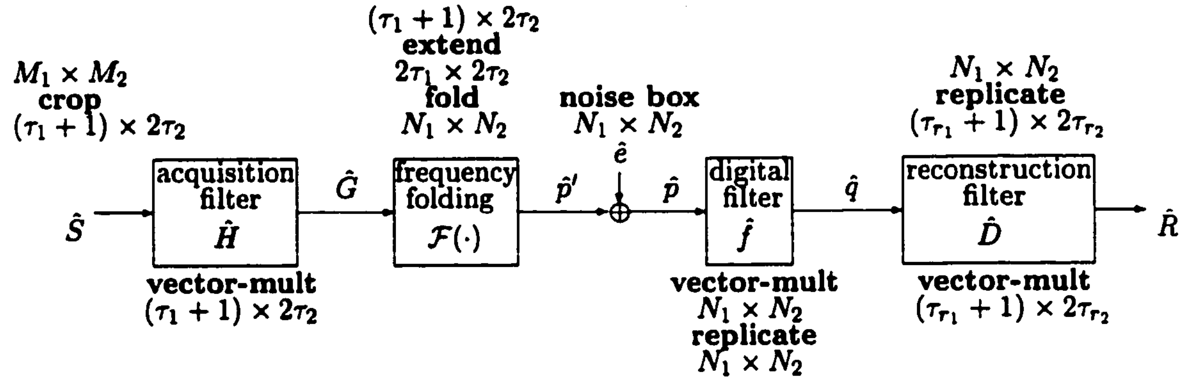
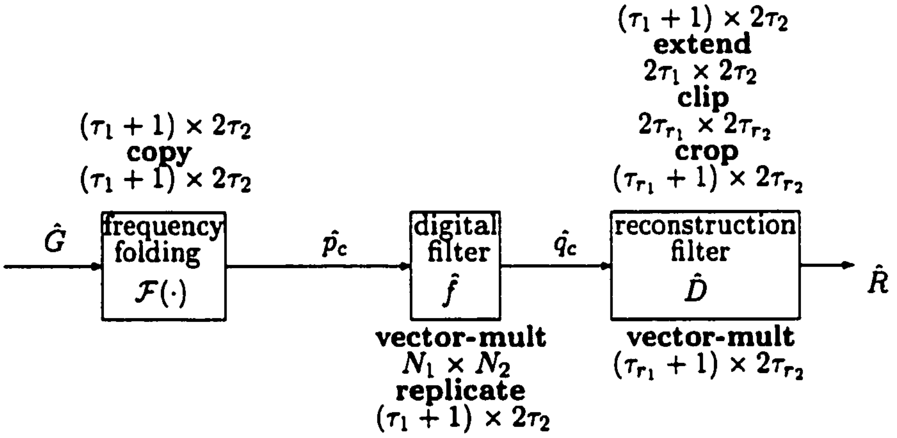


Figure 3.5: Algorithms used in transform processing (composite output from sampling box).



$(\tau_1 + 1) \times 2\tau_2$
copy
$(\tau_1 + 1) \times 2\tau_2$
$(\tau_1 + 1) \times 2\tau_2$
extend
$2\tau_1 \times 2\tau_2$
clip
$2\tau_{r_1} \times 2\tau_{r_2}$
crop
$(\tau_{r_1} + 1) \times 2\tau_{r_2}$
$\hat{G}$
frequency folding
$\mathcal{F}(\cdot)$
$\hat{p}_c$
digital filter
$\hat{f}$
$\hat{q}_c$
reconstruction filter
$\hat{D}$
$\hat{R}$
vector-mult
$N_1 \times N_2$
replicate
$(\tau_1 + 1) \times 2\tau_2$
vector-mult
$(\tau_{r_1} + 1) \times 2\tau_{r_2}$

Figure 3.6: Algorithms used in transform processing (cascaded output from sampling box).

$(\tau_{r_1} + 1) \times 2\tau_{r_2}$ variables of COMPLEX type.

3.2 Algorithms

Figures 3.5 to 3.8 are annotated with the names of the algorithms used to create and manipulate the Fourier transform data structures. The input dimensions are shown above and the output dimensions are shown below the algorithm names. The algorithms are:

– copy – Section 3.2.2;

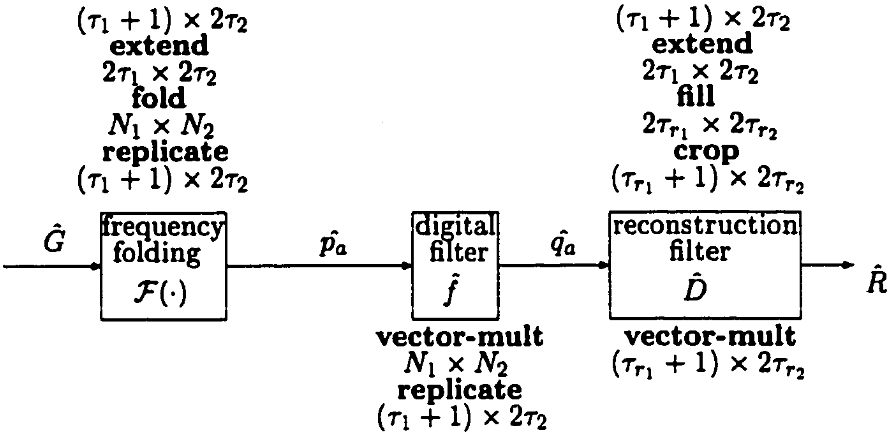


Figure 3.7: Algorithms used in transform processing (aliased output from sampling box).

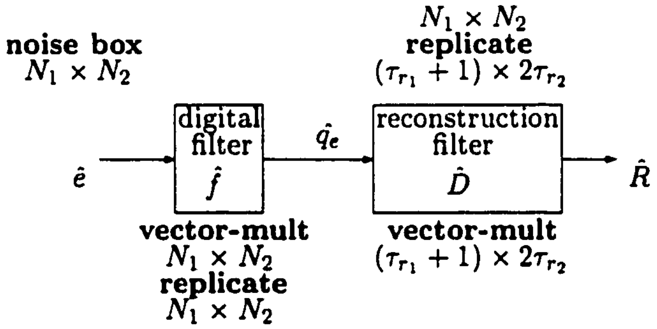


Figure 3.8: Algorithms used in noise transform processing.

- crop - Section 3.2.3;
- extend - Section 3.2.4;
- fill (and/or clip) - Section 3.2.5;
- frequency fold (fold) - Section 3.2.6;
- noise - Section 3.2.8;
- replicate (and/or clip) - Section 3.2.10;
- vector multiplication (vector-mult) - Section 3.2.11.

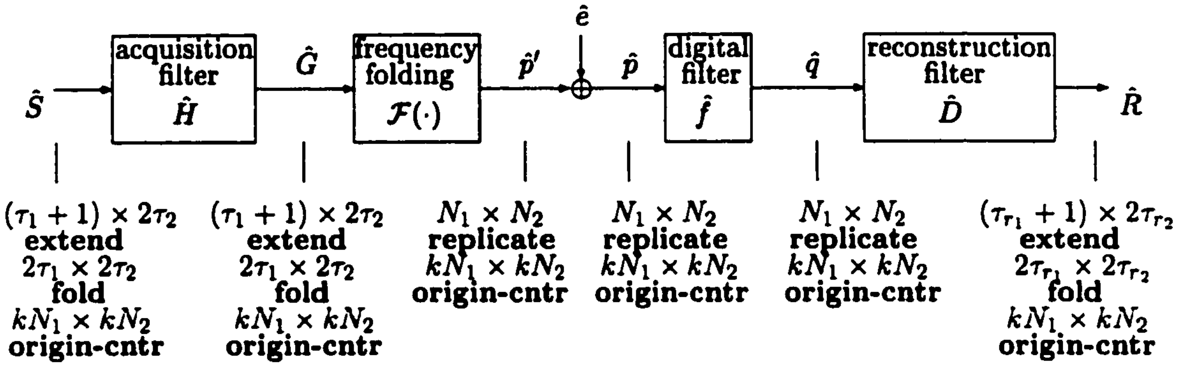

$\hat{S}$
acquisition filter $\hat{H}$
$\hat{G}$
frequency folding $\mathcal{F}(\cdot)$
$\hat{p}'$
$\hat{e}$
$\hat{p}$
digital filter $\hat{f}$
$\hat{q}$
reconstruction filter $\hat{D}$
$\hat{R}$
$(\tau_1+1) \times 2\tau_2$ extend $2\tau_1 \times 2\tau_2$ fold $kN_1 \times kN_2$ origin-cntr
$(\tau_1+1) \times 2\tau_2$ extend $2\tau_1 \times 2\tau_2$ fold $kN_1 \times kN_2$ origin-cntr
$N_1 \times N_2$ replicate $kN_1 \times kN_2$ origin-cntr
$N_1 \times N_2$ replicate $kN_1 \times kN_2$ origin-cntr
$N_1 \times N_2$ replicate $kN_1 \times kN_2$ origin-cntr
$(\tau_{r_1}+1) \times 2\tau_{r_2}$ extend $2\tau_{r_1} \times 2\tau_{r_2}$ fold $kN_1 \times kN_2$ origin-cntr

Figure 3.9: Algorithms used in frequency domain display (composite output from sampling box).

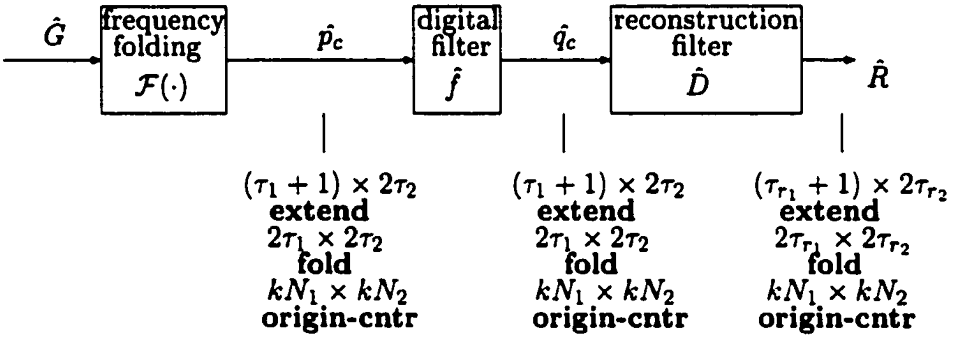

$\hat{G}$
frequency folding $\mathcal{F}(\cdot)$
$\hat{p}_c$
digital filter $\hat{f}$
$\hat{q}_c$
reconstruction filter $\hat{D}$
$\hat{R}$
$(\tau_1+1) \times 2\tau_2$ extend $2\tau_1 \times 2\tau_2$ fold $kN_1 \times kN_2$ origin-cntr
$(\tau_1+1) \times 2\tau_2$ extend $2\tau_1 \times 2\tau_2$ fold $kN_1 \times kN_2$ origin-cntr
$(\tau_{r_1}+1) \times 2\tau_{r_2}$ extend $2\tau_{r_1} \times 2\tau_{r_2}$ fold $kN_1 \times kN_2$ origin-cntr

Figure 3.10: Algorithms used in frequency domain display (cascaded output from sampling box).

Figures 3.9 to 3.12 are annotated with the names of the algorithms used to display the contents of the Fourier transform data structures in the frequency domain. As in Figures 3.5 to 3.8, input and output data structure dimensions are shown respectively above and below the algorithm names. The algorithms are:

- extend - Section 3.2.4;
- fill (and/or clip) - Section 3.2.5;
- frequency fold (fold) - Section 3.2.6;
- origin center (origin-cntr) - Section 3.2.9;
- replicate (and/or clip) - Section 3.2.10.

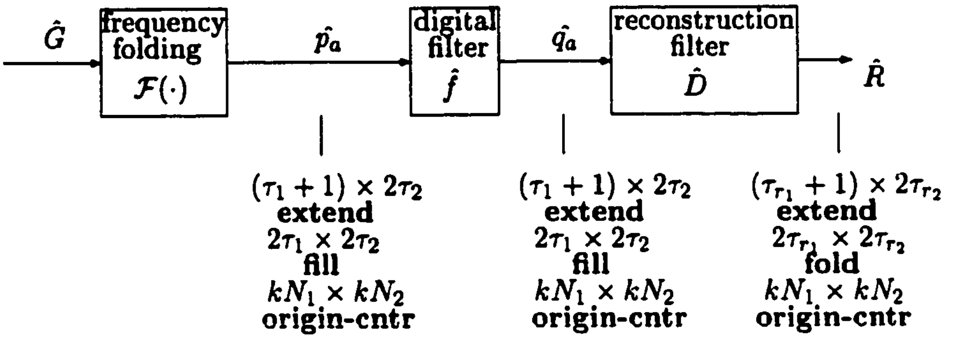


Figure 3.11: Algorithms used in frequency domain display (aliased output from sampling box).

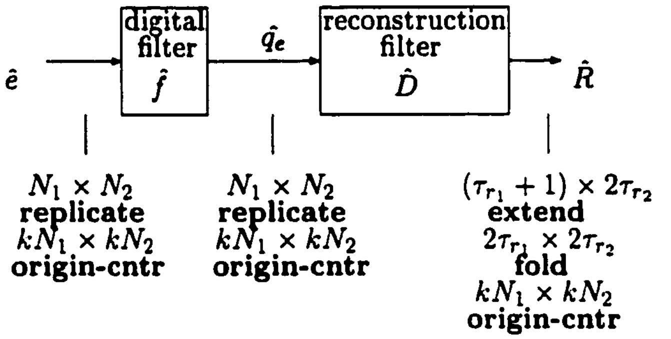
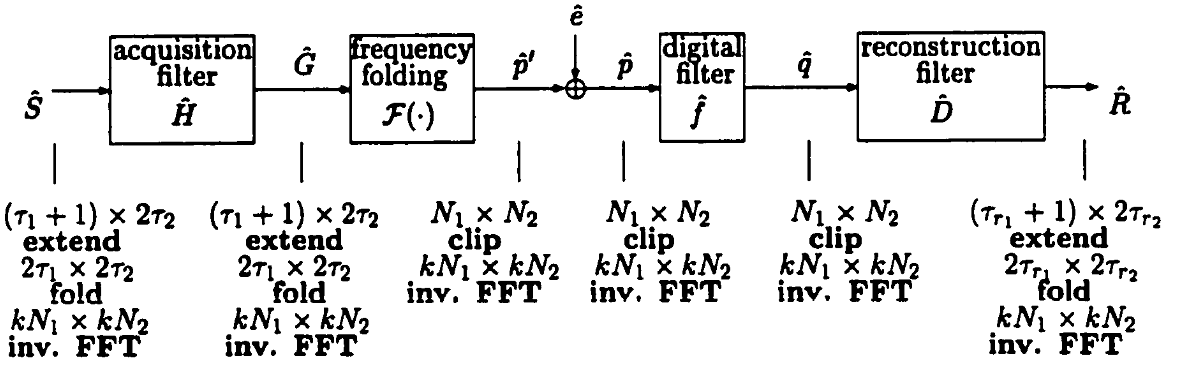


Figure 3.12: Algorithms used in frequency domain display of noise component.

Figures 3.13 to 3.16 are annotated with the names of the algorithms used to display the contents of the Fourier transform data structures in the spatial domain. Again, input and output data structure dimensions are shown respectively above and below the algorithm names. The algorithms are:

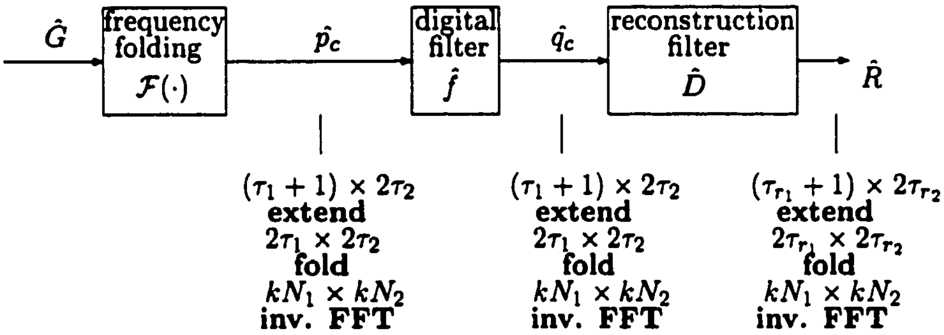
- clip (and/or zero pad) - Section 3.2.1;
- extend - Section 3.2.4;
- frequency fold (fold) - Section 3.2.6;
- inverse Fast Fourier Transform (inv. FFT) - Section 3.2.7.

In all discussions which follow, the terms "cropped matrix representation" and "cropped data structure" refer to a matrix representation in which the coefficients that can be gener-



$\hat{S}$
acquisition filter $\hat{H}$
$\hat{G}$
frequency folding $\mathcal{F}(\cdot)$
$\hat{p}'$
$\hat{e}$
$\hat{p}$
digital filter $\hat{f}$
$\hat{q}$
reconstruction filter $\hat{D}$
$\hat{R}$
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ fold $kN_1\times kN_2$ inv. FFT
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ fold $kN_1\times kN_2$ inv. FFT
$N_1\times N_2$ clip $kN_1\times kN_2$ inv. FFT
$N_1\times N_2$ clip $kN_1\times kN_2$ inv. FFT
$N_1\times N_2$ clip $kN_1\times kN_2$ inv. FFT
$(\tau_{r_1}+1)\times 2\tau_{r_2}$ extend $2\tau_{r_1}\times 2\tau_{r_2}$ fold $kN_1\times kN_2$ inv. FFT

Figure 3.13: Algorithms used in spatial domain display (composite output from sampling box).



$\hat{G}$
frequency folding $\mathcal{F}(\cdot)$
$\hat{p}_c$
digital filter $\hat{f}$
$\hat{q}_c$
reconstruction filter $\hat{D}$
$\hat{R}$
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ fold $kN_1\times kN_2$ inv. FFT
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ fold $kN_1\times kN_2$ inv. FFT
$(\tau_{r_1}+1)\times 2\tau_{r_2}$ extend $2\tau_{r_1}\times 2\tau_{r_2}$ fold $kN_1\times kN_2$ inv. FFT

Figure 3.14: Algorithms used in spatial domain display (cascaded output from sampling box).

ated using complex conjugacy are not included. The terms "full-size matrix representation" and "full-size data structure" refer to a matrix representation that is not cropped.

3.2.1 Clip

The **clip algorithm** creates a new full-size data structure and stores in it the resized contents of an existing full-size data structure that holds a periodic matrix representation. This is accomplished by clipping or zero padding, as appropriate.

parameters:

- a pointer to an array of type COMPLEX, which contains a full-size, periodic matrix representation;

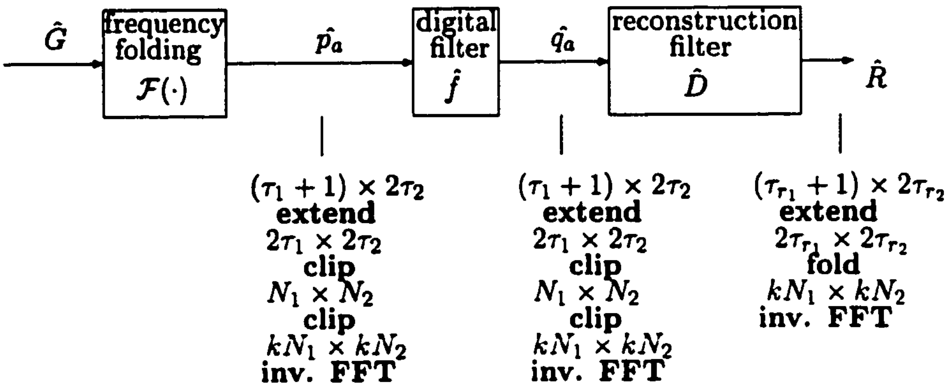


Figure 3.15: Algorithms used in spatial domain display (aliased output from sampling box).

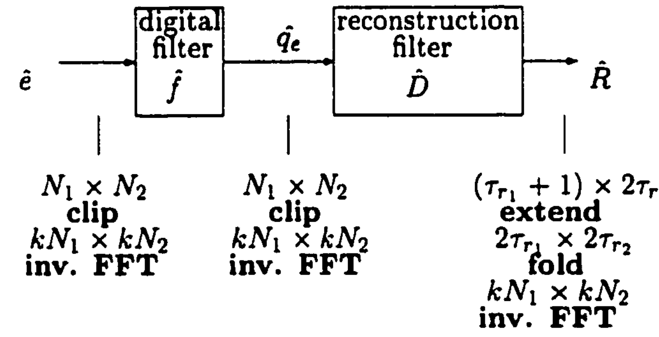


Figure 3.16: Algorithms used in spatial domain display of noise component.

- the dimensions of the input array;

- the dimensions of the outp'it array.

pseudo-code:

```
COMPLEX *clip(S, M1_from, M2_from, M1_to, M2_to)
{
COMPLEX *clipped_S;
int     m,n;
  clipped_S = allocate(M1_to*M2_to,sizeof(COMPLEX));
  for (m=0; m<M1_to/2; m++)
     if (m >= M1_from/2)
       for (n=0; n<M2_to; n++)
          clipped_S[m*M2_to+n] = 0;
     else
     {
      for (n=0; n<M2_to/2; n++)
         if (n >= M2_from/2)
```

```
           clipped_S[m*M2_to+n] = 0;
         else
           clipped_S[m*M2_to+n] = S[m*M2_from+n];
      for (n=1; n<=M2_to/2; n++)
         if (n > M2_from/2)
           clipped_S[(m+1)*M2_to-n] = 0;
         else
           clipped_S[(m+1)*M2_to-n] = S[(m+1)*M2_from-n];
     }
  for (m=1; m<=M1_to/2; m++)
     if (m > M1_from/2)
       for (n=0; n<M2_to; n++)
          clipped_S[(M1_to-m)*M2_to+n] = 0;
     else
     {
      for (n=0; n<M2_to/2; n++)
         if (n >= M2_from/2)
           clipped_S[(M1_to-m)*M2_to+n] = 0;
         else
           clipped_S[(M1_to-m)*M2_to+n] = S[(M1_from-m)*M2_from+n];
      for (n=1; n<=M2_to/2; n++)
         if (n > M2_from/2)
           clipped_S[(M1_to-m+1)*M2_to-n] = 0;
         else
           clipped_S[(M1_to-m+1)*M2_to-n] = S[(M1_from-m+1)*M2_from-n];
     }
  return (clipped_S);
}
```

3.2.2 Copy

The **copy** algorithm creates a new data structure and transfers into it the contents of an existing data structure of the same size.

parameters:

- a pointer to an array of type COMPLEX, which contains a matrix representation;
- the dimensions of the array.

pseudo-code:

```
COMPLEX *copy(G, M1, M2)
{
COMPLEX *copied_G;
int     n1,n2;
  copied_G = allocate(M1*M2,sizeof(COMPLEX));
  for (n1=0; n1<M1; n1++)
    for (n2=0; n2<M2; n2++)
       copied_G[n1*M1+n2] = G[n1*M1+n2];
  return(copied_G);
}
```

3.2.3 Crop

The **crop** algorithm creates a new data structure and stores in it a clipped and/or zero-padded matrix representation formed from an existing full-size data structure. according to the specifications of the passband parameters and discarding that part of the data structure which can be generated using complex conjugacy. This is accomplished by:

1. discarding any coefficients beyond the frequency coordinates $[\tau_1, 2\tau_2 - 1]$;
2. zero-padding the array. if $\tau_1 > M_1/2$ and/or $\tau_2 > M_2/2$;
3. summing column $(M_2 - \tau_2)$ into column τ_2, if $\tau_2 < M_2/2$;
4. summing row $(M_1 - \tau_1)$ into row τ_1, if $\tau_1 < M_1/2$;
5. dividing column $(M_2 - \tau_2)$ by 2, if $\tau_2 \leq M_2/2$;
6. dividing row $(M_1 - \tau_1)$ by 2, if $\tau_1 \leq M_1/2$;
7. dividing columns (τ_2) and $(M_2 - \tau_2)$ by 2, if $\tau_2 > M_2/2$;
8. dividing rows (τ_1) and $(M_1 - \tau_1)$ by 2, if $\tau_1 > M_1/2$.

parameters:

- a pointer to an array of type COMPLEX, which contains a full-size matrix representation;
- the dimensions of the input array;
- the dimensions of the output array.

pseudo-code:

```
COMPLEX *crop(S, M1_from, M2_from, M1_to, M2_to)
{
COMPLEX *cropped_S;
int m,n;
  cropped_S = allocate(M1_to*M2_to,sizeof(COMPLEX));
  for (m=0; m<M1_to; m++)
  {
   if (m > M1_from/2)
   {
    for (n=0; n<M2_to; n++)
       cropped_S[m*M2_to+n] = 0;
    if (m == (M1_to-1))
      for (n=0; n<M2_to; n++)
         cropped_S[M1_from/2*M2_to+n] /= 2;
   }
   else
   {
    for (n=0; n<=M2_to/2; n++)
       if (n > M2_from/2)
       {
        cropped_S[m*M2_to+n] = 0;
        if (n == (M2_to/2))
          cropped_S[m*M2_to+M2_from/2] /= 2;
       }
       else
       {
        cropped_S[m*M2_to+n] = S[m*M2_from+n];
        if ((m == (M1_to-1)) && ((M1_from/2) > (M1_to-1)))
          cropped[m*M2_to+n] += S[(M1_from-m)*M2_from+n];
       }
    for (n=1; n<M2_to/2; n++)
       if (n > M2_from/2)
       {
        cropped_S[(m+1)*M2_to-n] = 0;
        if (n == (M2_to/2-1))
          cropped_S[(m+1)*M2_to-M2_from/2] /= 2;
       }
       else
       {
        cropped_S[(m+1)*M2_to-n] = S[(m+1)*M2_from-n];
        if ((m == (M1_to-1)) && ((M1_from/2) > (M1_to-1)))
          cropped_S[(m+1)*M2_to-n] += S[(M1_from-m)*M2_from+n];
       }
    if (M2_to/2 < M2_from/2)
      cropped_S[(m+1)*M2_to-n] += S[(m+1)*M2_from-n];
   }
  }
  for (n=0; n<M2; n++)
     cropped_S[(M1-1)*M2+n] /= 2;
  for (m=0; m<M1; m++)
     cropped_S[m*M2 + M2/2] /= 2;
  return (cropped_S);
}
```

3.2.4 Extend

The **extend** algorithm creates a new data structure and stores in it a full-size matrix representation created from an existing data structure that contains a cropped matrix representation by generating the coefficients for $-\tau_1 < \nu_1 < 0$ using complex conjugacy to reconstruct a $2\tau_1 \times 2\tau_2$ data structure from the stored $(\tau_1 + 1) \times 2\tau_2$ array.

parameters:

- a pointer to an array of type COMPLEX, which contains a cropped matrix representation;
- the dimensions of the input array;

- the dimensions of the output array.

pseudo-code:

```
COMPLEX *extend(F, M1_from, M2_from, M1_to, M2_to)
{
COMPLEX *F2D;
int     n1,n2,n1_pr;
  F2D = allocate(M1_to*M2_to,sizeof(COMPLEX));
  for (n1=0; n1<=M1_to/2; n1++)
  {
   for (n2=0; n2<=M2_to/2; n2++)
      F2D[n1*M2_to+n2] = F[n1*M2_from+n2];
   for (n2=1; n2<M2_to/2; n2++)
      F2D[(n1+1)*M2_to-n2] = F[(n1+1)*M2_from-n2];
  }
  for (n1=1, n1_pr=(M1_to-1); n1<M1_to/2; n1++, n1_pr--)
  {
   F2D[n1_pr*M2_to] = F2D[n1*M2_to];
   for (n2=1; n2<M2_to; n2++)
      F2D[n1_pr*M2_to+M2_to-n2] = F2D[n1*M2_to+n2];
  }
  return F2D;
}
```

3.2.5 Fill

In the discussion which follows, the term "filler data structure" refers to the $N_1 \times N_2$ matrix representation of a composite image formed from frequency folding. The **fill** algorithm creates a new full-size data structure, copies to it the contents of an existing data structure containing a full-size matrix representation and either clips it or pads it with the contents of an $N_1 \times N_2$ filler data structure.

parameters:

- a pointer to an array of type COMPLEX, which contains a full-size matrix representation;

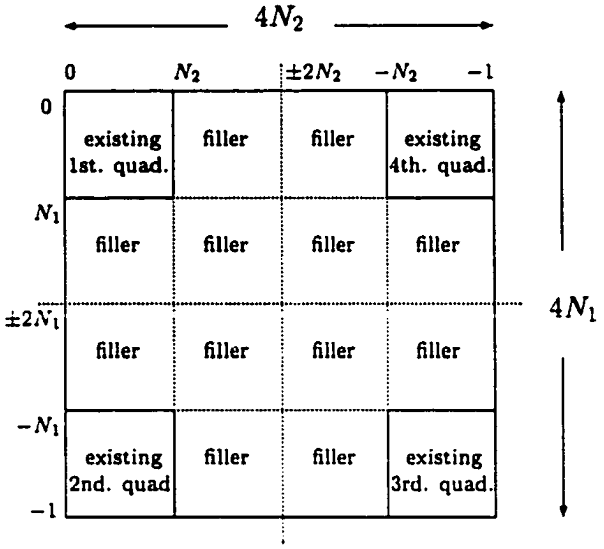


Figure 3.17: The filled data structure for spatial domain display ($\tau = N$).

- a pointer to an array of type COMPLEX, which contains a filler data structure;
- the dimensions of the input array;
- the dimensions of the output array.

pseudo-code:

```
COMPLEX *fill(S, filler, M1_from, M2_from, M1_to, M2_to)
{
COMPLEX *filled_S;
int     m,n,x,y;
  filled_S = allocate(M1_to*M2_to,sizeof(COMPLEX));
  for (m=0; m<M1_to/2; m++)
  {
   x = fmod(m,N1);
   if (m >= M1_from/2)
     for (n=0; n<M2_to; n++)
     {
      y = fmod(n,N2);
      filled_S[m*M2_to+n] = filler[x*N2+y];
     }
   else
   {
    for (n=0; n<M2_to/2; n++)
  {
   if (n >= M2_from/2)
   {
    y = fmod(n,N2);
    filled_S[m*M2_to+n] = filler[x*N2+y];
   }
   else
     filled_S[m*M2_to+n] = S[m*M2_from+n];
  }
  for (n=1; n<=M2_to/2; n++)
  {
   if (n > M2_from/2)
   {
    y = fmod(n,N2);
    filled_S[(m+1)*M2_to-n] = filler[(x+1)*N2-y];
   }
   else
     filled_S[(m+1)*M2_to-n] = S[(m+1)*M2_from-n];
  }
 }
}
for (m=1; m<=M1_to/2; m++)
{
 x = N1 - 1 - fmod((m-1),N1);
 if (m > M1_from/2)
 {
  for (n=0; n<M2_to; n++)
  {
   y = fmod(n,N2);
   filled_S[(M1_to-m)*M2_to+n] = filler[x*N2+y];
  }
 }
 else
 {
  for (n=0; n<M2_to/2; n++)
  {
   if (n >= M2_from/2)
   {
    y = fmod(n,N2);
    filled_S[(M1_to-m)*M2_to+n] = filler[x*N2+y];
   }
   else
     filled_S[(M1_to-m)*M2_to+n] = S[(M1_from-m)*M2_from+n];
  }
  for (n=1; n<=M2_to/2; n++)
```

```
      {
       if (n > M2_from/2)
       {
        y = fmod(n,N2);
        filled_S[(M1_to-m+1)*M2_to-n-1] = filler[(x+1)*N2-y-1];
       }
       else
         filled_S[(M1_to-m+1)*M2_to-n-1] = S[(M1_from-m+1)*M2_from-n-1];
      }
     }
    }
    return (filled_S);
}
```

3.2.6 Frequency-fold

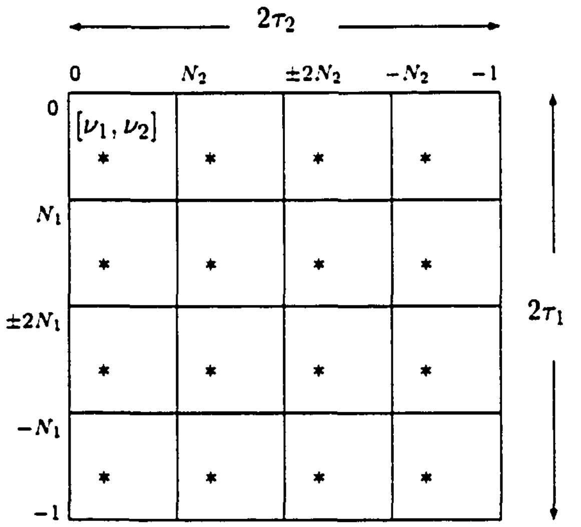


Figure 3.18: Frequency folding due to sampling.

The **frequency-fold** algorithm is used to introduce the aliasing due to sampling into the model and to synthesize the output images in the case that the display size is not $2\tau_1 \times 2\tau_2$. The operation of the **frequency-fold** algorithm is illustrated in Figures 3.18 and 3.19. Figure 3.18 shows a $2\tau_1 \times 2\tau_2 = 4N_1 \times 4N_2$ matrix representation being sampled N_1 times per vertical period and N_2 times per horizontal period. To simulate the effects of sampling, every frequency index pair $[\nu_1, \nu_2]$ in the $N_1 \times N_2$ data structure created by the algorithm is calculated by "folding" (summing) the coefficients at each of the corresponding marked points in the input data structure. Because the possible folding scenarios will not all be as symmetrical as the one shown in Figure 3.18, the algorithm recognizes the packing method used in the transform data structures. For example, synthesis of the spatial domain display might require frequency-folding from $6N_1 \times 6N_2$ to $4N_1 \times 4N_2$. This would be accomplished as shown in Figure 3.19.

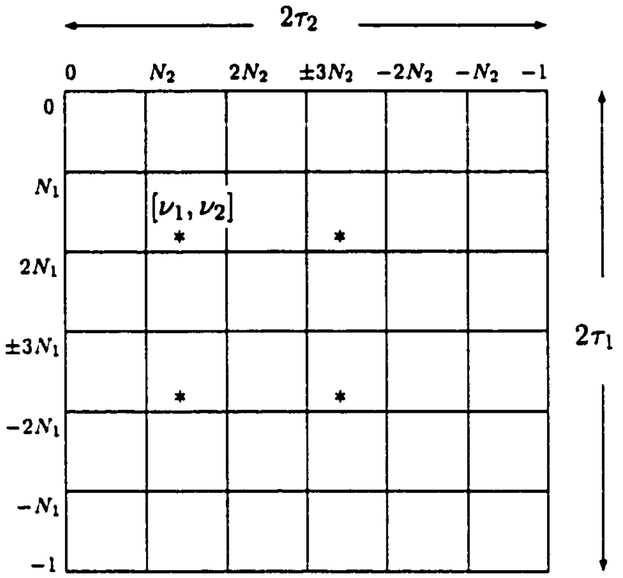


Figure 3.19: Frequency folding due to sampling.

parameters:

- a pointer to an array of type COMPLEX, which contains a full-size matrix representation;
- the dimensions of the input array;
- the dimensions of the output array.

pseudo-code:

```
COMPLEX* freq_fold2_D(F, M1_from, M2_from,M 1_to, M2_to)
{
COMPLEX *f_cap;
int     mu,nu,mu_pr,nu_pr;
  f_cap = allocate(M1_to*M2_to,sizeof(COMPLEX));
  for (mu=0; mu<M1_to; mu++)
     for (nu=0; nu<M2_to; nu++)
     {
      f_cap[mu][nu] = 0;
      for (mu_pr=mu; mu_pr<=M1_from/2; mu_pr+=M1_to)
      {
       for (nu_pr=nu; nu_pr<=M2_from/2; nu_pr+=M2_to)
          f_cap[mu][nu] += F[mu_pr*M2_from + nu_pr];
       for (nu_pr=nu-M2_to; nu_pr>=-M2_from/2; nu_pr-=M2_to)
          f_cap[mu][nu] += F[(mu_pr+1)*M2_from + nu_pr];
      }
      for (mu_pr=mu-M1_to; mu_pr>=-M1_from/2; mu_pr-=M1_to)
      {
       for (nu_pr=nu; nu_pr<=M2_from/2; nu_pr+=M2_to)
          f_cap[mu][nu] += F[(M1_from+mu_pr)*M2_from + nu_pr];
       for (nu_pr=nu-M2_to; nu_pr>=-M2_from/2; nu_pr-=M2_to)
          f_cap[mu][nu] += F[(M1_from+mu_pr+1)*M2_from + nu_pr];
      }
     }
  return f_cap;
}
```

3.2.7 Fast Fourier Transform (FFT)

This FFT algorithm computes the direct or inverse Fourier transform of a 2-D array of COMPLEX coefficients efficiently, in-place. In the environment, however, the original frequency domain matrix representation must be preserved, and so the FFT algorithm creates a new data structure and copies the array to be displayed into it before processing commences.

Any algorithm which computes an M-point Discrete Fourier Transform (DFT) array, and does so with $O(M \log_2(M))$ time complexity, is termed a 1-D *Fast Fourier Transform* [4]. The 2-D inverse (or direct) Fourier transform matrices are computed as a sequence of 1-D inverse (or direct) FFTs, as illustrated in Figure 3.20. *Successive doubling*, an iterative

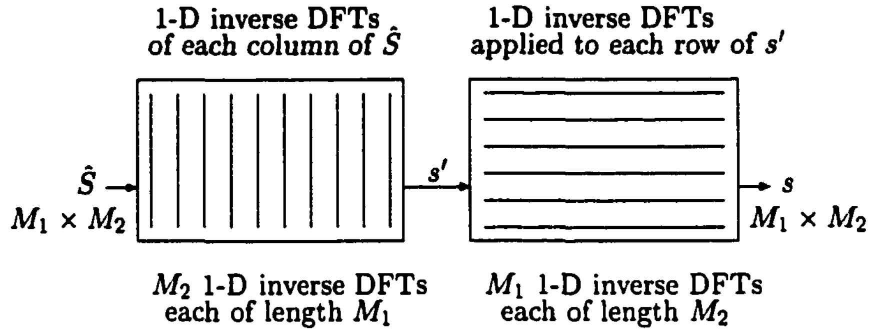

1-D inverse DFTs of each column of $\hat{S}$
1-D inverse DFTs applied to each row of s'
$\hat{S}$
$M_1 \times M_2$
s'
s
$M_1 \times M_2$
M_2 1-D inverse DFTs each of length M_1
M_1 1-D inverse DFTs each of length M_2

Figure 3.20: Computing a 2-D inverse DFT as a sequence of 1-D direct DFTs.

technique, is one of several FFT algorithms. It can be successfully utilized when M is a power of 2 [18] and it is implemented in the environment.

To verify the implementation and accuracy of the Fast Fourier Transform algorithm, (a) the direct FFT of an input scene defined in the spatial domain can be computed using the algorithm, (b) the output from the direct FFT can be fed into the inverse FFT algorithm, and (c) the resulting matrix of coefficients can be compared with that of the original scene. There should be no discernible difference greater than that attributable to floating point error.

parameters:

- a pointer to an array of type COMPLEX which contains a full-size matrix representation;
- a flag indicating direct (or inverse) transform required;

- the dimensions of the matrix representation — vertical, horizontal (<u>must</u> be powers of 2).

<u>pseudo-code:</u>

```
======================================================================
   Computes the Fourier transform of a 2-D set of COMPLEX coefficients.
======================================================================
COMPLEX* FFT2_D(s, direct, M, N)
{
COMPLEX  *s_cap,*wm,*wn;
int      betaM,betaN,m,n;
  s_cap = allocate(M*N,sizeof(COMPLEX));

/* Copy the existing data structure into the new */
  for (m=0; m<M; m++)
    for (n=0; n<N; n++)
       s_cap[m*N+n] = s[m*N+n];

  betaM = exponent(M);             (M = 2**betaM)
  betaN = exponent(N);             (N = 2**betaN)

  wm = rootsOfUnity(M,direct);
  wn = rootsOfUnity(N,direct);

  for (n=0; n<N; n++)
     FFT_col(s,wm,M,N,n,betaM);
  for (m=0; m<M; m++)
     FFT_row(s,wn,N,m,betaN);
  if (!direct)
    for (m=0; m<M; m++)
       for (n=0; n<N; n++)
          s[m,n] = M*N*s[m,n];
  return s;
}

======================================================================
   Computes the roots of unity for use as a look-up table by the 1-D
   FFT procedures.
   parameters:
     the number of roots required;
     a flag indicating direct (or inverse) transform required.
======================================================================
COMPLEX* rootsOfUnity(divisions, direct)
{
COMPLEX *w,*one,*prev;
int j;
  w = allocate(divisions+1,sizeof(COMPLEX));
  w[0].r = 1;
  w[0].i = 0;
  w[1].r = cos(2*PI/divisions);
  w[1].i = -sin(2*PI/divisions);
  for (j=2; j<(divisions/2); j++)
  {
   w[j].r = w[j-1].r * w[1].r - w[j-1].i * w[1].i;
   w[j].i = w[j-1].i * w[1].r + w[j-1].r * w[1].i;
  }
  if (!direct)
    for (j=1; j<(divisions/2); j++)
       w[j].i = - w[j].i;
  return w;
}

========================================================================
   Computes the FFT of one column of the 2-D array.
   parameters:
     a pointer to an array of type COMPLEX which contains a full-size
         matrix representation;
     a pointer to an array of type COMPLEX which contains the roots of
         unity;
     the height of the matrix representation;
     the width of the matrix representation;
     column index;
     power of 2 of M.
========================================================================
void FFT_col(s, w, M, N, col, beta)
{
COMPLEX  temp;
int period,step,n,m,t;
  bitReversalSort_col(s,beta,M,N,col);
  for (period=1, step=M/2; period<M; period*=2, step/=2)
    for (n=0; n<M; n+=period)
      for (m=0; (2*m)<M; n++, m+=step)
      {
       t = n + period;
       temp = s[t*N + col] * w[m];
       s[t*N + col] = s[n*N + col] - temp;
       s[n*N + col] = s[n*N + col] + temp;
      }
  for (t=0; t<M; t++)
     s[t*N + col] = s[t*N + col]/M;
}

========================================================================
   Computes the FFT of one row of the 2-D array.
========================================================================
void FFT_row(s, w, M, N, row, beta)
{  ...
   differs from FFT_col only in indexing and
     parameter limits
   ...
}

========================================================================
   Performs a bit reversal sort on a column of the 2-D array.
   parameters:
     a pointer to an array of type COMPLEX which contains a full-size
         matrix representation;
     power of 2 of M;
     the height of the matrix representation;
     the width of the matrix representation;
     column index.
========================================================================
void bitReversalSort_col(s, beta, M, N, col)
{
COMPLEX temp;
int m,mDash;
  for (m=0; m<M; m++)
  {
   mDash = bitReverse(m,beta);
   if (m<mDash)
   {
    temp = s[m*N + col];
    s[m*N + col] = s[mDash*N + col];
    s[mDash*N + col] = temp;
   }
  }
}

========================================================================
   Performs a bit reversal sort on a row of the 2-D array.
========================================================================
void bitReversalSort_row(s, beta, M, N, row)
```

```
   ...
   differs from bitReversalSort_col only in indexing
     and parameter limits
   ...

========================================================================
   Performs a bit reverse on n, where 0 < n < 2**beta.
   parameters:
     a pointer to an array of type COMPLEX which contains a full-size
         matrix representation;
     power of 2 of M;
     the height of the matrix representation;
     the width of the matrix representation;
     column index.
========================================================================
int bitReverse(n, beta)
{
int nDash,b;
  nDash = 0;
  for (b=0; b<beta; b++)
  {
   nDash = 2*nDash + (fmod(n, 2));
   n = n/2;
  }
  return nDash;
}
```

3.2.8 Noise box

The noise box uses a (pseudo-)random number generator to create an $N_1 \times N_2$ matrix representation. The set of routines utilized is drawn from a custom ANSI C library for random number generation [20]. The use of this library is recommended by the routines' principal author as a replacement for the ANSI C **rand** and **srand** functions, particularly in simulation applications where the statistical "goodness" of the random number generator is important. The generator used in this library is a so-called "Lehmer random number generator" which returns a pseudo-random number uniformly distributed between 0.0 and

1.0. The period is $(MODULUS - 1)$ where $MODULUS = 2,147,483,647$ and the smallest and largest possible values are $(1/MODULUS)$ and $1 - (1/MODULUS)$ respectively [23]. Library routines are used to generate random variables from a Gaussian distribution with the specified input parameters [15].

Generator	Range (x)	Mean	Variance
Gauss(m, s)	all x	m	s^2

parameters:

- standard deviation of input scene.

pseudo-code:

```
COMPLEX *noise_box2_D(s_std_dev_2D)
{
COMPLEX *e2D,*noise;
double  mean,std_dev;
int     n,direct;
  noise = allocate(N[1]*N[0]*sizeof(COMPLEX));

  mean = 0;
  std_dev = s_std_dev_2D/SNR;

  PutSeed();                        /* reseed the random number gen */
  for (n=0; n<N[1]*N[0]; n++)
  {
   noise[n] = Gauss(mean,std_dev);
  }

  direct = 1;
  e2D = FFT2_D(noise,direct,N[1],N[0]);
  return e2D;
}
```

```
========================================================================
   This procedure is used to initialize the random number generator.
   The initial seed is obtained from the system clock.
```

```
========================================================================
static long PutSeed()
{
static long seed;
  seed = time(NULL) % MODULUS;
  return seed;
}

=======================================================================
   This procedure returns a Gaussian distributed real.
   (NOTE: use s > 0)
   parameters:
     the mean of the brightness values used to define the digitized
         image (m);
     the standard deviation of the brightness values used to define
         the digitized image (s).
=======================================================================
double Gauss(m, s)
{
  return(m + s * Normal());
}

=======================================================================
   This procedure returns a standard normal distributed real.
   It uses a very accurate approximation of the Normal idf due
          to Odeh & Evans.
=======================================================================
double Normal()
{
double p0 = 0.322232431088;
double p1 = 1.0;
double p2 = 0.342242088547;
double p3 = 0.204231210245e-1;
double p4 = 0.453642210148e-4;
double q0 = 0.099348462606;
double q1 = 0.588581570495;
double q2 = 0.531103462366;
double q3 = 0.103537752850;
double q4 = 0.385607006340e-2;
double u, t, p, q, Random();
  u = Random();
  if (u < 0.5)
    t = sqrt(-2*log(u));
  else
    t = sqrt(-2*log(1.0 - u));
  p = p0 + t*(p1 + t*(p2 + t*(p3 + t*p4)));
  q = q0 + t*(q1 + t*(q2 + t*(q3 + t*q4)));
  if (u < 0.5)
    return(p/q - t);
  else
    return(t - p/q);
}

=========================================================================
   Random is a Lehmer generator that returns a pseudo-random real
     number uniformly distributed between 0.0 and 1.0.
   The period is (MODULUS - 1) where MODULUS = 2,147,483,647 and the
     smallest and largest possible values are (1 / MODULUS) and
     1 - (1 / MODULUS) respectively.
   parameters:
     initial seed.
=========================================================================
double Random(seed)
{
const long Q = MODULUS / MULTIPLIER;
const long R = MODULUS % MULTIPLIER;
long t;
  t = MULTIPLIER * (seed % Q) - R*(seed/Q);
  if (t > 0)
    seed = t;
  else
    seed = t + MODULUS;
  return (seed/MODULUS);
}
```

3.2.9 Origin-center

The objective of the **origin-center** algorithm is to reorient a $kN_1 \times kN_2$ transform data structure for frequency domain display purposes. The data structure is divided into four equal quadrants, each of dimension $(k/2)N_1 \times (k/2)N_2$, as shown in Figure 3.21 with $k = 4$. Elements of each pair of diagonally opposite quadrants are exchanged, in-place. This requires a conceptual shift from the general philosophy used throughout the implementation. Other algorithms consider the first "quadrant" to include the row with index equal to half

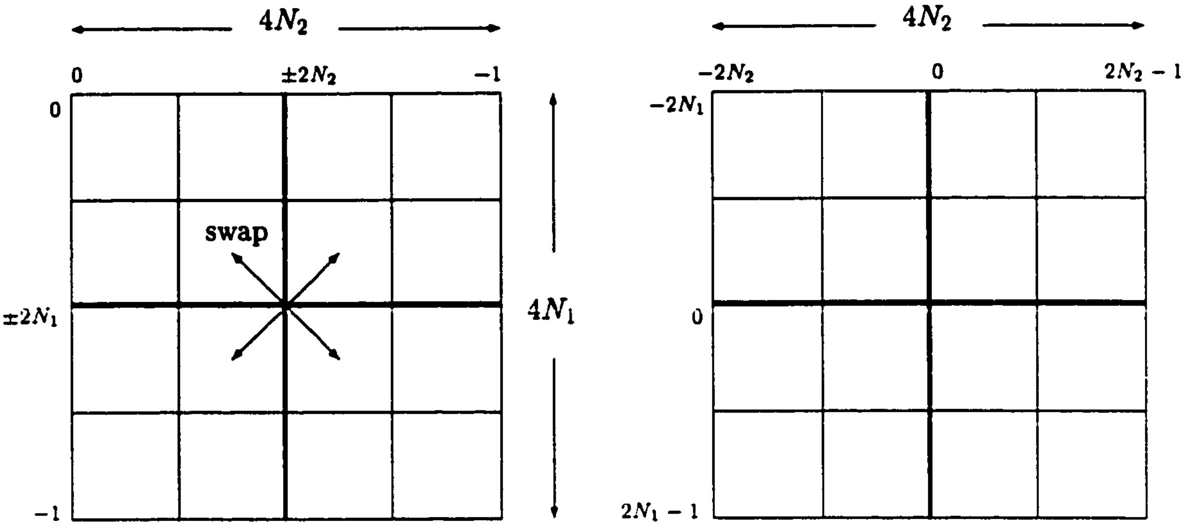


Figure 3.21: Origin-centering.

the height of the array and the column with index equal to half the width of the array. The **origin-center** algorithm excludes this row and column from the first quadrant. It can readily be seen, however, that the integrity of the data structure is maintained. The row and column in question are the ones that contain the sum of the coefficients at ± the lowest stored frequencies and can, therefore, be just as correctly placed in the other quadrants. For precision, the coefficients in the first row and column of the origin-centered array should be divided by 2 (the $[-2N_1, -2N_2]$ element by 4). It was determined by experimentation, however, that for display purposes there is no visible difference.

parameters:

- a pointer to an array of type double, which contains the magnitude values for display;
- the dimensions of the array.

pseudo-code:

```
void origin_center(disp, M, N)
{
double   temp;
int      m,n;
  for (m=0; m<(M/2); m++)
    for (n=0; n<(N/2); n++)
    {
     temp = disp[m*N + n];
     disp[m*N + n] = disp[(m+M/2)*N + (n+N/2)];
     disp[(m+M/2)*N + (n+N/2)] = temp;
     temp = disp[(m+M/2)*N + n];
     disp[(m+M/2)*N + n] = disp[m*N + (n+N/2)];
     disp[m*N + (n+N/2)] = temp;
    }
}
```

3.2.10 Replicate

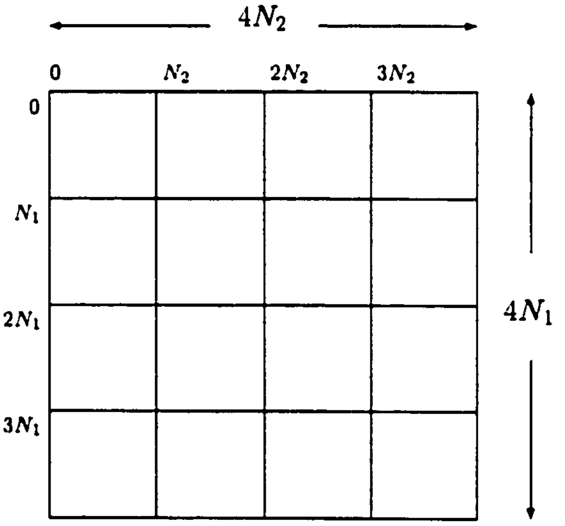


Figure 3.22: The replicated data structure for frequency domain display.

The **replicate** algorithm creates a $kN_1 \times kN_2$ data structure from an $N_1 \times N_2$ array of transform coefficients by periodically replicating the contents of the array in both dimensions, as shown in Figure 3.22 with $k = 4$. If $k < 1$, the $N_1 \times N_2$ data structure is clipped to $kN_1 \times kN_2$.

parameters:

- a pointer to an array of type COMPLEX, which contains a matrix representation;
- the dimensions of the input array;
- the dimensions of the output array.

pseudo-code:

```
COMPLEX *replicate(F, M1_from, M2_from, M1_to, M2_to)
{
COMPLEX *F2D;
int     n1,n2;
  if ((M1_from>M1_to) || (M2_from>M2_to))
    {
     F = clip(F,M1_from,M2_from,min(M1_from,M1_to),min(M2_from,M2_to));
     M1_from = min(M1_from,M1_to);
     M2_from = min(M2_from,M2_to);
    }
  for (n1=0; n1<M1_from; n1++)
     for (n2=0; n2<M2_to; n2++)
        F2D[n1*M2_to + n2] = F[n1*M2_from + fmod(n2,M2_from)];
  for (n1=M1_from; n1<M1_to; n1++)
     for (n2=0; n2<M2_to; n2++)
        F2D[n1*M2_to + n2] = F2D[(n1-M1_from)*M2_to + n2];
  return F2D;
}
```

3.2.11 Vector multiplication

The **vector multiplication** algorithm creates a 2-D filter transform data structure and stores in it the matrix representation of a 2-D filter computed as the cross-product of the

1-D arrays of separable vertical and horizontal components.

parameters:

- a pointer to an array of type COMPLEX, which contains the transform coefficients of the vertical filter component;
- a pointer to an array of type COMPLEX, which contains the transform coefficients of the horizontal filter component;
- the length of the vertical filter component array;
- the length of the horizontal filter component array.

pseudo-code:

```
COMPLEX *vector_mult(matrxA, matrxB, A, B)
{
COMPLEX *matrxC;                  /* matrxC is A x B  */
COMPLEX temp;
int a,b;
  matrxC = allocate(A*B*sizeof(COMPLEX));
  for (a=0; a<A; a++)
     for (b=0; b<B; b++)
     {
      matrxC[a*A+b].r = matrxA[a].r*matrxB[b].r -
                                    matrxA[a].i*matrxB[b].i);
      matrxC[a*A+b].i = matrxA[a].i*matrxB[b].r -
                                    matrxA[a].r*matrxB[b].i);
     }
  return matrxC;
}
```

Chapter 4

Time/Memory Considerations

4.1 Response Time

The facility to visualize the matrix representation of the input scene at various processing stages within the c/d/c system model provides insight into the extent to which each system component contributes to effects and artifacts observed in the output image. However, if the matrix representation could not be displayed interactively within a reasonable period of time, the environment would not be very useful. One objective of this research, therefore, was to provide an environment with acceptable response times which would facilitate modification of the c/d/c system model parameters and allow convenient display of the matrix representation of the input scene at the input and output of each component of the c/d/c system model.

4.1.1 Spatial domain convolution

One of the computational techniques that is naturally part of the c/d/c system model (depending on implementation details) is 2-D spatial domain convolution, which is CPU-time intensive. Recall that the *convolution* $g = h \circledast s$, which results from passing the periodic function s through the 2-D filter h to obtain the periodic function g, is defined in the spatial domain as

$$g(x_1, x_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x'_1 - x_1, x'_2 - x_2)s(x'_1, x'_2)dx'_1 dx'_2$$

for $(x_1, x_2) \in \Re \times \Re$. In this equation, s is a function defined for all $(x_1, x_2) \in \Re \times \Re$. If. instead, s were a digitized input scene defined by brightness values specified on a grid of points within a 2-D spatial coordinate system, then the analogous definition would be

$$g[m_1, m_2] = \sum_{m'_1=-\infty}^{\infty} \sum_{m'_2=-\infty}^{\infty} h[m_1 - m'_1, m_2 - m'_2]s[m'_1, m'_2]. \tag{4.1}$$

The array h is the *point spread function* of the convolution operation which weights the contribution of any $s[m'_1, m'_2]$ to a given $g[m_1, m_2]$ as a function of the distance between the points $[m_1, m_2]$ and $[m'_1, m'_2]$. The calculation of each value $g[m_1, m_2]$, therefore, could potentially necessitate the summation of every $s[m'_1, m'_2]$ weighted according to the point spread function.

Equivalently, Equation 4.1 can be rewritten as

$$g[m_1, m_2] = \sum_{m'_1=-\infty}^{\infty} \sum_{m'_2=-\infty}^{\infty} h[m'_1, m'_2]s[m_1 - m'_1, m_2 - m'_2].$$

Although the summation in the discrete convolution equation is theoretically infinite, in most applications all the non-zero values of h are contained within a relatively small region centered on [0,0]. That is, there exist positive integer constants M_1', M_2' and the associated set of array indices

$$K_h = \{[m_1', m_2'] : m_1' = 0, \pm 1, \dots, \pm M_1', m_2' = 0, \pm 1, \dots, \pm M_2'\}$$

such that $h[m_1', m_2'] = 0$ for all $[m_1', m_2'] \notin K_h$. In this case, the summation in the discrete convolution equation may be restated as

$$g[m_1, m_2] = \sum\sum_{[m_1', m_2'] \in K_h} h[m_1', m_2'] s[m_1 - m_1', m_2 - m_2'].$$

Note that, by convention, the cardinality of K_h is $|K_h| = (2M_1' + 1)(2M_2' + 1)$. That is, the size of the smallest rectangular region that includes all the non-zero values of h is the filter kernel size, $|K_h|$. It is reasonable to consider that $|K_h|$ is no larger than the dimensions of the digitized input scene, because each point in g could then be affected in spatial domain (circular) convolution by every point in the digitized input scene because of the periodicity of s. Realistically, $|K_h|$ would not be that large for most applications.

Similarly, the discrete convolution equation corresponding to reconstruction is

$$r[m_1, m_2] = \sum\sum_{[m_1', m_2'] \in K_d} d[m_1', m_2'] q[m_1 - m_1', m_2 - m_2'].$$

The array d is the point spread function of the convolution operation $r = d \circledast q$. The size of the smallest rectangular region that includes all the non-zero values of d is the filter kernel size, $|K_d|$, and it is reasonable to consider that $|K_d|$ is no larger than the dimensions of the digitized reconstructed scene.

The discrete convolution equation corresponding to digital filtering is

$$q[m_1, m_2] = \sum_{[m_1', m_2'] \in K_f} \sum f[m_1', m_2'] p[m_1 - m_1', m_2 - m_2'].$$

The array f is the point spread function of the convolution operation $q = f \circledast p$. The size of the smallest rectangular region that includes all the non-zero values of f is the filter kernel size, $|K_f|$, and it is reasonable to consider that $|K_f| = N_1 N_2$ because p and q are periodic with period $N_1 \times N_2$.

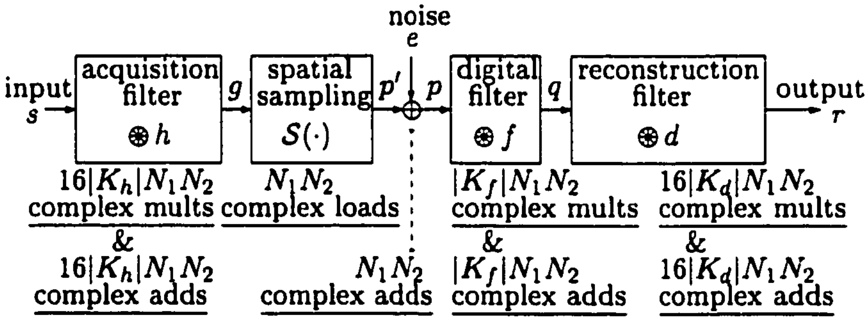

noise
e
input
s
acquisition filter
⊛h
g
spatial sampling
S(·)
p'
p
digital filter
⊛f
q
reconstruction filter
⊛d
output
r
16|K_h|N_1N_2 complex mults & 16|K_h|N_1N_2 complex adds
N_1N_2 complex loads
N_1N_2 complex adds
|K_f|N_1N_2 complex mults & |K_f|N_1N_2 complex adds
16|K_d|N_1N_2 complex mults & 16|K_d|N_1N_2 complex adds

Figure 4.1: Spatial domain processing complexity.

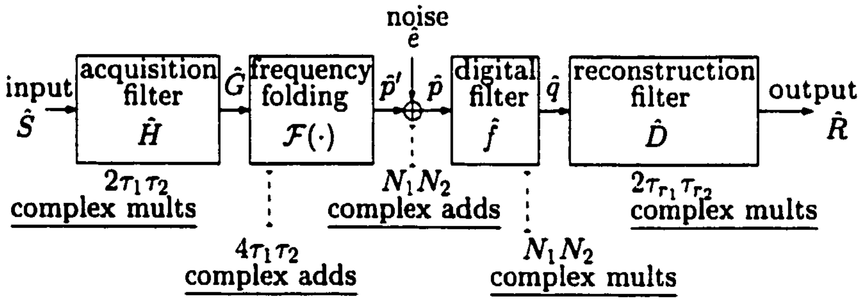


Figure 4.2: Frequency domain processing complexity.

4.1.2 Parameters affecting response time

Figures 4.1 and 4.2 compare the number and type of operations required to process a digitized input scene through the c/d/c system model in the spatial domain and frequency domain respectively. This example assumes

- sampling grid size of $N_1 \times N_2$,
- digitized input scene size of $4N_1 \times 4N_2$,
- digitized reconstructed scene (display) size of $4N_1 \times 4N_2$,
- representation passband $\tau_1 \times \tau_2 = 2N_1 \times 2N_2$,
- reconstruction passband $\tau_{r_1} \times \tau_{r_2} = 2N_1 \times 2N_2$,
- filter kernel sizes $|K_h|, |K_f|$ and $|K_d|$,
- processing of composite image.

Figure 4.1 shows the operations that would be required if processing were performed in the spatial domain, and Figure 4.2 shows the operations required if processing is performed in the frequency domain. In addition to the time complexity of Figure 4.1 or 4.2, the time complexity required to display any image in the domain not used for processing is $O(N_d \log_2(N_d))$, where N_d is the 2-D size of the display.

The parameters which affect response time are, therefore:

- sampling grid, $N_1 \times N_2$;
- representation passband parameters, τ_1, τ_2 (related to N_1, N_2, so not independent parameters);
- reconstruction passband parameters, τ_{r_1}, τ_{r_2} (related to N_1, N_2, so not independent parameters);

- display size, N_d (related to $N_1 \times N_2$, so not an independent parameter);
- filter kernel size, $|K_h|$ (limited to $4N_1 \times 4N_2$ by digitized input scene size);
- filter kernel size, $|K_f|$ (limited to $N_1 \times N_2$ by the periodicity of p and q);
- filter kernel size, $|K_d|$ (limited to $4N_1 \times 4N_2$ by display size).

4.1.3 Algorithms affecting response time

Because the size of the filter kernels, $|K_h|$, $|K_d|$ and $|K_f|$, in the example are such that $|K_h|, |K_d| \leq 16N_1N_2$ and $|K_f| \leq N_1N_2$, Figure 4.1 shows that the time complexity of processing in the spatial domain is $O(N_1^2N_2^2)$. This time complexity is attributable to spatial domain convolution. From Figure 4.2, the time complexity of processing in the frequency domain is shown to be $O(N_1N_2)$, because typically $\tau_1 = 2N_1$, $\tau_2 = 2N_2$.

In addition to the time complexities shown in Figures 4.1 and 4.2, application of a 2-D FFT is required each time it is necessary to convert a matrix representation from one domain to the other. This time penalty is incurred whether processing occurs in the spatial domain or the frequency domain, as shown in Table 4.1.

domain	process / display	time complexity
spatial	process	$O(N_1^2N_2^2)$
frequency	display	$O(N_1N_2\log_2(N_1N_2))$
frequency	process	$O(\tau_1\tau_2) \simeq O(N_1N_2)$
spatial	display	$O(N_1N_2\log_2(N_1N_2))$

Table 4.1: Comparison of the time complexity of processing in the spatial domain versus processing in the frequency domain.

The implementation of spatial domain convolution and the calculation of direct and inverse Fourier transforms was carefully considered. To minimize the response time experienced by the user, processing occurs exclusively in the frequency domain so that spatial domain convolution can be achieved by multiplication in the frequency domain on a frequency-by-frequency basis. In addition, *pre-processing* of input scene data is performed to ensure that Fourier matrix representations are the standard format of the data supplied to the model.

4.1.4 Implementation conditions affecting response time

As discussed in Chapter 1, the environment provides the user with the capability to design a c/d/c system model by modifying any or all of the system and/or component parameters. In the interests of reducing response time to a minimum when desired, the environment operates in two distinct modes. In 2-D mode, the user has the capability to redesign a c/d/c system model by changing the system and/or component parameters and, subsequent to every change, the input scene is processed through the system to the currently selected c/d/c model stage. In 1-D mode, the user retains the capability to redesign the digital imaging system, but changes to system or component parameters do not affect the displayed 2-D image representation while the environment remains in 1-D mode. Instead, a representative 1-D scene, selected from the rows or columns of the 2-D input scene is displayed and interactively updated as changes are made to the horizontal or vertical components of the system parameters. The 1-D mode of operation allows the effects of parameter changes to be viewed instantly without the delay imposed by a 2-D FFT. When 1-D mode is exited, the environment reverts to 2-D mode and the effects of the altered system parameters become visible when the 2-D scene is processed through the end-to-end system.

4.2 Memory Requirements

The amount of memory required by the environment was also carefully considered. For optimum performance, as an input scene is processed through the c/d/c system model, it would be propitious to perform all intermediate calculations once and store the results. Subsequently, any matrix representation in either domain would be immediately available for display. However, this would be feasible in terms of memory only if the sampling grid were small, in which case processing time would not be a problem.

4.2.1 Systematic study of passband limits

A study was conducted with $\tau_1 \times \tau_2 = N_1 \times N_2, 2N_1 \times 2N_2, 3N_1 \times 3N_2$, and $4N_1 \times 4N_2$ using the various input scenes shown in Figure 4.3. The object of the study was to determine the representation passband parameter beyond which any aliasing contribution from frequencies beyond the representation passband is invariably negligible. The system model component parameters were:

- Dirichlet scene filter with

$$\hat{T} = \begin{cases} 1 & \omega = 0, \pm 1, \ldots, \pm\tau \\ 0 & \text{otherwise} \end{cases}$$

  i.e., input scene band-limited by τ, but no ringing suppression on input;

- Gaussian acquisition filter (see Figure 4.7 with the parameter values indicated in Table 4.2);

- CLS restoration filter with the parameter values indicated in Table 4.3;

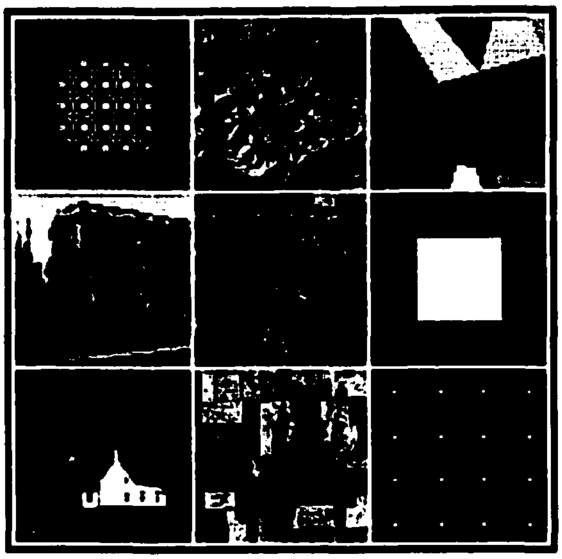


Figure 4.3: $N_1 \times N_2$ spatial domain display of input scenes "zone," "Yukon," "polygons," "Oxford," "MontrealNE," "square," "Oban," "Alberta," and "multiple." ($\tau_1 \times \tau_2 = 2N_1 \times 2N_2$.)

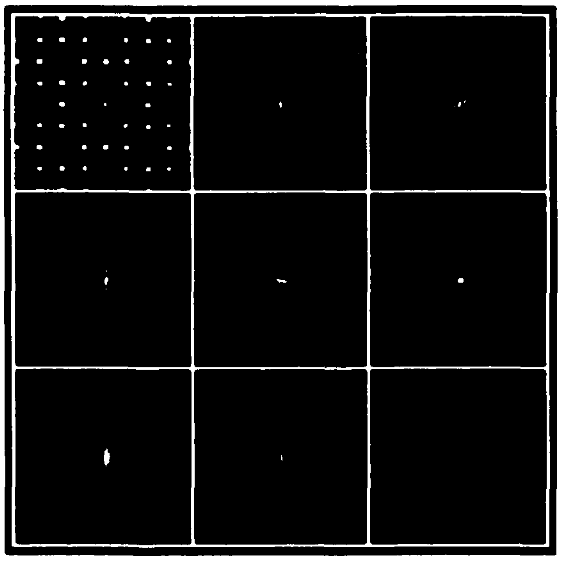


Figure 4.4: $N_1 \times N_2$ frequency domain display of input scenes "zone," "Yukon," "polygons," "Oxford," "MontrealNE," "square," "Oban," "Alberta," and "multiple." ($\tau_1 \times \tau_2 = 2N_1 \times 2N_2$.)

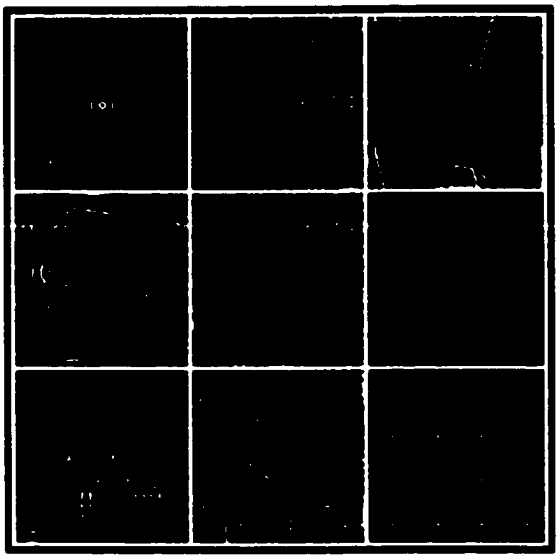


Figure 4.5: $N_1 \times N_2$ spatial domain display of aliased component of reconstructed scenes "zone," "Yukon," "polygons," "Oxford," "MontrealNE," "square," "Oban," "Alberta," and "multiple." ($\tau_{r_1} \times \tau_{r_2} = 4N_1 \times 4N_2$.)

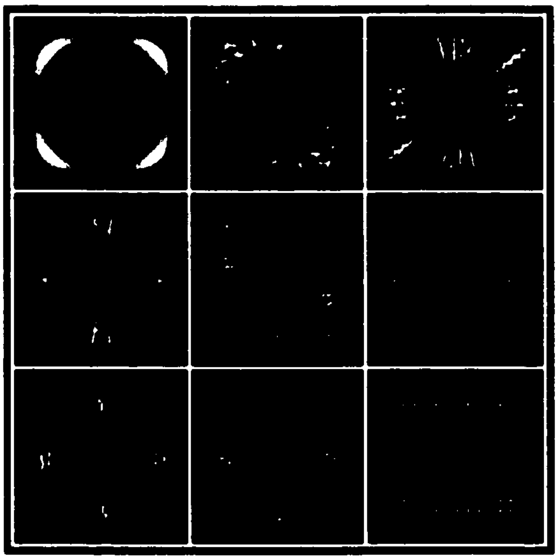


Figure 4.6: $N_1 \times N_2$ frequency domain display of aliased component of reconstructed scenes "zone," "Yukon," "polygons," "Oxford," "MontrealNE," "square," "Oban," "Alberta," and "multiple." ($\tau_{r_1} \times \tau_{r_2} = 4N_1 \times 4N_2$.)

$\hat{H}(1/2\xi)$	β
0.01	2.5
0.08	1.8
0.17	1.5
0.27	1.3
0.32	1.2
0.39	1.1
0.46	1.0
1.0	≈ 0

Table 4.2: $\hat{H}(1/2\xi)$ values corresponding to Gaussian filter parameter values

λ	β
0.0004	2.5
0.003	1.8
0.007	1.5
0.009	1.3
0.004	1.2
0.005	1.1
0.0001	1.0
0.001	≈ 0

Table 4.3: CLS filter parameter values corresponding to Gaussian filter parameter values

- PCC reconstruction filter ($\alpha = -0.5$).

Eight different Gaussian filter parameter values were used in the study to ensure that artifacts due to aliasing were not invariably masked by blurring. In Figure 4.7, the segment of each curve beyond the Nyquist frequency controls the contribution to aliasing effects in the reconstructed image. It will be noted that the case $\beta = 2.5$ appears as though it would all but obliterate any frequencies beyond the sampling passband. The case $\beta \approx 0$ (an all-pass acquisition filter) was included as a control case, although only the cases $1.0 \leq \beta \leq 2.5$ are realistic. The eight different CLS filter parameter values were empirically

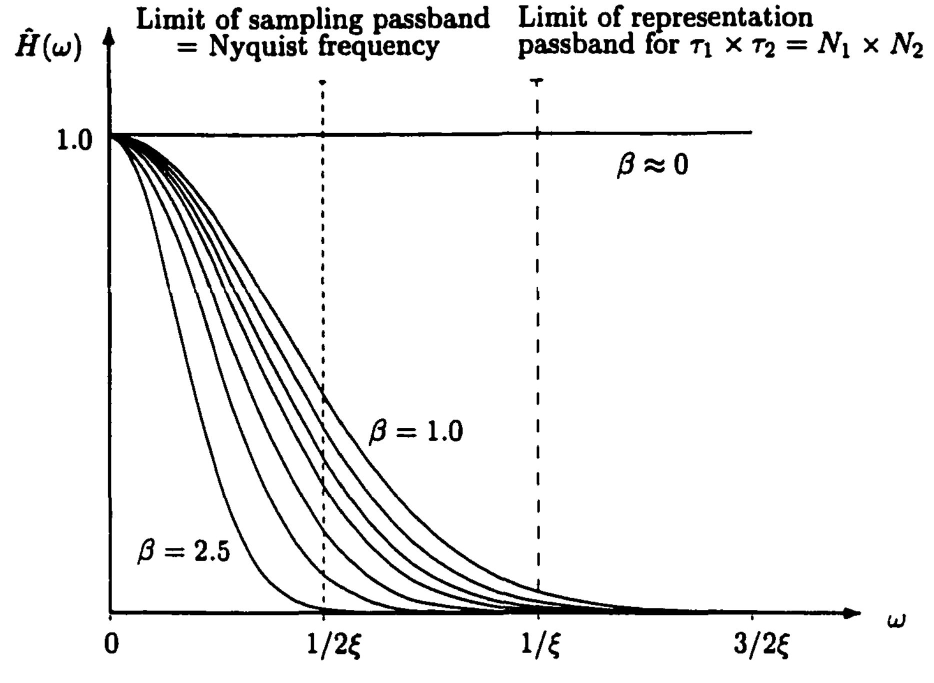


Figure 4.7: Varying β in filter definition $\hat{H}$

chosen by inspection of the 1-D CLS filter responses to produce best-case restoration for the corresponding Gaussian filter parameter values. The study was conducted using digital input scenes with $M_1 \times M_2 \geq 512 \times 512$ and a sampling grid size of $N_1 \times N_2 = 64 \times 64$. These choices ensured that any increase in $\tau_1 \times \tau_2$ increased the number of coefficients with the potential to contribute to aliasing from frequencies beyond the sampling passband.

In selecting the reconstruction passband parameters for the study, it was important to consider the formation of the aliased component of the reconstructed output in the frequency domain. Figure 4.8 demonstrates the relationships among the sampling, representation, and reconstruction passbands in each of the four cases included in the study. Each diagram depicts the matrix representation $\hat{R}_a$. Each blank grid square is an $N_1 \times N_2$ matrix representation formed by frequency folding from the cross-hatched area bounded by

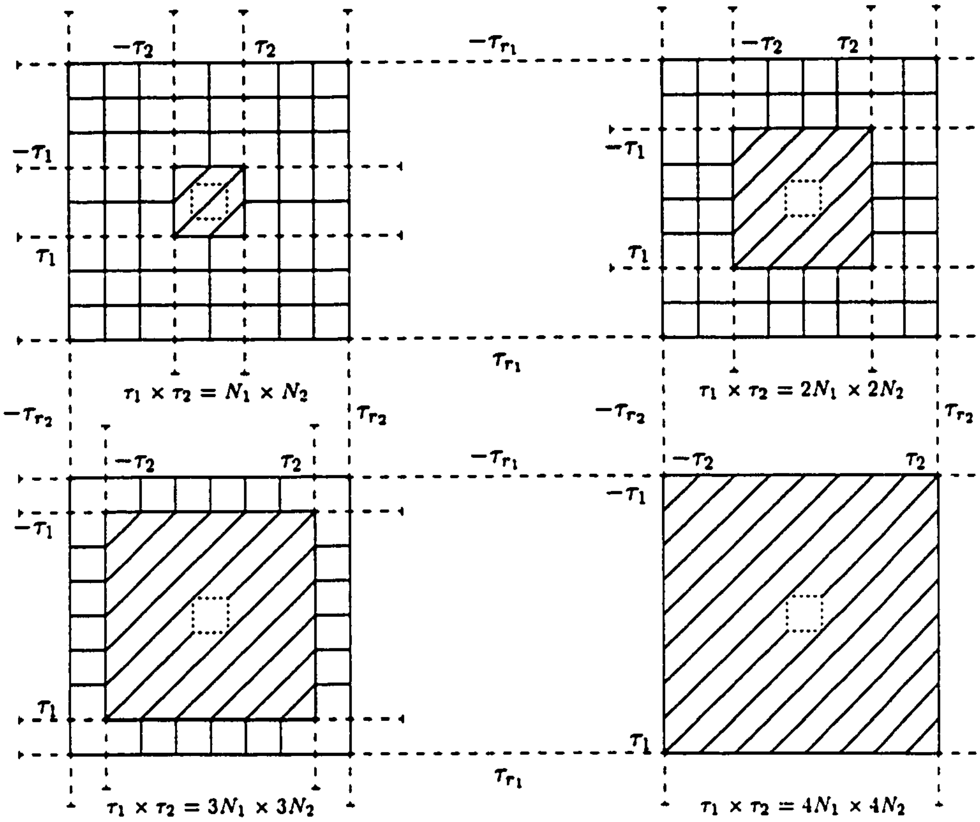

$-\tau_2$
τ_2
$-\tau_1$
τ_1
$-\tau_{r_1}$
τ_{r_1}
$-\tau_{r_2}$
τ_{r_2}
$\tau_1 \times \tau_2 = N_1 \times N_2$
$\tau_1 \times \tau_2 = 2N_1 \times 2N_2$
$\tau_1 \times \tau_2 = 3N_1 \times 3N_2$
$\tau_1 \times \tau_2 = 4N_1 \times 4N_2$

Figure 4.8: Relationship between sampling, representation & reconstruction passbands in the formation of $\hat{R}_a$ at each value of $\tau_1 \times \tau_2$.

the representation passband parameters and periodically replicated to fill the area bounded by the reconstruction passband. The grid squares beneath the cross-hatched area have the potential to have been affected by the cascaded component in each case. In order to evaluate the fidelity metric $\|r_a\|$, where

$$\|r_a\| = \sqrt{\sum_{|\nu_1| \leq \tau_{r_1}} \sum_{|\nu_2| \leq \tau_{r_2}} |\hat{R}_a[\nu_1, \nu_2]|^2}, \tag{4.2}$$

with the objective of establishing a representation passband parameter beyond which changes in aliasing effects are negligible, it is essential that the reconstruction passband parameter limits be fixed for all four cases, i.e., that $\tau_{r_1} \times \tau_{r_2} \neq \tau_1 \times \tau_2$, which could produce misleading results due to the increase in the number of coefficients being summed as $\tau_1 \times \tau_2$ increases. Furthermore, the reconstruction passband must be as large as is necessary not to mask, by clipping, any change in aliasing effects which might otherwise have been discernible. To fulfill these requirements for the study, the reconstruction passband parameters were stipulated to be $\tau_{r_1} \times \tau_{r_2} = 4N_1 \times 4N_2$.

Figures 4.3 to 4.6 display the images with $\tau_1 \times \tau_2 = 2N_1 \times 2N_2$ and $\beta = 2.5$. The images were resampled from $4N_1 \times 4N_2$ to $N_1 \times N_2$ matrix representations by folding in the frequency domain. Resampling was necessary to accommodate display of the scenes within the limited space available on a page; unfortunately, resampling resulted in the production of more aliasing artifacts, most-readily apparent in the images of input scene "zone." Figure 4.5 is a qualitative display of the aliased component of the reconstructed images from Figure 4.3. Note that contrast stretching was used in all the figures. The fidelity metric $\|r_a\|$ recorded in Tables 4.4 and 4.5 provides a quantitative evaluation of aliasing as it is affected by blurring and input scene passband parameters.

4.2.1.1 results

Tables 4.4 and 4.5 record the results obtained with the various images shown in Figure 4.3. In every instance, differences (Δ values) refer to the change in $\|r_a\|$ from $\tau_1 \times \tau_2 = N_1 \times N_2$ to $\tau_1 \times \tau_2 = 2N_1 \times 2N_2$. The results suggest that the aliasing contribution from frequencies beyond $\tau_1 \times \tau_2 = N_1 \times N_2$ are neither highly significant nor completely negligible, but that the contribution from frequencies beyond $\tau_1 \times \tau_2 = 2N_1 \times 2N_2$ are invariably negligible. That is, the value of $\|r_a\|$ remained constant for all images and all realistic Gaussian filter parameters as $\tau_1 \times \tau_2$ increased from $2N_1 \times 2N_2$ to $3N_1 \times 3N_2$ and $4N_1 \times 4N_2$. (The single

$\|r_a\|$	$\hat{H}(1/2\xi)$	$\tau = N$	$\tau = 2N$	$\tau = 3N$	$\tau = 4N$	$\Delta\|r_a\|$
zone	.01	0.723526	0.723526	0.723526	0.723526	$< 1 \times 10^{-6}$
Yukon	.01	2.049196	2.049196	2.049196	2.049196	$< 1 \times 10^{-6}$
polygons	.01	0.47781	0.47781	0.47781	0.47781	$< 1 \times 10^{-6}$
Oxford	.01	2.367778	2.367778	2.367778	2.367778	$< 1 \times 10^{-6}$
MontrealNE	.01	1.61808	1.61808	1.61808	1.61808	$< 1 \times 10^{-6}$
square	.0	0.005722	0.005722	0.005722	0.005722	$< 1 \times 10^{-6}$
Oban	.01	1.23698	1.23698	1.23698	1.23698	$< 1 \times 10^{-6}$
Alberta	.01	1.434278	1.434278	1.434278	1.434278	$< 1 \times 10^{-6}$
multiple	.01	0.006275	0.006275	0.006275	0.006275	$< 1 \times 10^{-6}$
zone	.08	1.07014	1.070141	1.070141	1.070141	$+1 \times 10^{-6}$
Yukon	.08	2.511628	2.511628	2.511628	2.511628	$< 1 \times 10^{-6}$
polygons	.08	0.571819	0.571819	0.571819	0.571819	$< 1 \times 10^{-6}$
Oxford	.08	2.898134	2.898134	2.898134	2.898134	$< 1 \times 10^{-6}$
MontrealNE	.08	2.00945	2.00945	2.00945	2.00945	$< 1 \times 10^{-6}$
square	.08	0.006962	0.006962	0.006962	0.006962	$< 1 \times 10^{-6}$
Oban	.08	1.422269	1.422269	1.422269	1.422269	$< 1 \times 10^{-6}$
Alberta	.08	1.743171	1.743171	1.743171	1.743171	$< 1 \times 10^{-6}$
multiple	.08	0.00672	0.00672	0.00672	0.00672	$< 1 \times 10^{-6}$
zone	.17	1.323107	1.323159	1.323159	1.323159	$+5 \times 10^{-5}$
Yukon	.17	2.784922	2.784923	2.784923	2.784923	$+1 \times 10^{-6}$
polygons	.17	0.623858	0.623855	0.623855	0.623855	-3×10^{-6}
Oxford	.17	3.205666	3.205659	3.205659	3.205659	-7×10^{-6}
MontrealNE	.17	2.238565	2.238562	2.238562	2.238562	-3×10^{-6}
square	.17	0.007727	0.007727	0.007727	0.007727	$< 1 \times 10^{-6}$
Oban	.17	1.532338	1.532338	1.532338	1.532338	$< 1 \times 10^{-6}$
Alberta	.17	1.962509	1.962507	1.962507	1.962507	-2×10^{-6}
multiple	.17	0.00683	0.00683	0.00683	0.00683	$< 1 \times 10^{-6}$
zone	.27	1.816512	1.816988	1.816988	1.816988	$+5 \times 10^{-4}$
Yukon	.27	3.43091	3.430916	3.430916	3.430916	$+6 \times 10^{-6}$
polygons	.27	0.748745	0.748712	0.748712	0.748712	-3×10^{-5}
Oxford	.27	3.892756	3.89268	3.89268	3.89268	-8×10^{-5}
MontrealNE	.27	2.769258	2.769276	2.769276	2.769276	$+2 \times 10^{-5}$
square	.27	0.009409	0.009408	0.009408	0.009408	-1×10^{-6}
Oban	.27	1.826583	1.826597	1.826597	1.826597	$+1 \times 10^{-5}$
Alberta	.27	2.446286	2.446278	2.446278	2.446278	-8×10^{-6}
multiple	.27	0.007637	0.007637	0.007637	0.007637	$< 1 \times 10^{-6}$

Table 4.4: Aliasing metric in a systematic study of passband limits ($N_1 \times N_2 = 64 \times 64$).

$\|r_a\|$	$\breve{H}(1/2\xi)$	$\tau = N$	$\tau = 2N$	$\tau = 3N$	$\tau = 4N$	$\Delta\|r_a\|$
zone	.32	2.930809	2.932119	2.932119	2.932119	$+1 \times 10^{-3}$
Yukon	.32	5.152396	5.152419	5.152419	5.152419	$+2 \times 10^{-5}$
polygons	.32	1.077683	1.077583	1.077583	1.077583	-1×10^{-4}
Oxford	.32	5.637332	5.63714	5.63714	5.63714	-2×10^{-4}
MontrealNE	.32	4.165566	4.165647	4.165647	4.165647	$+8 \times 10^{-5}$
square	.32	0.013526	0.013522	0.013522	0.013522	-4×10^{-6}
Oban	.32	2.624406	2.624416	2.624416	2.624416	$+1 \times 10^{-5}$
Alberta	.32	3.620907	3.620948	3.620948	3.620948	$+4 \times 10^{-5}$
multiple	.32	0.00971	0.00971	0.00971	0.00971	$< 1 \times 10^{-6}$
zone	.39	3.25179	3.255657	3.255657	3.255657	$+4 \times 10^{-3}$
Yukon	.39	5.480417	5.480565	5.480565	5.480565	$+1 \times 10^{-4}$
polygons	.39	1.137731	1.137418	1.137418	1.137418	-3×10^{-4}
Oxford	.39	5.969209	5.968836	5.968836	5.968836	-4×10^{-4}
MontrealNE	.39	4.438917	4.439439	4.439439	4.439439	$+5 \times 10^{-4}$
square	.39	0.014365	0.014349	0.014349	0.014349	-2×10^{-5}
Oban	.39	2.780056	2.780297	2.780297	2.780297	$+2 \times 10^{-4}$
Alberta	.39	3.88481	3.88524	3.88524	3.88524	$+4 \times 10^{-4}$
multiple	.39	0.010428	0.010429	0.010429	0.010429	$+1 \times 10^{-6}$
zone	.46	4.755487	4.765907	4.765907	4.765908	$+1 \times 10^{-2}$
Yukon	.46	7.678156	7.678869	7.678869	7.678869	$+7 \times 10^{-4}$
polygons	.46	1.537412	1.536653	1.536653	1.536653	-8×10^{-4}
Oxford	.46	8.060959	8.060604	8.060604	8.060604	-4×10^{-4}
MontrealNE	.46	6.21492	6.216867	6.216867	6.216867	$+2 \times 10^{-3}$
square	.46	0.019222	0.019176	0.019176	0.019176	-5×10^{-5}
Oban	.46	3.788168	3.788828	3.788828	3.788828	$+7 \times 10^{-4}$
Alberta	.46	5.336709	5.338817	5.338817	5.338817	$+2 \times 10^{-3}$
multiple	.46	0.013125	0.01313	0.01313	0.01313	$+5 \times 10^{-6}$
zone	1.0	16.171013	36.32796	56.043322	73.878323	$+2 \times 10^{1}$
Yukon	1.0	15.882069	20.055142	21.518896	22.997176	$+4 \times 10^{0}$
polygons	1.0	3.015319	3.903643	4.965883	6.035234	$+8 \times 10^{-1}$
Oxford	1.0	16.16214	21.829248	23.423581	24.473141	$+6 \times 10^{0}$
MontrealNE	1.0	13.78602	18.05924	19.578237	19.377056	$+4 \times 10^{0}$
square	1.0	0.0385	0.043623	0.059359	0.078138	$+5 \times 10^{-3}$
Oban	1.0	7.571706	9.08459	9.349442	9.398874	$+2 \times 10^{0}$
Alberta	1.0	12.129478	18.272926	19.866667	20.046139	$+6 \times 10^{0}$
multiple	1.0	0.033473	0.04104	0.041878	0.056072	$+1 \times 10^{-2}$

Table 4.5: Aliasing metric in a systematic study of passband limits [cont.] ($N_1 \times N_2 = 64 \times 64$).

exception was an increase of 1×10^{-6} in the fidelity metric for the image "zone," exhibited when the Gaussian parameter was $\beta = 1.0$ and $\tau_1 \times \tau_2 = 4N_1 \times 4N_2$.) The magnitude[1] of the change in $\|r_a\|$, as $\tau_1 \times \tau_2$ increased from $N_1 \times N_2$ to $2N_1 \times 2N_2$, consistently increased for each image as blurring decreased but, even with blurring at a practical minimum ($\beta = 1.0$), the change in $\|r_a\|$ did not exceed 0.25% for any image and for most was less then 0.05%. In Figure 4.7, the segment of each curve beyond $\omega = 1/\xi$ controls any change in the aliasing effects as $\tau_1 \times \tau_2$ increases beyond $N_1 \times N_2$.

No qualitative changes were visible in the images reproduced in Figures 4.5 and 4.6 as the parameters of Tables 4.4 and 4.5 were varied, except in the control case $\beta \approx 0$, that is, the case in which the acquisition filter was an all-pass filter.

A comparison from Figure 4.8 of the formation of $\hat{R}_a$ in the three cases in which $\tau_1 \times \tau_2$ exceeds $N_1 \times N_2$ suggests only one plausible explanation for the observed results. Because the size of the matrix representation for $\hat{R}_a$ was kept constant, the fidelity metric $\|r_a\|$ was computed from the sum of the square of the magnitudes of $8N_1 \times 8N_2$ coefficients in all four cases. As $\|r_a\|$ remained constant for $\tau_1 \times \tau_2 \geq 2N_1 \times 2N_2$, it is logical to assume that the matrix representation of $\hat{R}_a$ was negligibly different in those three cases. Whether it is surmised that the reconstruction filter may have eliminated all frequencies beyond $|2N_1| \times |2N_2|$, or accepted that all the coefficients in the matrix representation are non-zero, the conclusion is compellingly the same. If the coefficients corresponding to the frequencies beyond $|2N_1| \times |2N_2|$ were zero, each $N_1 \times N_2$ grid square in the $4N_1 \times 4N_2$ area centered on $[0, 0]$ must be the same in each of the three cases. The effect of the cascaded component

[1] Each Fourier coefficient of the $N_1 \times N_2$ matrix representation which results from frequency folding might be greater than or less than the value obtained with other values of $\tau_1 \times \tau_2$ due to the summation of positive and/or negative coefficients.

was identical in each case (within the $4N_1 \times 4N_2$ area centered on $[0,0]$), therefore, the $N_1 \times N_2$ matrix representation formed by frequency folding from the area bounded by the representation passband also had to be the same in each case. For this to be true, all frequencies beyond $|2N_1| \times |2N_2|$ in the input scene representations had to be suppressed prior to frequency folding. Likewise, if all coefficients in the matrix representation of $\hat{R}_a$ were non-zero, each $N_1 \times N_2$ grid square in the area beyond the frequencies $|2N_1| \times |2N_2|$ must have been the same in each of the three cases. For this to be true, all frequencies beyond $|2N_1| \times |2N_2|$ in the input scene representations had to be suppressed prior to frequency folding.

Because filter parameters that simulate the realistic range of blurring were included in the study, it can be concluded that blurring due to acquisition imposes a limit on the effects of aliasing due to sampling. This conclusion is consistent with the experimental results obtained. In contrast to the quantitative results obtained with a Gaussian filter with parameters in the realistic range, the results obtained when blurring at acquisition was eliminated ($\hat{H}(1/2\xi) = 1.0$ in Table 4.5), show $\|r_a\|$ to be constantly increasing, for all input scenes, as $\tau_1 \times \tau_2$ increases from $N_1 \times N_2$ to $4N_1 \times 4N_2$; incidentally negating the possibility that all frequencies beyond $|2N_1| \times |2N_2|$ were eliminated at reconstruction. Similarly, when blurring at acquisition was eliminated, visible changes occurred in both the reconstructed scene and its aliased component, as $\tau_1 \times \tau_2$ increased from $N_1 \times N_2$ to $4N_1 \times 4N_2$, although none were apparent with a Gaussian acquisition filter with parameters in the realistic range.

These results obviate the need to include options in which $\tau_1 \times \tau_2 > 2N_1 \times 2N_2$ in the environment and, if that restriction were imposed, it would limit the size of all data structures to a maximum of $4N_1 \times 4N_2$. This limitation would have benefits both in terms of CPU time and memory. The $\tau_1 \times \tau_2 = 3N_1 \times 3N_2$ and $4N_1 \times 4N_2$ options have been retained for the benefit of the dubious who may wish to experiment for themselves, accepting the consequent time and memory penalties. The result, however, is of consequence not only to the implementation of the environment but also to the implementation of the c/d/c model in general.

4.2.2 Parameters affecting data structure sizes

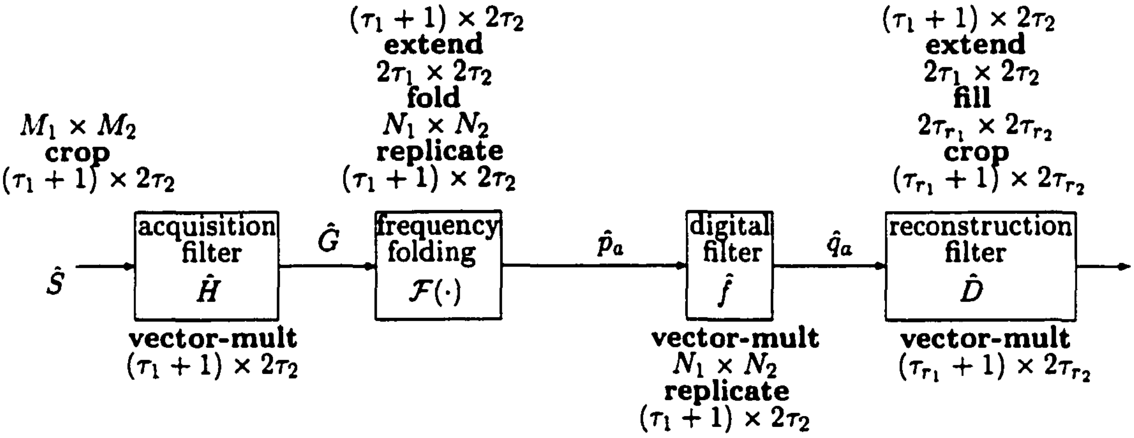


Figure 4.9: Maximum matrix sizes involved in transform processing (aliased component).

Figures 4.9, 4.10, and 4.11 show the data structure sizes required to process the Fourier matrix representation of an input scene through the c/d/c system model and display the aliased component of the selected image in the desired domain. It will be noted from Figure 4.11 that selection of the aliased component of p', p or q in the spatial domain results in the display of the inverse Fourier transform of the matrix representation band-limited by the sampling passband parameters. Even though the aperiodic (infinite) matrix representation

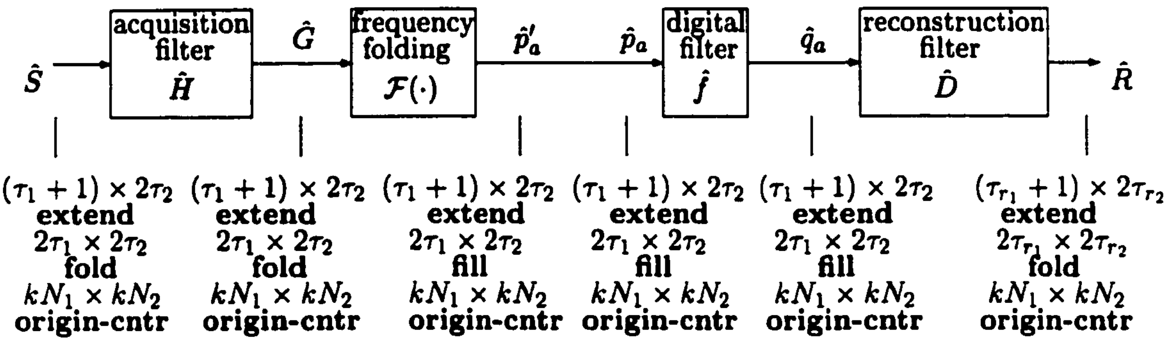

$\hat{S}$
acquisition filter $\hat{H}$
$\hat{G}$
frequency folding $\mathcal{F}(\cdot)$
$\hat{p}'_a$
$\hat{p}_a$
digital filter $\hat{f}$
$\hat{q}_a$
reconstruction filter $\hat{D}$
$\hat{R}$
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ fold $kN_1\times kN_2$ origin-cntr
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ fold $kN_1\times kN_2$ origin-cntr
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ fill $kN_1\times kN_2$ origin-cntr
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ fill $kN_1\times kN_2$ origin-cntr
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ fill $kN_1\times kN_2$ origin-cntr
$(\tau_{r_1}+1)\times 2\tau_{r_2}$ extend $2\tau_{r_1}\times 2\tau_{r_2}$ fold $kN_1\times kN_2$ origin-cntr

Figure 4.10: Maximum matrix sizes involved in frequency domain display (aliased component).

cannot be physically realized, this allows the user to visualize, prior to reconstruction, that element of the aliased component least likely to be eliminated by the reconstruction filter.

Attempts to increase the band-limits beyond the sampling passband in this instance result in "sampled" (dot matrix) images due to the underlying $N_1 \times N_2$ periodicity within the aperiodic matrix representation. That is, because $\hat{p}_a$ is the difference of $\hat{p}'$, which is periodic with period $N_1 \times N_2$, and $\hat{G}$, which is aperiodic but with most of the significant coefficients contained within the $N_1 \times N_2$ region centered at $[0,0]$, $\hat{p}_a$ has the characteristic of an $N_1 \times N_2$ periodic matrix representation except in the region affected by the significant coefficients of $\hat{G}$. If an inverse Fourier transformation were performed on a matrix representation

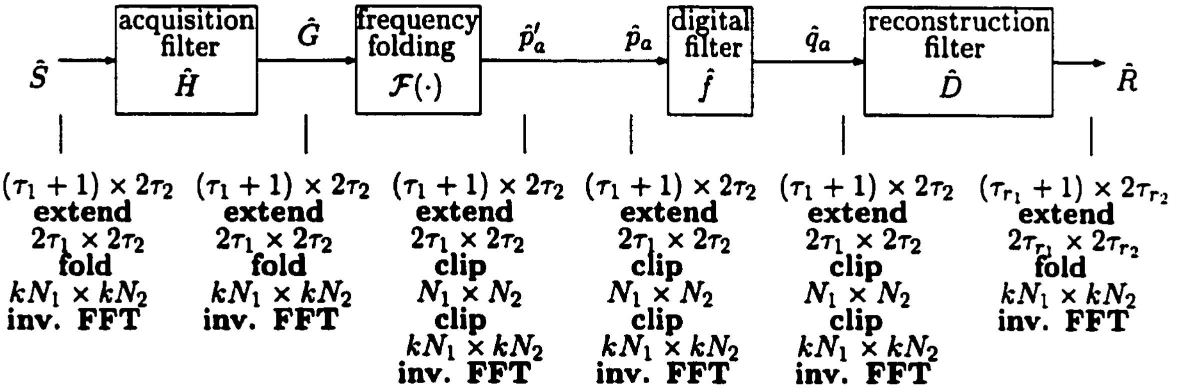

$\hat{S}$
acquisition filter $\hat{H}$
$\hat{G}$
frequency folding $\mathcal{F}(\cdot)$
$\hat{p}'_a$
$\hat{p}_a$
digital filter $\hat{f}$
$\hat{q}_a$
reconstruction filter $\hat{D}$
$\hat{R}$
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ fold $kN_1\times kN_2$ inv. FFT
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ fold $kN_1\times kN_2$ inv. FFT
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ clip $N_1\times N_2$ clip $kN_1\times kN_2$ inv. FFT
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ clip $N_1\times N_2$ clip $kN_1\times kN_2$ inv. FFT
$(\tau_1+1)\times 2\tau_2$ extend $2\tau_1\times 2\tau_2$ clip $N_1\times N_2$ clip $kN_1\times kN_2$ inv. FFT
$(\tau_{r_1}+1)\times 2\tau_{r_2}$ extend $2\tau_{r_1}\times 2\tau_{r_2}$ fold $kN_1\times kN_2$ inv. FFT

Figure 4.11: Maximum matrix sizes involved in spatial domain display (aliased component).

that contained four replications of the representation of a scene, every other value in the generated spatial domain scene would be zero. This is because the Fourier transformation recognizes the duplicated data. Likewise then, when an inverse Fourier transformation is performed on a matrix representation of $\hat{p}_a$ band-limited to a dimension greater than $N_1 \times N_2$, the multiple occurrences of the similar $N_1 \times N_2$ regions produce an image with a dot matrix appearance.

The parameters which affect data structure size are, therefore, the same as those which affect response time, namely the sampling grid size, the representation and reconstruction passband parameters, and the display size. It should be noted, as before, that all of these parameters are related to $N_1 \times N_2$.

4.2.3 Memory usage optimization

	composite image				min.
	input	$\bar{S}, \bar{G}, \bar{R}$	$\hat{p}', \hat{p}, \hat{q}$	display	$\bar{S}$
elements in data structure	M_1M_2 2^{20}	$2\tau_1\tau_2$ 2^{19}	N_1N_2 2^{16}	$16N_1N_2$ 2^{20}	$2\tau_1\tau_2$ 2^{19}
bytes/element	2^4	2^4	2^4	2^0	2^4
bytes/data structure	2^{24} 16MB	2^{23} 8MB	2^{20} 1MB	2^{20} 1MB	2^{23} 8MB
data structures	1	3	3	6	1
bytes/type	16MB	24MB	3MB	6MB	8MB
total w/o disp.	43MB			N/A	8MB
total w. display	49MB				N/A

Table 4.6: Sampling Grid $N_1 \times N_2 = 256 \times 256$ (composite image)

To minimize memory requirements:

- spatial domain matrix representations are not stored;

	composite image				min.
	input	$\hat{S},\hat{G},\hat{R}$	$\hat{p}',\hat{p},\hat{q}$	display	$\hat{S}$
elements in data structure	M_1M_2 2^{22}	$2\tau_1\tau_2$ 2^{21}	N_1N_2 2^{18}	$16N_1N_2$ 2^{22}	$2\tau_1\tau_2$ 2^{21}
bytes/element	2^4	2^4	2^4	2^0	2^4
bytes/data structure	2^{26} 67MB	2^{25} 34MB	2^{22} 4MB	2^{22} 4MB	2^{25} 34MB
data structures	1	3	3	6	1
bytes/type	67MB	102MB	12MB	24MB	34MB
total w/o disp.	181MB			N/A	34MB
total w. display	205MB				N/A

Table 4.7: Sampling Grid $N_1 \times N_2 = 512 \times 512$ (composite image)

- only two quadrants of the frequency domain matrix representations for $\hat{S}$, $\hat{G}$ and $\hat{R}$ (also $\hat{p}'$, $\hat{p}$ and $\hat{q}$ when cascaded or aliased components are being computed) are stored:

- frequency domain coefficients for the other two quadrants of $\hat{S}$, $\hat{G}$ and $\hat{R}$ (also $\hat{p}'$, $\hat{p}$ and $\hat{q}$ when appropriate) are generated by complex conjugacy when required for computation;

- spatial domain matrix representations are computed from 4-quadrant frequency domain matrix representations by inverse Fourier transform when required for display:

- the possible values of the user-selected parameters,

  - sampling grid, $N_1 \times N_2$,

  - representation passband parameters, τ_1, τ_2,

  - reconstruction passband parameters, τ_{r_1}, τ_{r_2},

  - display size, $kN_1 \times kN_2$,

are restricted, thus imposing an upper limit on the size of data structures.

	aliased component				min.
	input	$\breve{S}, \breve{G}, \breve{R}$	$\hat{p}', \hat{p}, \hat{q}$	display	$\breve{S}$
elements in data structure	M_1M_2 2^{20}	$2\tau_1\tau_2$ 2^{19}	$2\tau_1\tau_2$ 2^{19}	$16N_1N_2$ 2^{20}	$2\tau_1\tau_2$ 2^{19}
bytes/element	2^4	2^4	2^4	2^0	2^4
bytes/data structure	2^{24} 16MB	2^{23} 8MB	2^{23} 8MB	2^{20} 1MB	2^{23} 8MB
data structures	1	3	3	6	1
bytes/type	16MB	24MB	24MB	6MB	8MB
total w/o disp.	64MB			N/A	8MB
total w. display	70MB				N/A

Table 4.8: Sampling Grid $N_1 \times N_2 = 256 \times 256$ (aliased component)

	aliased component				min.
	input	$\breve{S}, \breve{G}, \breve{R}$	$\hat{p}', \hat{p}, \hat{q}$	display	$\breve{S}$
elements in data structure	M_1M_2 2^{22}	$2\tau_1\tau_2$ 2^{21}	$2\tau_1\tau_2$ 2^{21}	$16N_1N_2$ 2^{22}	$2\tau_1\tau_2$ 2^{21}
bytes/element	2^4	2^4	2^4	2^0	2^4
bytes/data structure	2^{26} 67MB	2^{25} 34MB	2^{25} 34MB	2^{22} 4MB	2^{25} 34MB
data structures	1	3	3	6	1
bytes/type	67MB	102MB	102MB	24MB	34MB
total w/o disp.	271MB			N/A	34MB
total w. display	295MB				N/A

Table 4.9: Sampling Grid $N_1 \times N_2 = 512 \times 512$ (aliased component)

For a 256 × 256 and a 512 × 512 sampling grid, Tables 4.6 and 4.7 compare the memory required to store the Fourier coefficients at the input and output of each component of the c/d/c system model, when displaying composite images. In addition, each table shows the absolute minimum memory that would be required to store Fourier coefficients ($\hat{S}$ only) and the memory which would be required to store all 6 display data structures. Each table assumes a typical scenario: $\tau_1 \times \tau_2 = \tau_{r_1} \times \tau_{r_2} = 2N_1 \times 2N_2$; display size $= 4N_1 \times 4N_2$.

Because maximum memory requirements are invoked when an aliased component is selected for display, Tables 4.8 and 4.9 compare the maximum memory required to store the Fourier coefficients of the aliased component for a 256 × 256 and a 512 × 512 sampling grid.

Chapter 5

Verification and Validation

5.1 Model Implementation

With a complicated system model implemented at the computational level, there must always be concern about verification and validation. Verification of the implementation of the c/d/c system model (i.e., verification of the environment) was achieved by manual calculation of the frequency domain matrix representation generated when a simple scene is processed through the model. The results obtained from the environment were then compared with the manual calculations.

Validation was achieved by a more intuitive approach. If the simple input scene is a uniform square on a contrasting uniform background, and if that square can be arbitrarily small and arbitrarily located within the scene, then the effects of sampling and the dependence of those effects on sample-scene phase can be illustrated and studied. With foreknowledge of the expected effects, it is possible to compare an expected result to the environment's output. Validation of the environment would be suggested by output from the environment

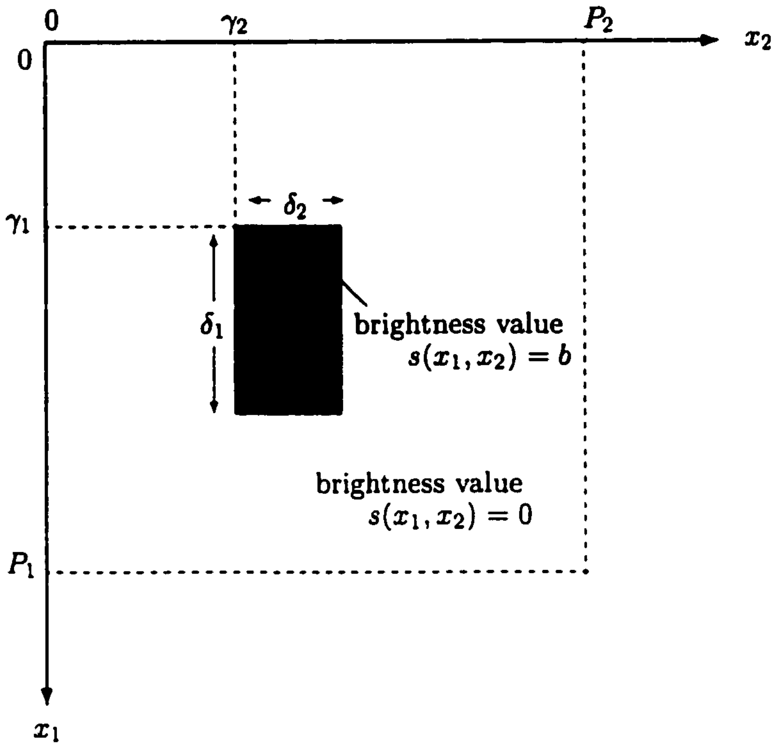


Figure 5.1: Input sce.ıe $s(x_1, x_2)$ used to verify model implementation (color inverted).

that is in accordance with intuitive expectations.

5.1.1 Verification

Verification of the environment was achieved by manual calculation of the frequency domain matrix representation generated when a simple scene is processed through the c/d/c system model. The results obtained from the environment were then compared with manual calculations to verify their accuracy.

Digitized input scenes being pre-processed for inclusion in the environment's library are represented by arrays of real brightness values b in the range $b = 0, 1, \dots, 255$. When displayed within the environment, the minimum value in a scene is assigned the brightness value 0, the maximum value in the scene is assigned the brightness value 255, and all intermediate values are correspondingly scaled. The brightness value 0 produces a black dot in a displayed dot matrix representation of the scene, the brightness value 255 produces a white dot, and brightness values between these limits produce intermediate gradations of the gray-scale. As illustrated in Figure 5.1, to define a simple 2-D scene in the frequency domain, the matrix representation of a high-contrast 2-D scene comprised of a uniform rectangle with brightness $s(x_1, x_2) = b$ on a contrasting, uniform background with brightness $s(x_1, x_2) = 0$ was calculated as follows.

Fourier Series in Two-Dimensions - Complex Form:

The complex-valued Fourier coefficients of the scene are

$$\begin{aligned}
\hat{S}[\nu_1, \nu_2] &= \frac{1}{P_1 P_2} \int_{P_1} \int_{P_2} s(x_1, x_2) \exp\left(-i2\pi\left(\frac{\nu_1 x_1}{P_1} + \frac{\nu_2 x_2}{P_2}\right)\right) dx_1 dx_2 \\
&= \frac{1}{P_1 P_2} \int_{\gamma_1}^{\gamma_1+\delta_1} \int_{\gamma_2}^{\gamma_2+\delta_2} s(x_1, x_2) \exp\left(-i2\pi\left(\frac{\nu_1 x_1}{P_1} + \frac{\nu_2 x_2}{P_2}\right)\right) dx_1 dx_2 \\
&= \frac{b}{P_1 P_2} \int_{\gamma_1}^{\gamma_1+\delta_1} \int_{\gamma_2}^{\gamma_2+\delta_2} \exp\left(-i2\pi\left(\frac{\nu_1 x_1}{P_1} + \frac{\nu_2 x_2}{P_2}\right)\right) dx_1 dx_2 \\
&= \frac{b}{P_1 P_2} \int_{\gamma_1}^{\gamma_1+\delta_1} \exp\left(-i2\pi\frac{\nu_1 x_1}{P_1}\right) dx_1 \int_{\gamma_2}^{\gamma_2+\delta_2} \exp\left(-i2\pi\frac{\nu_2 x_2}{P_2}\right) dx_2
\end{aligned}$$

for $\nu_1 = 0, \pm 1, \pm 2, \ldots, \pm \tau_1$ and $\nu_2 = 0, \pm 1, \pm 2, \ldots, \pm \tau_2$. To simplify this equation. let $a = -2\pi\nu/P$, and note that

$$\int \exp(iax)dx = \int (\cos(ax) + i\sin(ax))dx = \frac{1}{ia}\exp(iax)$$

Therefore,

$$
\begin{aligned}
\hat{S}[\nu_1,\nu_2] &= \frac{b}{P_1P_2}\left[\frac{P_1}{-i2\pi\nu_1}\exp(-i2\pi\nu_1 x_1/P_1)\right]_{\gamma_1}^{\gamma_1+\delta_1}\left[\frac{P_2}{-i2\pi\nu_2}\exp(-i2\pi\nu_2 x_2/P_2)\right]_{\gamma_2}^{\gamma_2+\delta_2} \\
&= \frac{b}{4\pi^2\nu_1\nu_2}\left[\frac{1}{-i}\exp(-i2\pi\nu_1 x_1/P_1)\right]_{\gamma_1}^{\gamma_1+\delta_1}\left[\frac{1}{-i}\exp(-i2\pi\nu_2 x_2/P_2)\right]_{\gamma_2}^{\gamma_2+\delta_2} \\
&= \frac{b}{4\pi^2\nu_1\nu_2}\left(\frac{1}{-i}(\exp(-i2\pi\nu_1\gamma_1/P_1)\exp(-i2\pi\nu_1\delta_1/P_1) - \exp(-i2\pi\nu_1\gamma_1/P_1))\right) \\
&\qquad \left(\frac{1}{-i}(\exp(-i2\pi\nu_2\gamma_2/P_2)\exp(-i2\pi\nu_2\delta_2/P_2) - \exp(-i2\pi\nu_2\gamma_2/P_2))\right) \\
&= \frac{b}{4\pi^2\nu_1\nu_2}\left(\frac{1}{i}(\exp(-i2\pi\nu_1\gamma_1/P_1) - \exp(-i2\pi\nu_1\gamma_1/P_1)\exp(-i2\pi\nu_1\delta_1/P_1))\right) \\
&\qquad \left(\frac{1}{i}(\exp(-i2\pi\nu_2\gamma_2/P_2) - \exp(-i2\pi\nu_2\gamma_2/P_2)\exp(-i2\pi\nu_2\delta_2/P_2))\right) \qquad (5.1)
\end{aligned}
$$

To express Equation 5.1 in terms of real and imaginary parts, once again let $a = -2\pi\nu/P$. then

$$
\begin{aligned}
\frac{1}{i}(\exp(ia\gamma) - \exp(ia\gamma)\exp(ia\delta)) &= \frac{1}{i}\exp(ia\gamma)(1 - \exp(ia\delta)) \\
&= \frac{1}{i}(\cos(a\gamma) + i\sin(a\gamma))(1 - \cos(a\delta) - i\sin(a\delta)) \\
&= \frac{1}{i}(\cos(a\gamma) - \cos(a\gamma)\cos(a\delta) - i\cos(a\gamma)\sin(a\delta) + \\
&\qquad i\sin(a\gamma) - i\sin(a\gamma)\cos(a\delta) + \sin(a\gamma)\sin(a\delta)) \\
&= -i\cos(a\gamma) + i\cos(a\gamma)\cos(a\delta) - \cos(a\gamma)\sin(a\delta) + \\
&\qquad \sin(a\gamma) - \sin(a\gamma)\cos(a\delta) - i\sin(a\gamma)\sin(a\delta)
\end{aligned}
$$

In this way, Equation 5.1 can be written as

$$
\begin{aligned}
\hat{S}[\nu_1,\nu_2] &= \frac{b}{4\pi^2\nu_1\nu_2}(\sin(2\pi\nu_1(\gamma_1+\delta_1)/P_1) - \sin(2\pi\nu_1\gamma_1/P_1) + \\
&\qquad i\cos(2\pi\nu_1(\gamma_1+\delta_1)/P_1) - i\cos(2\pi\nu_1\gamma_1/P_1)) \\
&\qquad (\sin(2\pi\nu_2(\gamma_2+\delta_2)/P_2) - \sin(2\pi\nu_2\gamma_2/P_2) + \\
&\qquad i\cos(2\pi\nu_2(\gamma_2+\delta_2)/P_2) - i\cos(2\pi\nu_2\gamma_2/P_2)) \\
&= \frac{\sqrt{b}}{2\pi\nu_1}(\sin(2\pi\nu_1(\gamma_1+\delta_1)/P_1) - \sin(2\pi\nu_1\gamma_1/P_1) + \\
&\qquad i\cos(2\pi\nu_1(\gamma_1+\delta_1)/P_1) - i\cos(2\pi\nu_1\gamma_1/P_1)) \\
&\quad \frac{\sqrt{b}}{2\pi\nu_2}(\sin(2\pi\nu_2(\gamma_2+\delta_2)/P_2) - \sin(2\pi\nu_2\gamma_2/P_2) + \\
&\qquad i\cos(2\pi\nu_2(\gamma_2+\delta_2)/P_2) - i\cos(2\pi\nu_2\gamma_2/P_2))
\end{aligned}
\tag{5.2}
$$

The coefficient matrix $\hat{S}[\nu_1,\nu_2]$ is, therefore, separable and can be computed as the product of two coefficient vectors.

Using a small input scene, $\hat{S}$ was computed manually from Equation 5.2. Equation 2.3 was used to compute the matrix representation of the acquisition filter $\hat{H}$, then $\hat{S}$ and $\hat{H}$ were substituted into Equation 2.1 to obtain $\hat{G}$. Using Equation 2.4, $\hat{p}$ was computed. Equation 2.6 was used to compute the matrix representation of the digital filter $\hat{f}$, then $\hat{p}$ and $\hat{f}$ were substituted into Equation 2.5 to obtain $\hat{q}$. Equation 2.9 was used to compute the matrix representation of the reconstruction filter $\hat{D}$, then $\hat{q}$ and $\hat{D}$ were substituted into Equation 2.7 to obtain $\hat{R}$. The $\hat{R}$ matrix representation was compared with the results from the environment to verify the implementation.

5.1.2 Validation

Validation was achieved by a more intuitive approach. Sixty-four different input scenes, each comprised of a small, bright, uniform $\delta_1 \times \delta_2$ square (the target) on a contrasting, uniform square background, were synthesized in the frequency domain utilizing Equation (5.2). Four different sizes of target were used:

1. $\delta_1 \times \delta_2 = \xi_1/2 \times \xi_2/2$,
2. $\delta_1 \times \delta_2 = \xi_1 \times \xi_2$,
3. $\delta_1 \times \delta_2 = 2\xi_1 \times 2\xi_2$,
4. $\delta_1 \times \delta_2 = 3\xi_1 \times 3\xi_2$.

In each case, ξ_1 and ξ_2 are respectively the vertical and horizontal inter-sample distances of the sampling grid. For each target size, 16 different input scenes were defined with the scenes differing from one another only in the sample-scene phase, i.e., only in the location of the target relative to the sampling grid.

5.2 Results

In the discussions that follow, let $\xi = \xi_1 = \xi_2$. A fixed set of c/d/c system model components and system parameters was specified as the test configuration.

1. $N_1 = N_2 = 16$.
2. $\xi = \xi_1 = \xi_2 = 1$.
3. $\tau_1 \times \tau_2 = 2N_1 \times 2N_2$.

4. $\tau_{r_1} \times \tau_{r_1} = 2N_1 \times 2N_2$.

5. Display size = $4N_1 \times 4N_2$.

6. Filters

   (a) The separable Gaussian acquisition filter was defined by Equation 2.2 with $\beta_1, \beta_2 = 2.5$ (see Figure 5.1);

   (b) The separable parametric cubic reconstruction filter was defined by Equation 2.8 with $\alpha_1, \alpha_2 = -0.5$;

   (c) The separable modified inverse digital filter was defined by Equation 2.6 with $\lambda_1, \lambda_2 = 0.1$.

7. No ringing suppression was applied to the input scene.

8. Signal to noise ratio was sufficiently high to make noise input negligible.

The filters and filter parameters chosen are typical of those found in an end-to-end digital imaging system. The sampling grid size was minimized in order to obtain reconstructed images (without additional aliasing artifacts) that would be small enough to print 32 to a page for inclusion in this dissertation. The passband parameters were set in accordance with the results of the study described in Chapter 4 and the display size set accordingly, again to avoid the introduction of aliasing artifacts at the display stage.

Once the images required for comparison had been obtained, the passband parameters were reset to $4N_1 \times 4N_2$ and the display size to the maximum of $8N_1 \times 8N_2$ to enable the capture of the highest resolution 1-D information available for construction of the illustrative graphs included in this chapter.

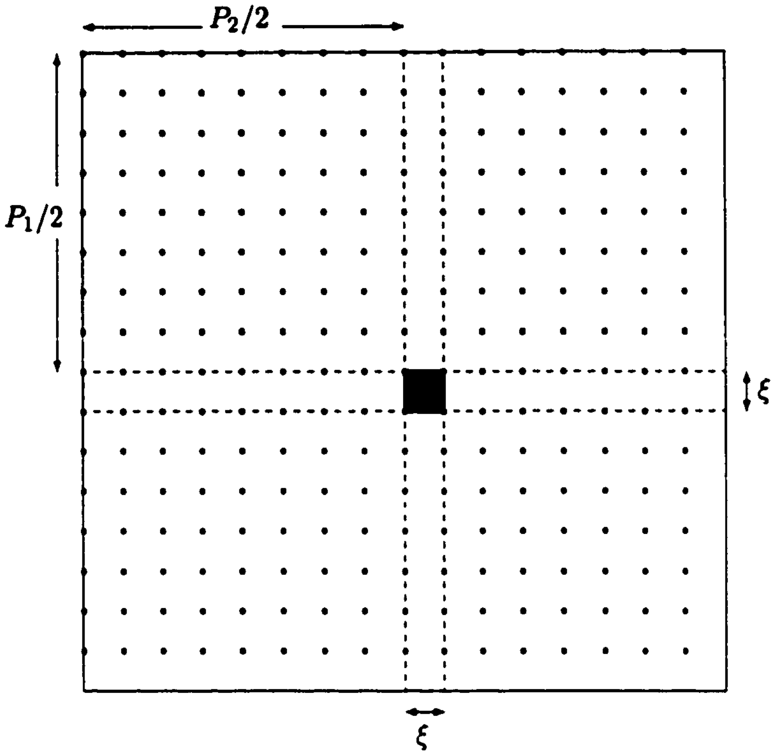


Figure 5.2: Stylized representation of $\xi \times \xi$ input scene 1 (color inverted).

5.2.1 $\xi \times \xi$ targets

Sixteen different input scenes, each comprised of a $\xi \times \xi$ white target on a black background. were synthesized in the frequency domain. The top left-hand corner of the white target was positioned in the center of the black background in the first input scene, as shown in Figure 5.2. Subsequent scenes were created by progressively subjecting the $\xi \times \xi$ target to

Numbered input scenes		horizontal right shift			
		0	$\xi/4$	$\xi/2$	$3\xi/4$
vertical down shift	0	1	2	3	4
	$\xi/4$	5	6	7	8
	$\xi/2$	9	10	11	12
	$3\xi/4$	13	14	15	16

Table 5.1: Sample-scene phase shift of target in numbered input scenes relative to input scene 1.

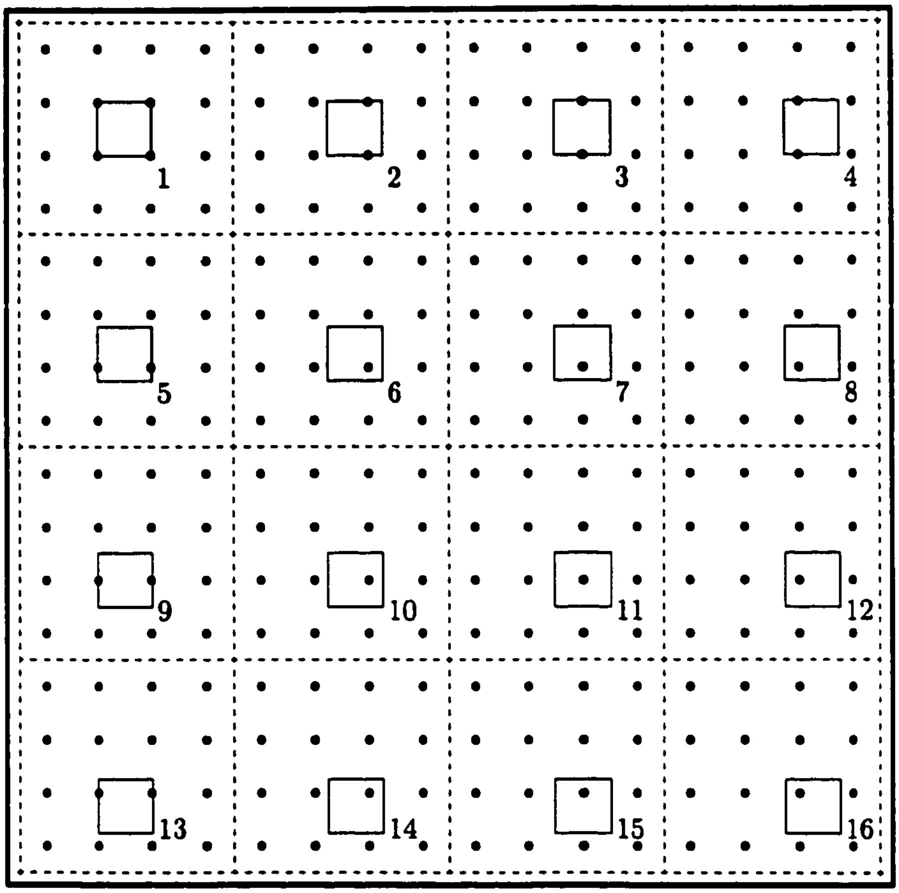


Figure 5.3: Placement of $\xi \times \xi$ targets on sampling grid.

three right shifts of $\xi/4$ to create the first four input scenes, then subjecting each of these to three downward shifts of $\xi/4$ to create 12 more input scenes. All 16 input scenes with a $\xi \times \xi$ target are defined by Table 5.1 which identifies the sample-scene phase shift of each $\xi \times \xi$ target relative to the target's position in Figure 5.2. Figure 5.3 graphically depicts the resultant position of the target in each input scene relative to the sampling grid. It should be noted that the 16 targets are combined in the same figure for illustrative purposes only. That is, each input scene contained only one target.

In Figures 5.4 to 5.7, the 16 different images are combined for ease of comparison. The numbered images are packed as shown in Figure 5.8 and the sample-scene phase shift

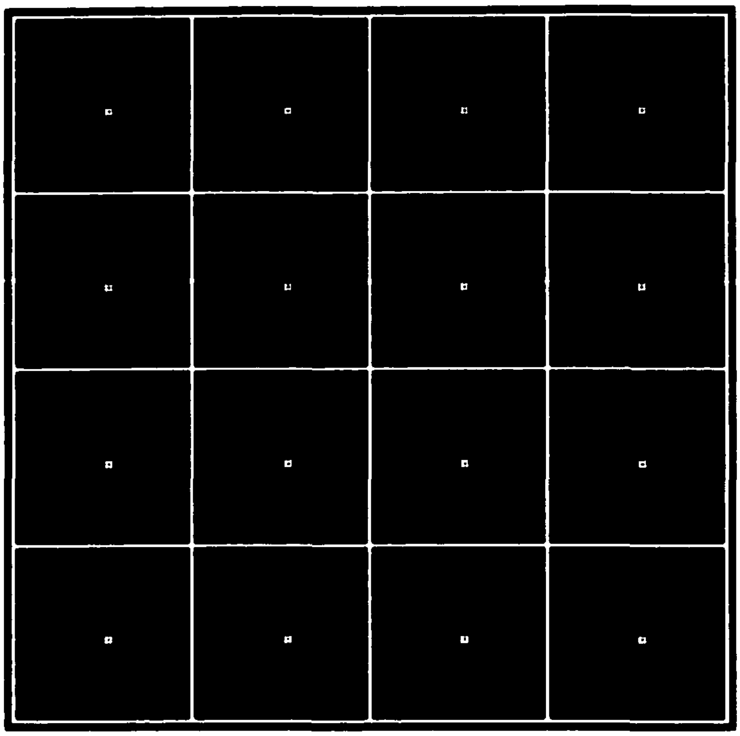


Figure 5.4: Input scenes with $\xi \times \xi$ targets.

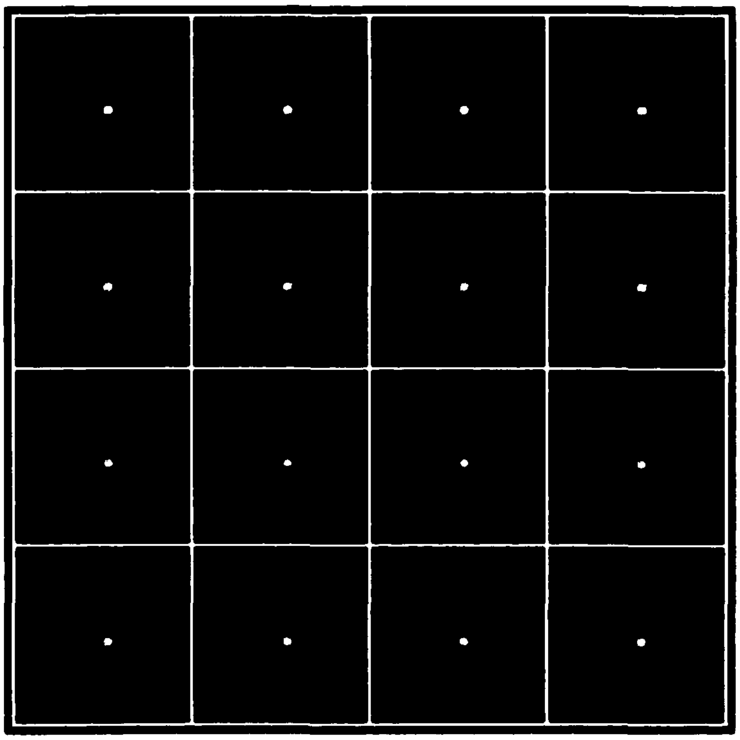


Figure 5.5: Blurred input scenes with $\xi \times \xi$ targets.

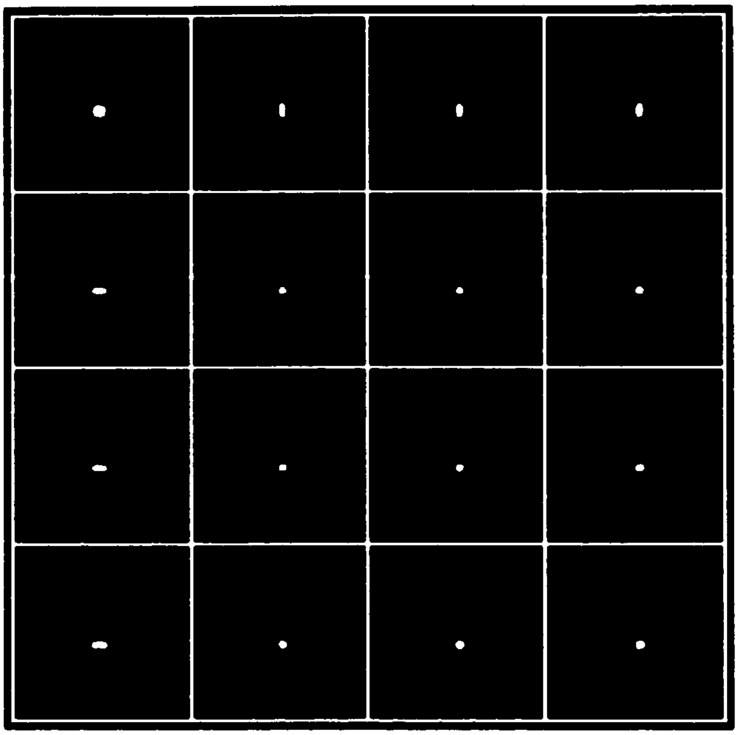


Figure 5.6: Reconstructed $\xi \times \xi$ targets.

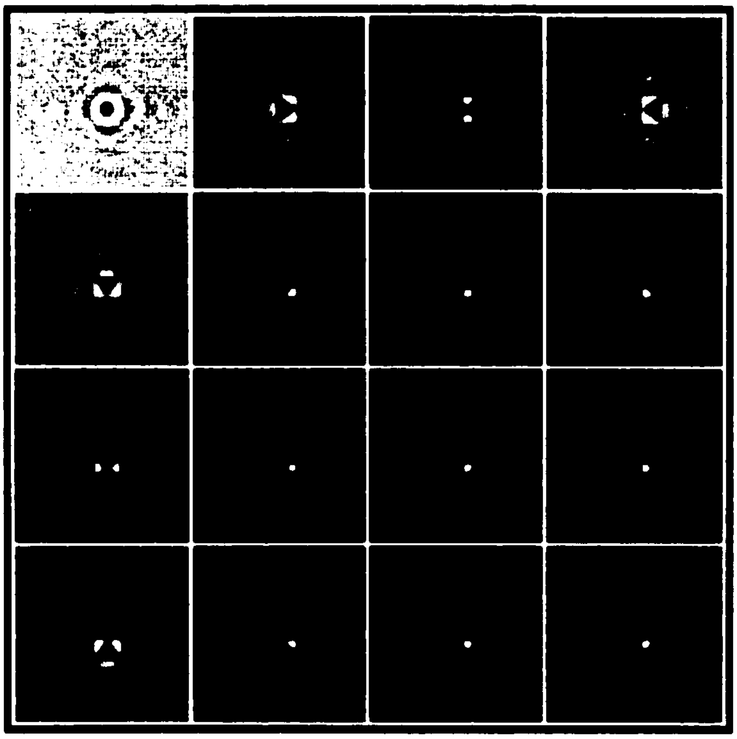


Figure 5.7: Aliased component of reconstructed $\xi \times \xi$ targets.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Figure 5.8: Numbered images in Figures 5.4 to 5.7.

relative to Figure 5.2 (input scene 1) corresponds to Table 5.1 and Figure 5.3. Figures 5.4 to 5.7 display the 16 images at three different stages of the end-to-end system.

1. Figure 5.4—synthesized input scenes, s;
2. Figure 5.5—blurred, pre-sampling images, g;
3. (a) Figure 5.6—reconstructed images, r;

   (b) Figure 5.7—aliased components, r_a.

Prior to sampling (Figures 5.4 and 5.5), the image of the target is identical in all 16 cases, independent of sample-scene phase. After sampling and reconstruction (Figures 5.6 and 5.7), the representation of the target varies significantly with sample-scene phase.

Inspection of Figure 5.3 indicates that there are only six unique conditions which result from sample-scene phase shift among the 16 2-D input scenes. Identifying scenes by the numbering system established in Table 5.1, it is evident that these groupings correspond exactly to the results which can be observed in the reconstructed images of Figures 5.6 and 5.7. Specifically, from Figure 5.3, it can be observed that:

1. scenes 1 and 11 are unique;
2. by rotation, scenes 3 and 9 have the same sample-scene phase;

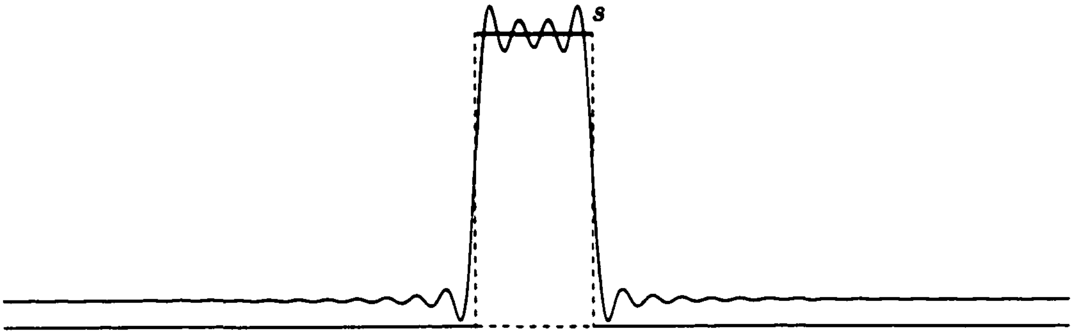


Figure 5.9: Fourier representation of a pulse train function with ringing due to truncation.

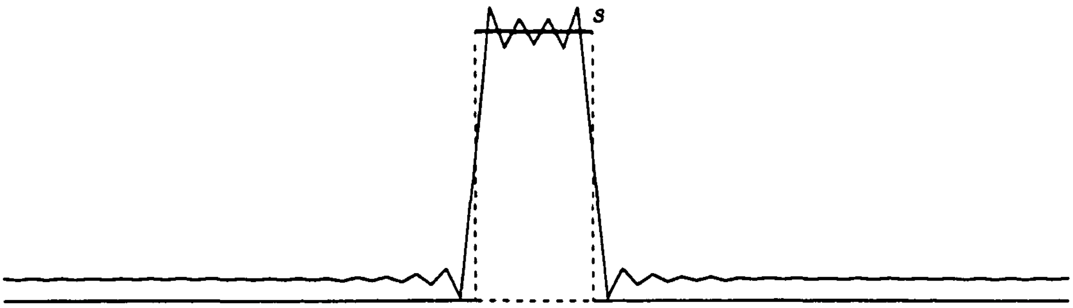


Figure 5.10: Stylized Fourier representation of a pulse train function with ringing.

3. the remaining scenes can be divided into three groups of four, each group of four having the same sample-scene phase on rotation. The groups are
   (a) 2, 4, 5, and 13;
   (b) 6, 8, 14, and 16;
   (c) 7, 10, 12, and 15.

Equation (5.2) defines each 2-D input scene as the product of two coefficient vectors, each of which represents a pulse train function, such as is shown in Figure 5.9. The effects of sample-scene phase shift on the 1-D functions as they are processed through the c/d/c model can be used as a predictor for the results expected from the 2-D implementation. The 1-D functions corresponding to the coefficient vectors were studied with the pulse located at each of the four possible positions relative to the sampling grid.

The band-limited Fourier series representation of the function, s, is subject to ringing, and no smoothing was applied to the synthesized input scenes to suppress these effects. The ringing artifacts are represented in the graphs to follow in the stylized manner exhibited in Figure 5.10 because Figures 5.10 to 5.16, Figures 5.23 to 5.28, Figures 5.35 to 5.40 and Figures 5.47 to 5.52 were generated using output from the environment, which supports a maximum of 8 coefficients per sampling interval in the matrix representation of a scene.

Figures 5.11 to 5.13 demonstrate one reason why the 2-D scenes vary one from another after sampling. Each figure shows a 1-D horizontal cross-section through the $\xi \times \xi$ high-contrast target of a 2-D input scene. Each figure is comprised of

1. the ideal ξ-width pulse train function (as solid straight lines);

2. the band-limited Fourier series representation of that function, s (as in Figure 5.10):

3. the output from the acquisition filter, g (as a smooth curve);

4. the sampled sequence, p' (denoted by bullets).

No two of the p' sequences are the same due to the sample-scene phase shift in the location of the ξ-width pulse.

Figures 5.14 to 5.16 show how the aliased component, r_a, is affected by sample-scene phase shift. In this case, each figure is comprised of

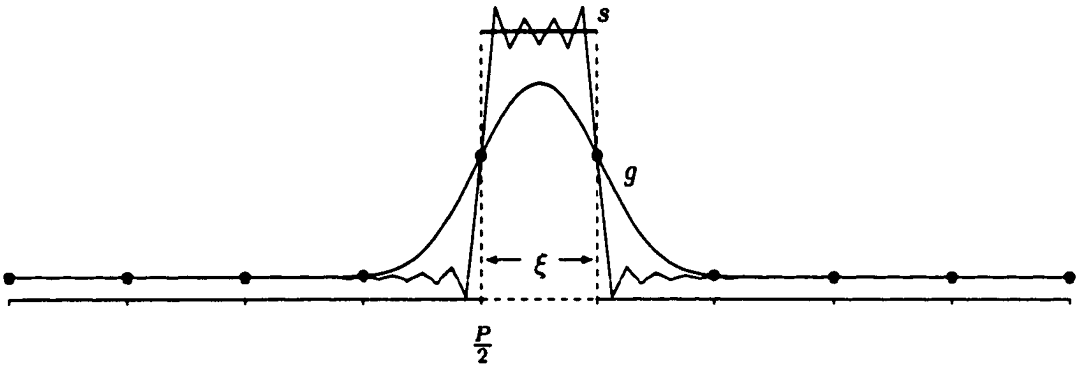


Figure 5.11: Pulse train *s* and *g* functions — sample-scene phase shift centered pulse between sampling points.

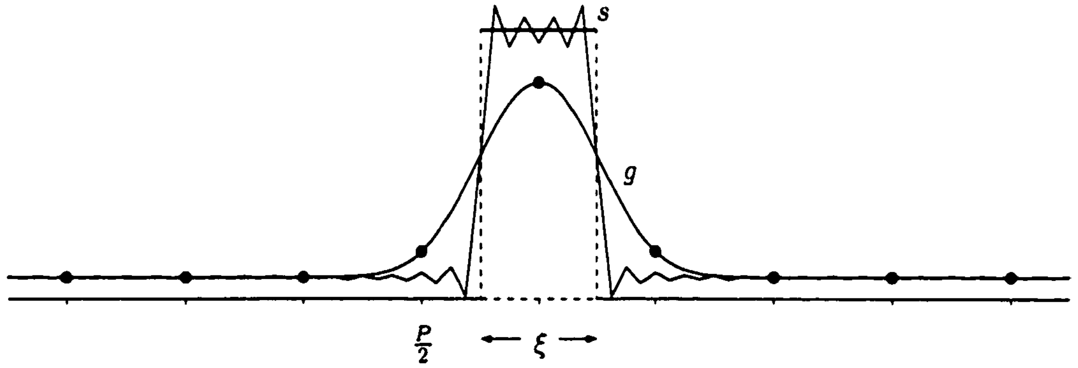


Figure 5.12: Pulse train *s* and *g* functions — sample-scene phase shift centered pulse on sampling point.

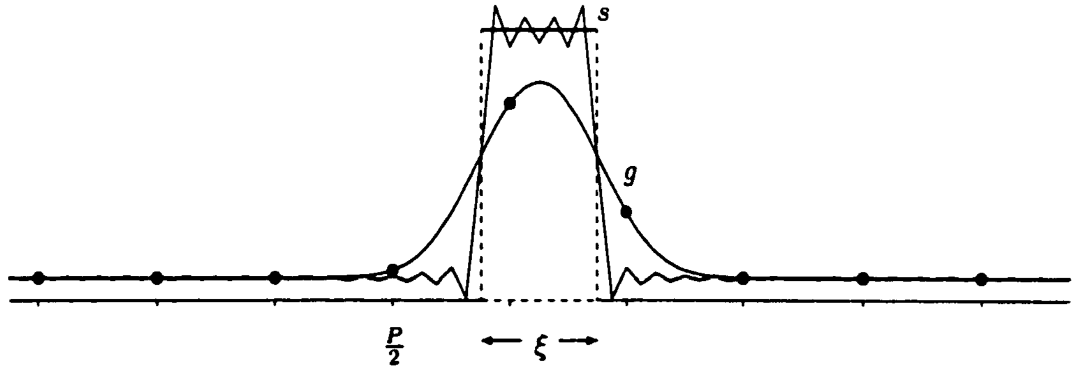


Figure 5.13: Pulse train *s* and *g* functions — sample-scene phase shift positioned pulse asymmetrically relative to sampling points.

1. the ideal ξ-width pulse train function;
2. the sampled sequence, p' (denoted by bullets);
3. the output sequence, q, from the digital filter (denoted by open circles);
4. the end-to-end system output, r (as a smooth curve);
5. the aliased component of the end-to-end system output, r_a (as a smooth curve).

Input scene number	pre-sampling 1-D functions	post-sampling 1-D functions
1,5,9,13	Figure 5.11	Figure 5.14
3,7,11,15	Figure 5.12	Figure 5.15
4,8,12,16	Figure 5.13	Figure 5.16
2,6,10,14	v.m.i. of Figure 5.13	v.m.i. of Figure 5.16

Table 5.2: Relationship between numbered input scenes and 1-D figures (v.m.i. denotes "vertical mirror image").

Table 5.2 identifies the provenance of the six 1-D figures. In addition, corresponding 1-D figures derived from input scene 2 *et al.* produce the vertical mirror image of Figures 5.13 and 5.16. From Figures 5.14 to 5.16, it will be noted that no two of the q sequences are the same due to the sample-scene phase shift in the location of the ξ-width pulse. The combined effects of sample-scene phase shift and aliasing on the reconstructed image might be expected therefore to be significant, at least when the dimension of the target is comparable to a unit area of the sampling grid, as in this case.

As specified in Table 5.1, only four horizontal and the same four vertical sample-scene phase shifts were implemented to generate the 16 2-D input scenes with a $\xi \times \xi$ target. Equivalently, each pulse train function was subjected to one of only four possible sample-scene phase shifts.

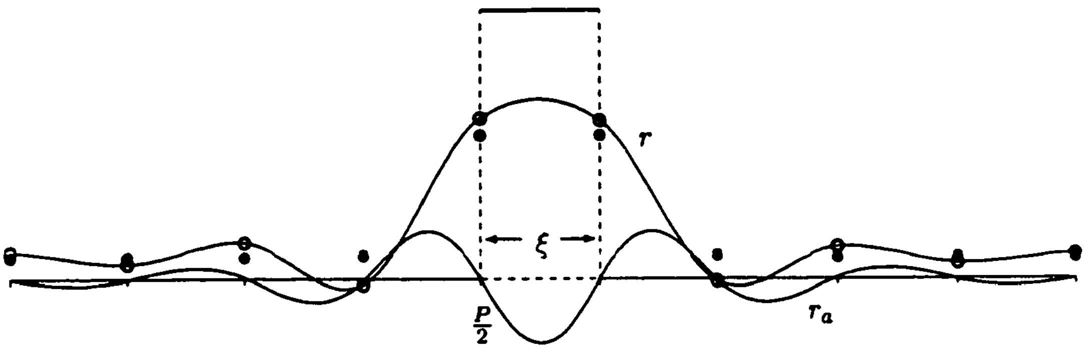


Figure 5.14: Pulse train r and r_a functions — sample-scene phase shift centered pulse between sampling points.

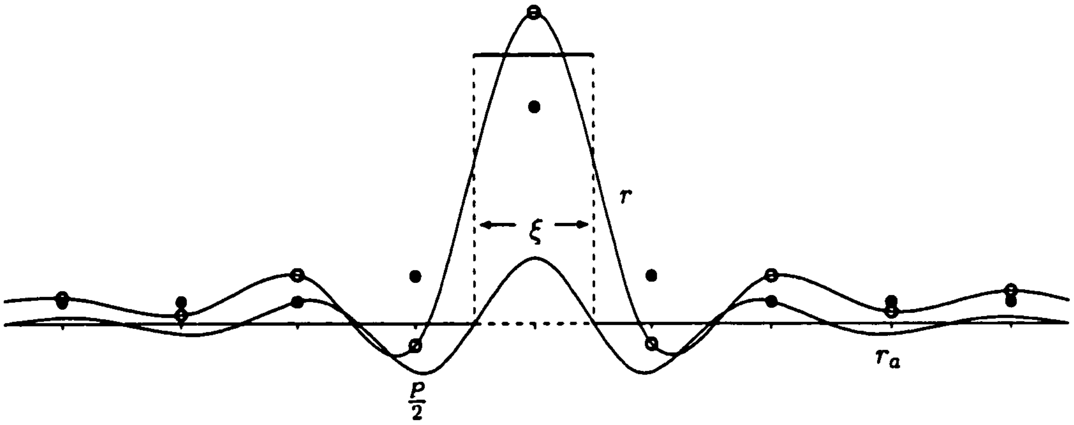


Figure 5.15: Pulse train r and r_a functions — sample-scene phase shift centered pulse on sampling point.

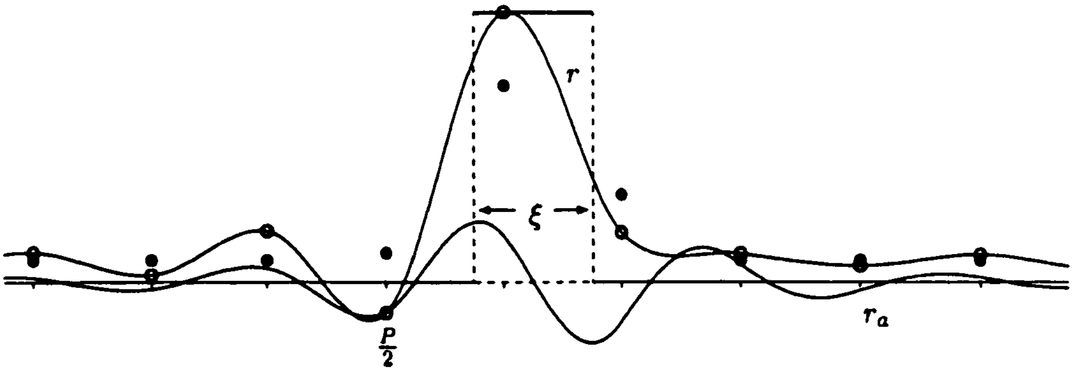


Figure 5.16: Pulse train r and r_a functions — sample-scene phase shift positioned pulse asymmetrically relative to sampling points.

1. Figure 5.11 depicts the p' sequence which results from sampling g when the midpoint of the ξ-width pulse is centered between two sampling points.

2. Figure 5.12 depicts the p' sequence which results from sampling g when the midpoint of the ξ-width pulse is centered on a sampling point.

3. Figure 5.13 and its vertical mirror image depict p' sequences which result from sampling g when the midpoint of the ξ-width pulse is asymmetrically positioned relative to the sampling points.

Reconstructed Target Prediction		corresponding
r pulse 1	r pulse 2	input scene(s)
Figure 5.14	Figure 5.14	1
Figure 5.14	v.m.i. of Figure 5.16	2, 5
Figure 5.14	Figure 5.15	3, 9
Figure 5.14	Figure 5.16	4, 13
v.m.i. of Figure 5.16	v.m.i. of Figure 5.16	6
v.m.i. of Figure 5.16	Figure 5.15	7, 10
v.m.i. of Figure 5.16	Figure 5.16	8, 14
Figure 5.15	Figure 5.15	11
Figure 5.15	Figure 5.16	12, 15
Figure 5.16	Figure 5.16	16

Table 5.3: Cross-product of r functions should predict target characteristics in corresponding reconstructed image.

Because each of the 2-D scenes is the product of two pulse train functions, one horizontal and one vertical, the information contained in Figures 5.14 to 5.16 can be used to predict the characteristics of the 2-D targets in the reconstructed images. Table 5.3 associates two graphs with each input scene on the basis of sample-scene phase shift. Because the target is symmetrical, the input scene can be rotated indiscriminately, so that only the number of combinations (as opposed to permutations) of the possible vertical and horizontal sample-scene phase shifts need be considered. The cross-product of the two r functions should, in each case, predict the characteristics of the target in the corresponding reconstructed image. However, it has already been noted that a 1-D sample-scene phase shift of $\xi/4$ results in a reconstructed function which is the vertical mirror image of that which results from a 1-D sample-scene phase shift of $3\xi/4$, and so the number of different combinations of vertical and horizontal phase-shifted pulse train functions reduces to those listed in Table 5.4.

reconstructed target prediction		corresponding
r pulse 1	r pulse 2	input scenes
Figure 5.14	Figure 5.14	1
Figure 5.14	Figure 5.16	2, 4, 5, 13
Figure 5.14	Figure 5.15	3, 9
Figure 5.15	Figure 5.15	11
Figure 5.15	Figure 5.16	7, 10, 12, 15
Figure 5.16	Figure 5.16	6, 8, 14, 16

Table 5.4: Possible combinations of phase-shifted pulse train functions.

A direct correspondence between the input scene groupings in Table 5.4 and those predicted by inspection of Figure 5.3 is immediately evident. Comparison of the reconstructed images in Figure 5.6 further corroborates the projections. It should be noted that contrast stretching over the range of brightness values in each output image was employed, thus there is no indication of the relative brightness values among images. However, the following observations contribute to the validation of the environment. (Recall that the images in Figure 5.6 are numbered as specified in Figure 5.8 with the sample-scene phase shift identified in Table 5.1.)

- Image 1 is unique. The depiction of the target in the reconstructed image appears approximately four times as large as the target in the input scene and the artifacts due to aliasing are symmetric. These observations are consistent with a 2-D image generated from two orthogonal, reconstructed pulse-train functions, r, from Figure 5.14.

- Image 11 is unique. The depiction of the target in the reconstructed image appears only slightly larger than the target in the input scene and the artifacts due to aliasing are symmetric. These observations are consistent with a 2-D image generated from two orthogonal, reconstructed pulse-train functions, r, from Figure 5.15.

- Images 3 and 9 can be rotated to produce the same reconstructed image. The depiction of the target in the reconstructed image is rectangular. The artifacts due to aliasing are symmetric about the horizontal and vertical axes. These observations are consistent with a 2-D image generated from an r function from Figure 5.14 and an orthogonal r function from Figure 5.15.

- Images 2, 4, 5, and 13 can be rotated to produce the same reconstructed image. Again the depiction of the target in the reconstructed image is rectangular. The artifacts due to aliasing are symmetric in the direction of the long side of the rectangle and asymmetric in the other direction. These observations are consistent with a 2-D image generated from an r function from Figure 5.14 and an orthogonal r function from Figure 5.16.

- Images 6, 8, 14, and 16 can be rotated to produce the same reconstructed image. The depiction of the target in the reconstructed image appears only slightly larger than the target in the input scene and the artifacts due to aliasing are asymmetric about the horizontal and vertical axes. These observations are consistent with a 2-D image generated from two orthogonal, reconstructed pulse-train functions, r, from Figure 5.16.

- Images 7, 10, 12, and 15 can be rotated to produce the same reconstructed image. The depiction of the target in the reconstructed image appears only slightly larger than the target in the input scene and the artifacts due to aliasing are symmetric in one direction and asymmetric in the other. These observations are consistent with a 2-D image generated from an r function from Figure 5.15 and an orthogonal r function from Figure 5.16.

Further corroboration of the correctness of the results can be obtained from the following observations on the comparison of the reconstructed aliased components in Figure 5.7. Once again, it should be noted that contrast stretching over the range of brightness values in each output image was employed, thus there is no indication of the relative brightness values among images.

- Image 1 is unique and symmetric, which is consistent with a 2-D image generated from two orthogonal, reconstructed pulse-train functions, r_a, from Figure 5.14.

- Image 11 is unique and symmetric, which is consistent with a 2-D image generated from two orthogonal, reconstructed pulse-train functions, r_a, from Figure 5.15.

- Images 3 and 9 can be rotated to produce the same reconstructed image. The aliasing is symmetric about the horizontal and vertical axes. These observations are consistent with a 2-D image generated from an r_a function from Figure 5.14 and an orthogonal r_a function from Figure 5.15.

- Images 2, 4, 5, and 13 can be rotated to produce the same reconstructed image. The aliasing is symmetric in one direction and asymmetric in the other direction. These observations are consistent with a 2-D image generated from an r_a function from Figure 5.14 and an orthogonal r_a function from Figure 5.16.

- Images 6, 8, 14, and 16 can be rotated to produce the same reconstructed image. The aliasing is asymmetric about the horizontal and vertical axes. These observations are consistent with a 2-D image generated from two orthogonal, reconstructed pulse-train functions, r_a, from Figure 5.16.

- Images 7, 10, 12, and 15 can be rotated to produce the same reconstructed image. The aliasing is symmetric in one direction and asymmetric in the other. These observations are consistent with a 2-D image generated from an r_a function from Figure 5.15 and an orthogonal r_a function from Figure 5.16.

5.2.2 $\xi/2 \times \xi/2$ targets

Sixteen different input scenes, each comprised of a $\xi/2 \times \xi/2$ white target on a black background, were synthesized in the frequency domain. The top left-hand corner of the target was positioned in the center of the black background in the first input scene, as shown in Figure 5.17. Subsequent scenes were created as in Section 5.2.1, and all 16 input scenes are again defined by Table 5.1 which identifies the sample-scene phase shift of each $\xi/2 \times \xi/2$ target relative to the target's position in Figure 5.17. Figure 5.18 graphically depicts the resultant position of the target in each input scene relative to the sampling grid.

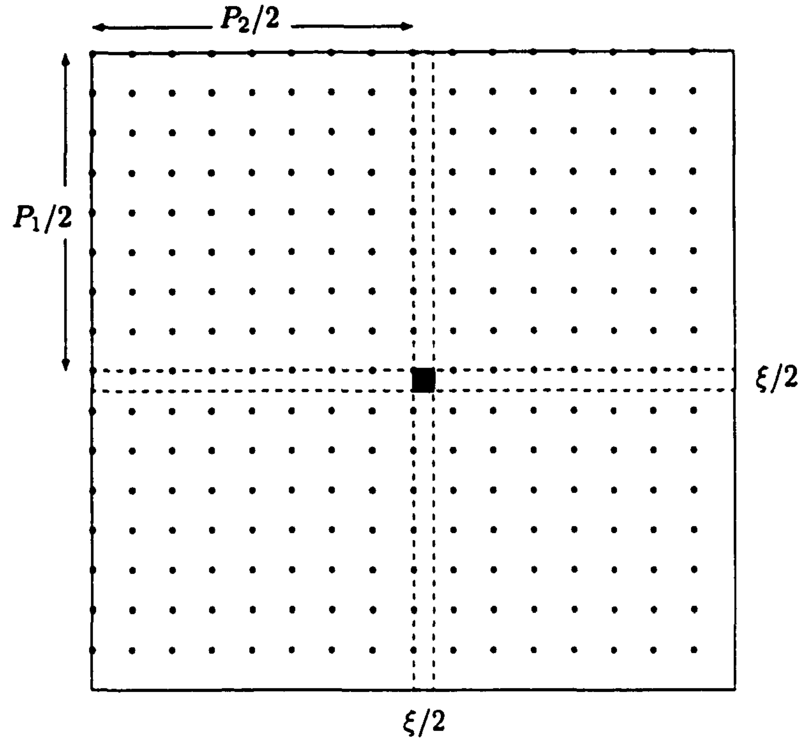


Figure 5.17: Stylized representation of $\xi/2 \times \xi/2$ input scene 1 (color inverted).

The 16 different images are combined in Figures 5.19 to 5.22 and the sample-scene phase shift relative to Figure 5.17 (input scene 1) corresponds to Table 5.1 and Figure 5.18. Figures 5.19 to 5.22 display the 16 images at the same three stages of the end-to-end system as are used in Section 5.2.1.

Once again, prior to sampling (Figures 5.19 and 5.20), the image of the target is identical in all 16 cases independent of sample-scene phase. After sampling and reconstruction (Figures 5.21 and 5.22), the representation of the target varies significantly with sample-scene phase. Table 5.5 summarizes the observations which can be made by comparing and contrasting corresponding $\xi \times \xi$ and $\xi/2 \times \xi/2$ target images.

Figures 5.23 to 5.25 demonstrate why the observed results were obtained. Each figure shows a 1-D horizontal cross-section through the $\xi/2 \times \xi/2$ high-contrast target of a 2-D

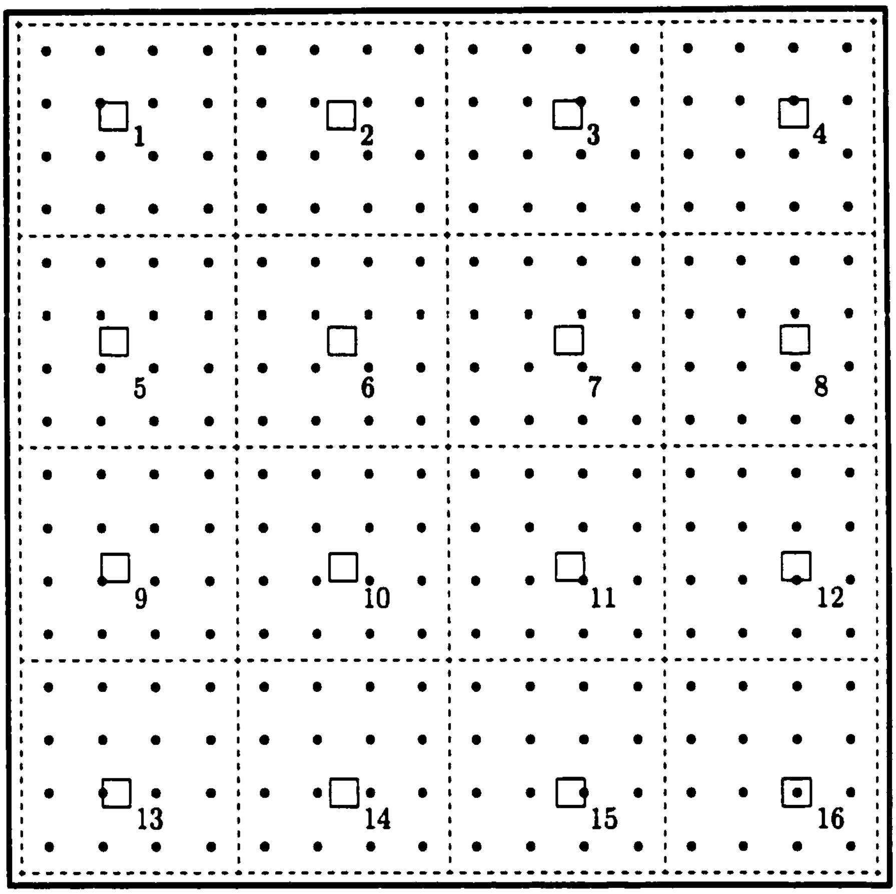


Figure 5.18: Placement of $\xi/2 \times \xi/2$ targets on sampling grid.

input scene having the same components as in Section 5.2.1. No two of the p' sequences are the same due to the sample-scene phase shift in the location of the $\xi/2$-width pulse.

Figures 5.26 to 5.28 show how the aliased component, r_a, is affected by sample-scene phase shift. Table 5.6 identifies the provenance of the six 1-D figures. In addition, corresponding 1-D figures derived from input scene 3 *et al.* produce the vertical mirror images of Figures 5.25 and 5.28. From Figures 5.26 to 5.28, it will again be noted that no two of the q sequences are the same due to the sample-scene phase shift in the location of the $\xi/2$-width pulse.

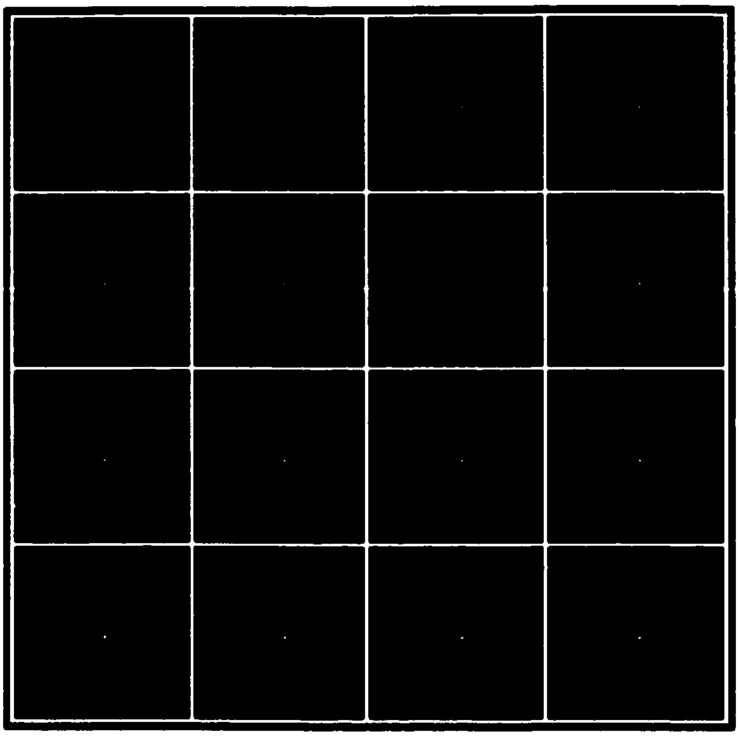


Figure 5.19: Input scenes with $\xi/2 \times \xi/2$ targets.

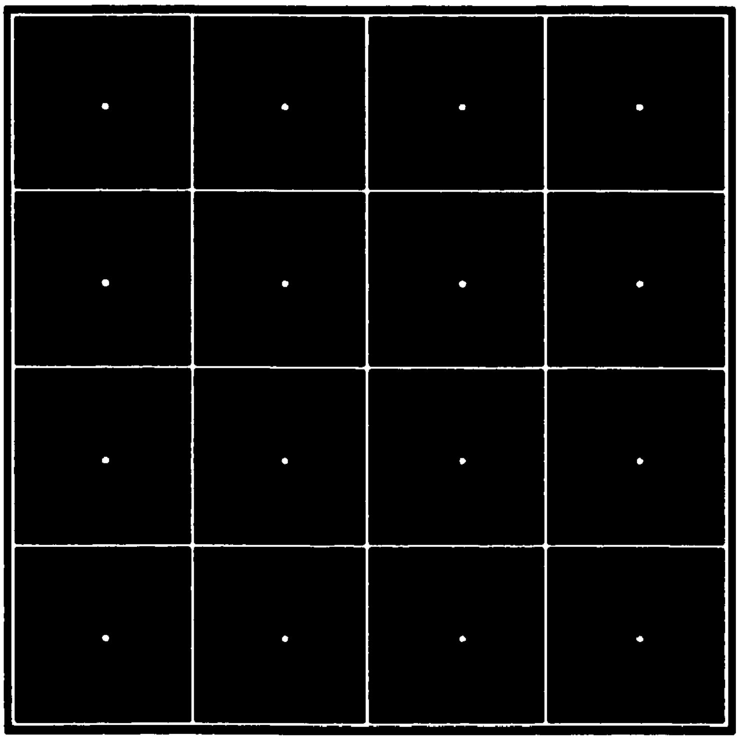


Figure 5.20: Blurred input scenes with $\xi/2 \times \xi/2$ targets.

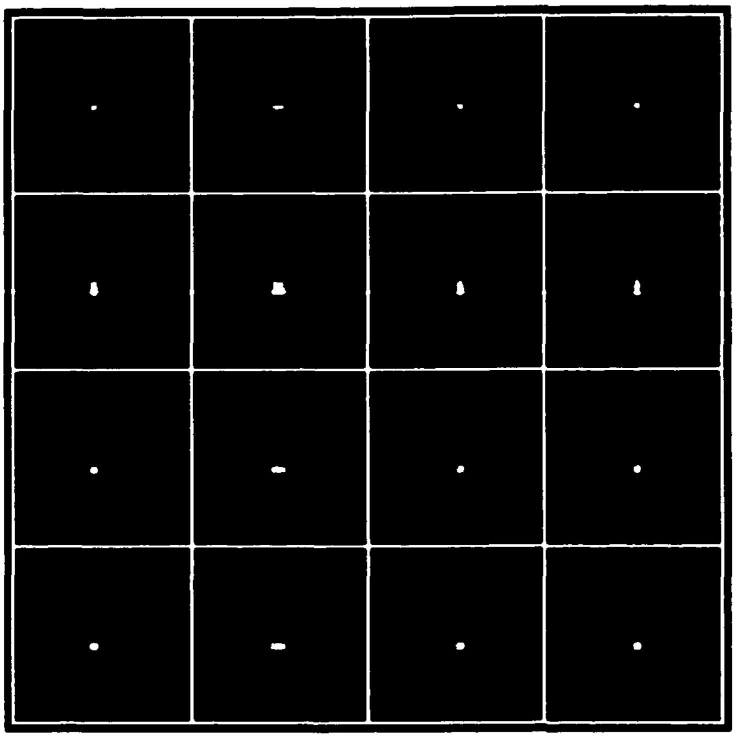


Figure 5.21: Reconstructed $\xi/2 \times \xi/2$ targets.

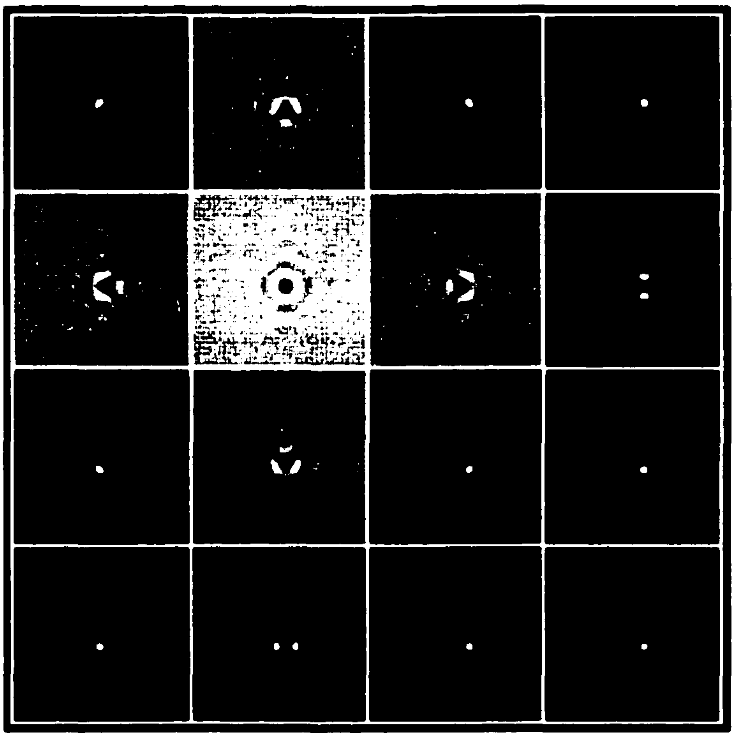


Figure 5.22: Aliased component of reconstructed $\xi/2 \times \xi/2$ targets.

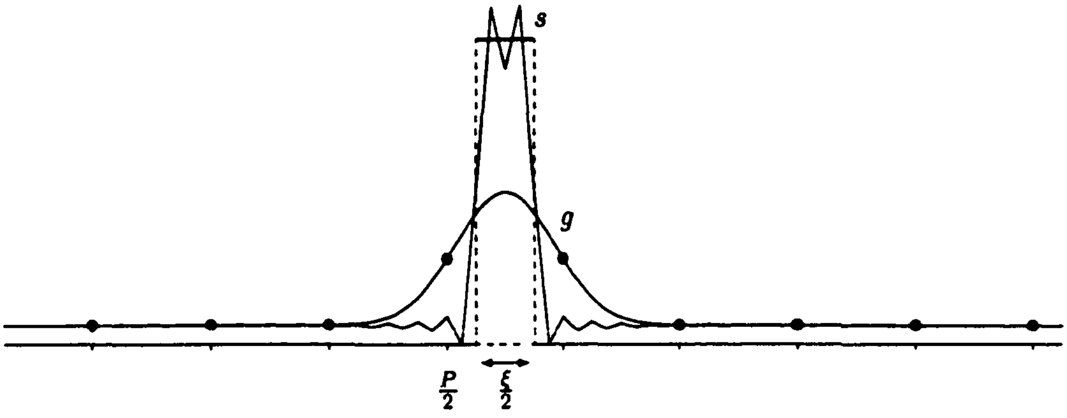


Figure 5.23: Pulse train s and g functions — sample-scene phase shift centered pulse between sampling points.

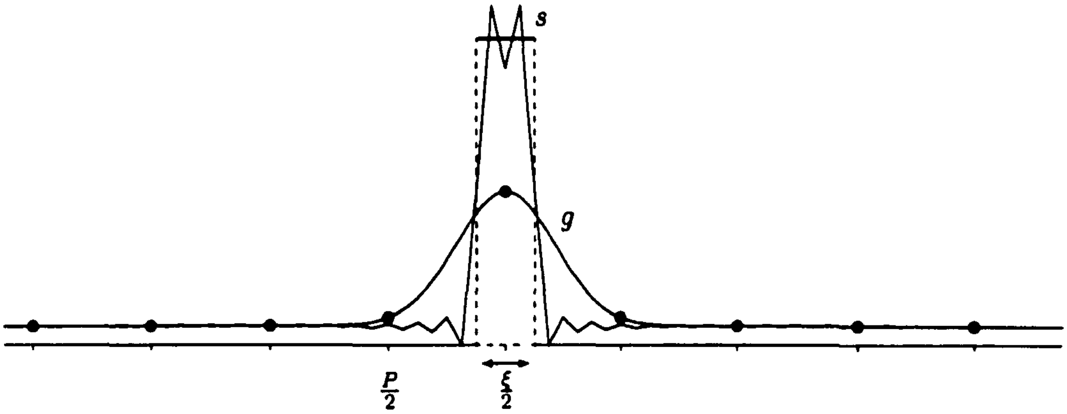


Figure 5.24: Pulse train s and g functions — sample-scene phase shift centered pulse on sampling point.

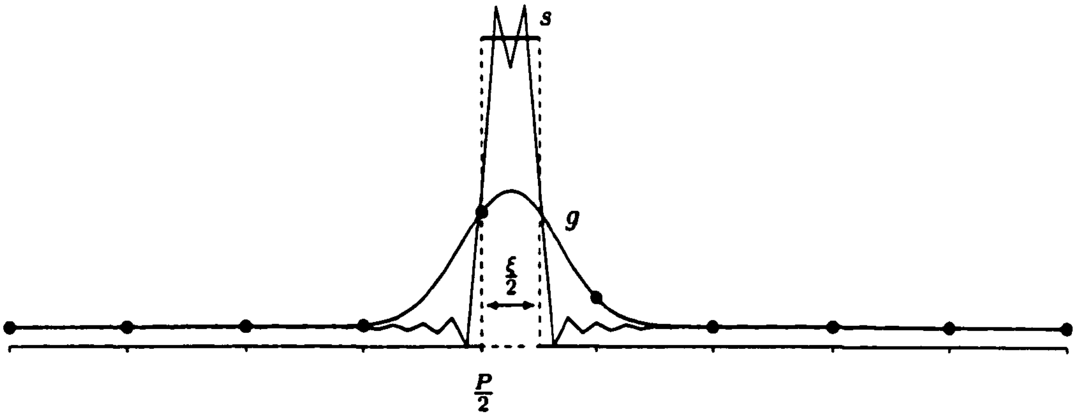


Figure 5.25: Pulse train s and g functions — sample-scene phase shift positioned pulse asymmetrically relative to sampling points.

$\xi \times \xi$ Figure	$\xi/2 \times \xi/2$ Figure	Observations
5.4	5.19	All 16 images are identical in both cases and the ratio of the area covered by the representation of the $\xi \times \xi$ target to the area covered by the representation of the $\xi/2 \times \xi/2$ target is approximately 4:1, as might be expected.
5.5	5.20	All 16 images are identical in both cases but the ratio of the area covered by the representation of the $\xi \times \xi$ target to the area covered by the representation of the $\xi/2 \times \xi/2$ target is approximately 1:1.
5.6	5.21	Consider a periodic replication of both figures. A 4 × 4 frame from Figure 5.6 with input scene 1 at the top left hand corner is almost indistinguishable from a 4 × 4 frame from Figure 5.21 with input scene 6 at the top left hand corner.
5.7	5.22	As for the previous case.

Table 5.5: Comparison of scenes with $\xi \times \xi$ and $\xi/2 \times \xi/2$ targets.

The combined effects of sample-scene phase shift and aliasing on the reconstructed image might be expected therefore to be significant, when the dimension of the target is less than or comparable to a unit area of the sampling grid.

As specified in Table 5.1, only four horizontal and the same four vertical sample-scene phase shifts were implemented to generate the 16 2-D input scenes with a $\xi/2 \times \xi/2$ target. Equivalently, each pulse train function was subjected to one of the four possible sample-

Input scene number	pre-sampling 1-D functions	post-sampling 1-D functions
2,6,10,14	Figure 5.23	Figure 5.26
4,8,12,16	Figure 5.24	Figure 5.27
1,5,9,13	Figure 5.25	Figure 5.28
3,7,11,15	v.m.i. of Figure 5.25	v.m.i. of Figure 5.28

Table 5.6: Relationship between numbered input scenes and 1-D figures (v.m.i. denotes "vertical mirror image").

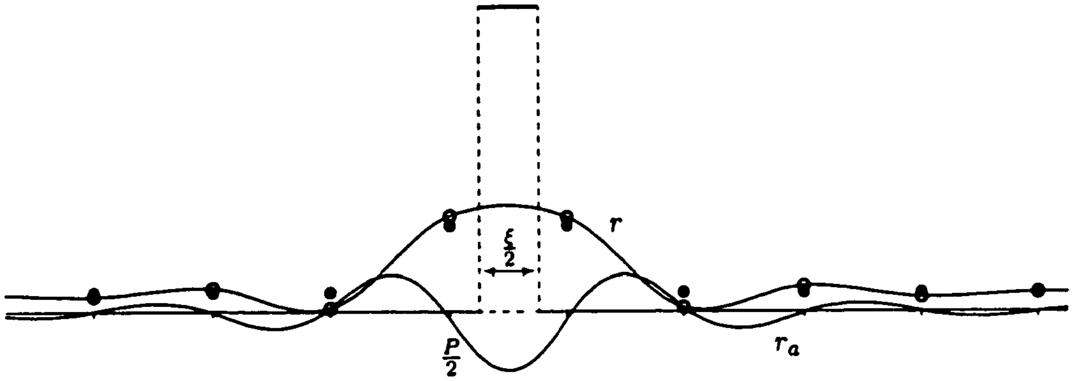


Figure 5.26: Pulse train r and r_a functions — sample-scene phase shift centered pulse between sampling points.

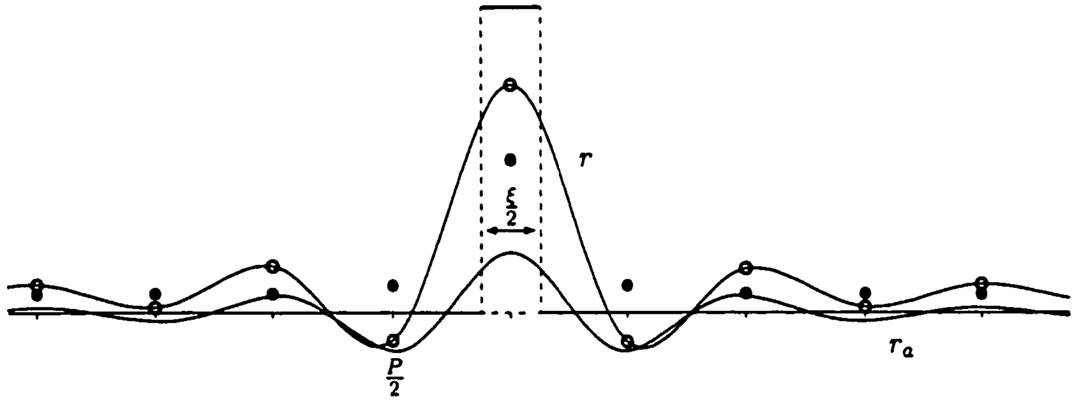


Figure 5.27: Pulse train r and r_a functions — sample-scene phase shift centered pulse on sampling point.

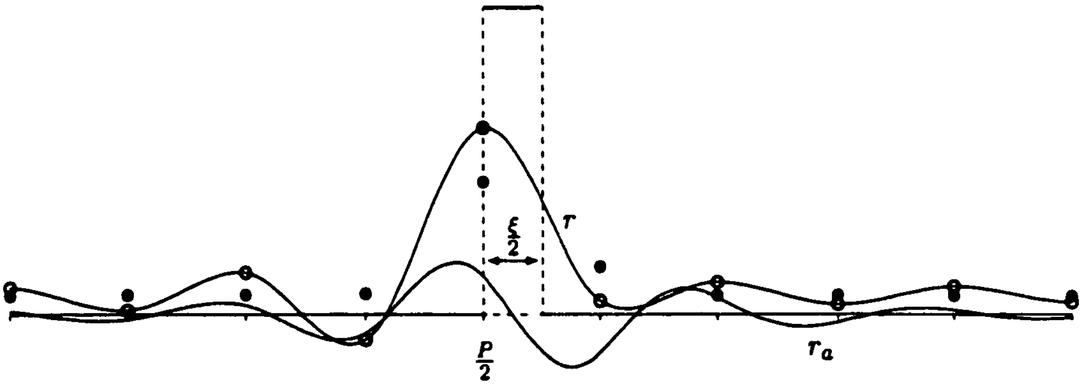


Figure 5.28: Pulse train r and r_a functions — sample-scene phase shift positioned pulse asymmetrically relative to sampling points.

scene phase shifts specified in Section 5.2.1. Because each of the 2-D scenes is the product of two pulse train functions, one horizontal and one vertical, the information contained in Figures 5.26 to 5.28 can again be used to predict the characteristics of the 2-D targets in the reconstructed images.

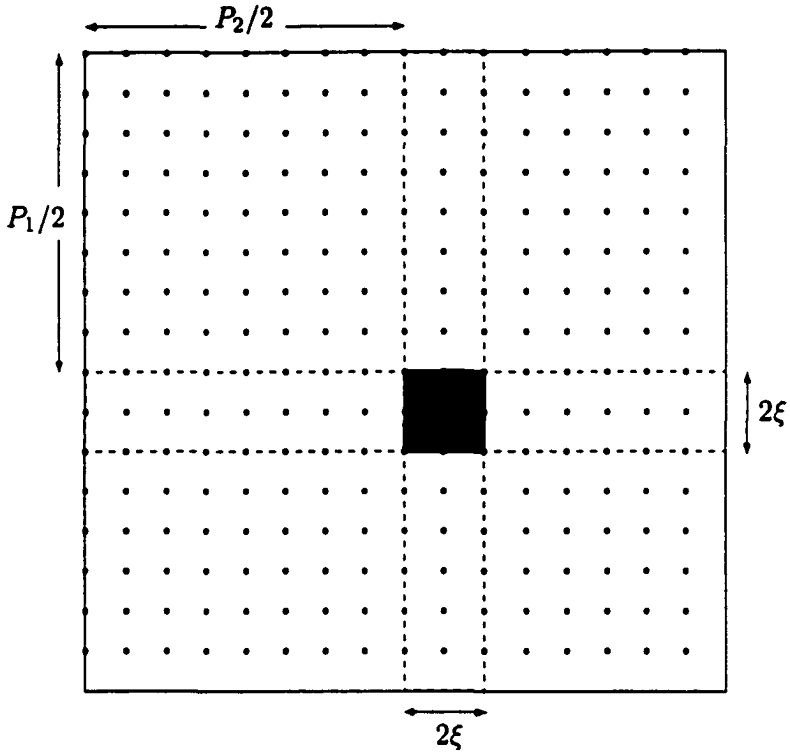


Figure 5.29: Stylized representation of $2\xi \times 2\xi$ input scene 1 (color inverted).

5.2.3 $2\xi \times 2\xi$ targets

Sixteen different input scenes, each comprised of a $2\xi \times 2\xi$ white target on a black background, were synthesized in the frequency domain. The top left-hand corner of the white target was positioned in the center of the black background in the first input scene, as shown in Figure 5.29. Subsequent scenes were created as in Section 5.2.1, and all 16 input scenes are again defined by Table 5.1 which identifies the sample-scene phase shift of each $2\xi \times 2\xi$

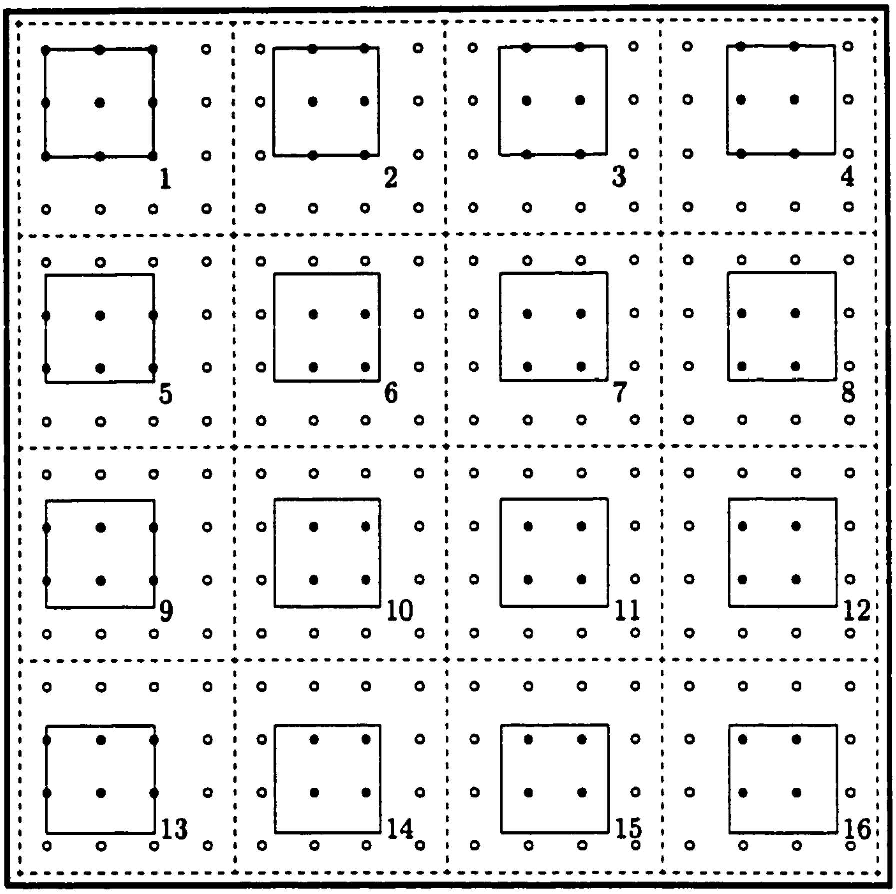


Figure 5.30: Placement of 2ξ × 2ξ targets on sampling grid.

target relative to the target's position in Figure 5.29. Figure 5.30 graphically depicts the resultant position of the target in each input scene relative to the sampling grid.

The 16 different images are combined in Figures 5.31 to 5.34, and the sample-scene phase shift relative to Figure 5.29 (input scene 1) corresponds to Table 5.1 and Figure 5.30. Figures 5.31 to 5.34 display the 16 images at the same three stages of the end-to-end system as are used in Section 5.2.1.

Prior to sampling (Figures 5.31 and 5.32), the image of the target is identical in all 16 cases independent of sample-scene phase.

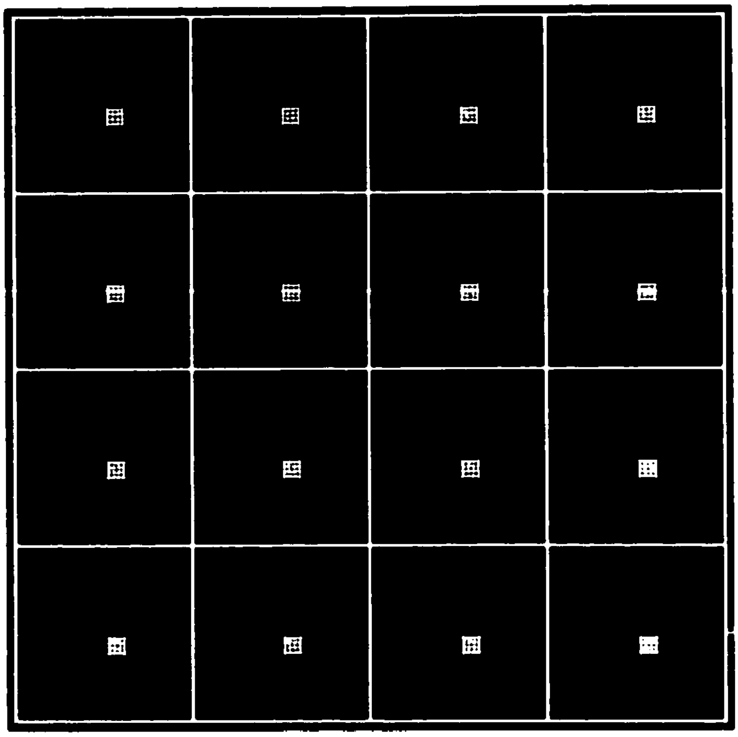


Figure 5.31: Input scenes with $2\xi \times 2\xi$ targets.

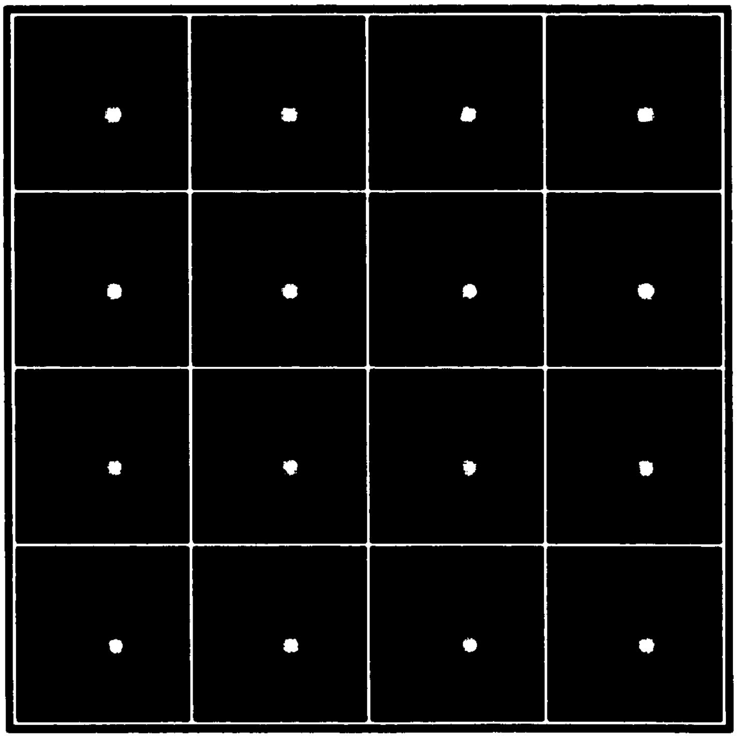


Figure 5.32: Blurred input scenes with $2\xi \times 2\xi$ targets.

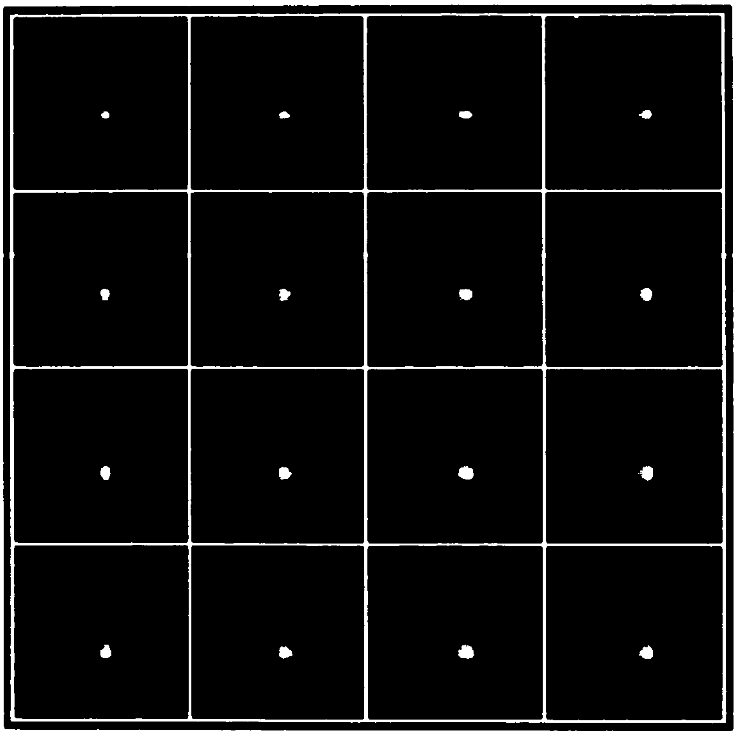


Figure 5.33: Reconstructed $2\xi \times 2\xi$ targets.

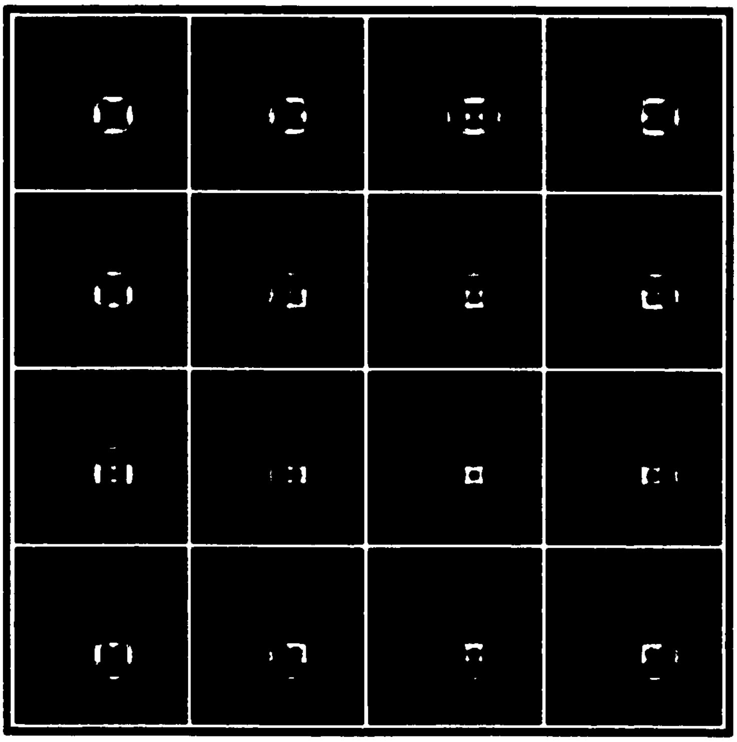


Figure 5.34: Aliased component of reconstructed $2\xi \times 2\xi$ targets.

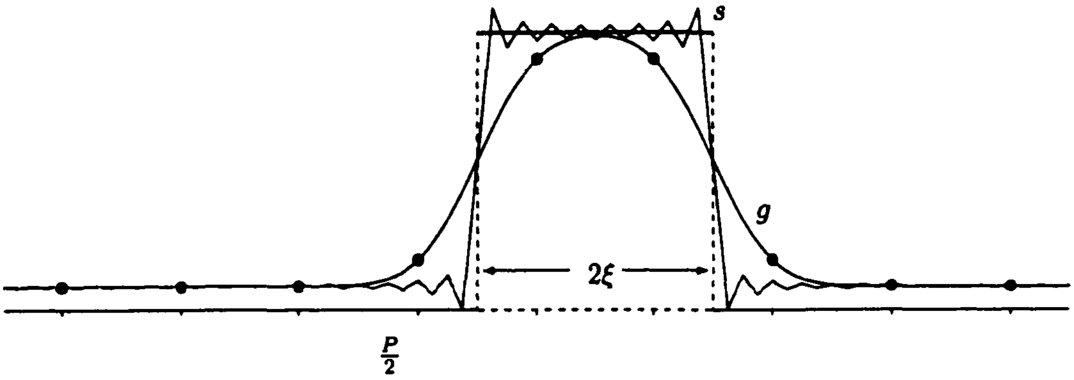


Figure 5.35: Pulse train *s* and *g* functions — sample-scene phase shift centered pulse between 2 sampling points.

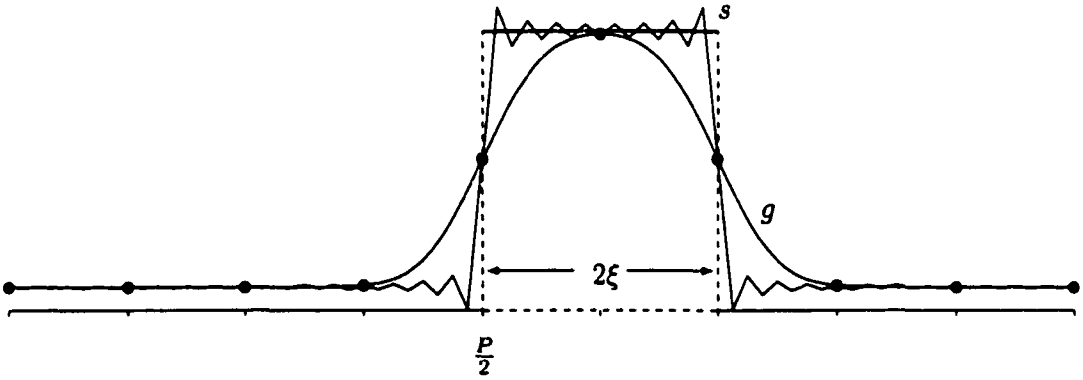


Figure 5.36: Pulse train *s* and *g* functions — sample-scene phase shift centered pulse on sampling point.

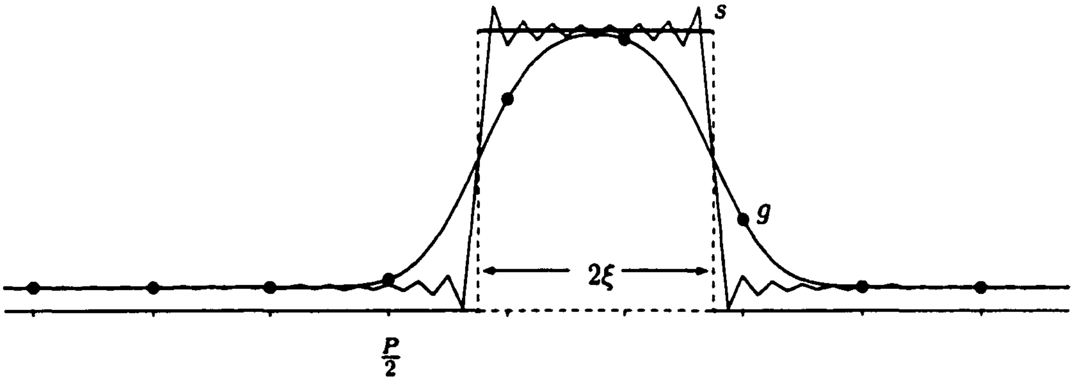


Figure 5.37: Pulse train *s* and *g* functions — sample-scene phase shift positioned pulse asymmetrically relative to sampling points.

After sampling and reconstruction (Figures 5.33 and 5.34), the representation of the target varies much less significantly with sample-scene phase than in the cases where the dimension of the target is comparable to or less than a unit area of the sampling grid. Table 5.7 summarizes the observations which can be made by comparing and contrasting corresponding $\xi \times \xi$ and $2\xi \times 2\xi$ target images.

$\xi \times \xi$ Figure	$2\xi \times 2\xi$ Figure	Observations
5.4	5.31	All 16 images are identical in both cases and the ratio of the area covered by the representation of the $\xi \times \xi$ target to the area covered by the representation of the $2\xi \times 2\xi$ target is approximately 1:4, as might be expected.
5.5	5.32	All 16 images are identical in both cases and the ratio of the area covered by the representation of the $\xi \times \xi$ target to the area covered by the representation of the $2\xi \times 2\xi$ target is approximately 1:4.
5.6	5.33	All 16 images in Figure 5.33 are similar to each other and similar in shape and size to the original input scenes.
5.7	5.34	While still displaying the same characteristics relative to sample-scene phase shift as were exhibited in Figure 5.7, all 16 images in Figure 5.34 are similar to each other in shape and size.

Table 5.7: Comparison of scenes with $\xi \times \xi$ and $2\xi \times 2\xi$ targets.

Figures 5.35 to 5.37 show a 1-D horizontal cross-section through the $2\xi \times 2\xi$ target of a 2-D input scene having the same components as in Section 5.2.1. Again, no two of the p' sequences are the same due to the sample-scene phase shift in the location of the 2ξ-width pulse.

Figures 5.38 to 5.40 show that the aliased component is much less significantly affected by phase shift than in the previous cases ($\xi \times \xi$ and $\xi/2 \times \xi/2$). Table 5.8 identifies the

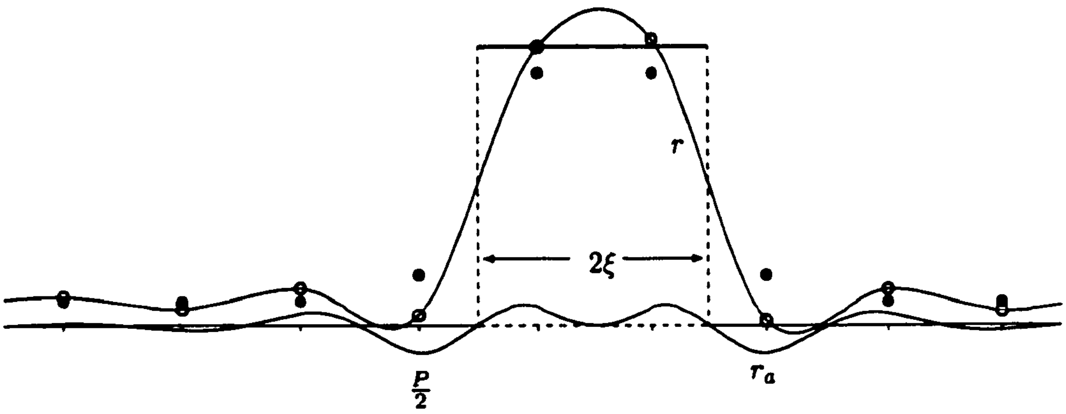


Figure 5.38: Pulse train r and r_a functions — sample-scene phase shift centered pulse between 2 sampling points.

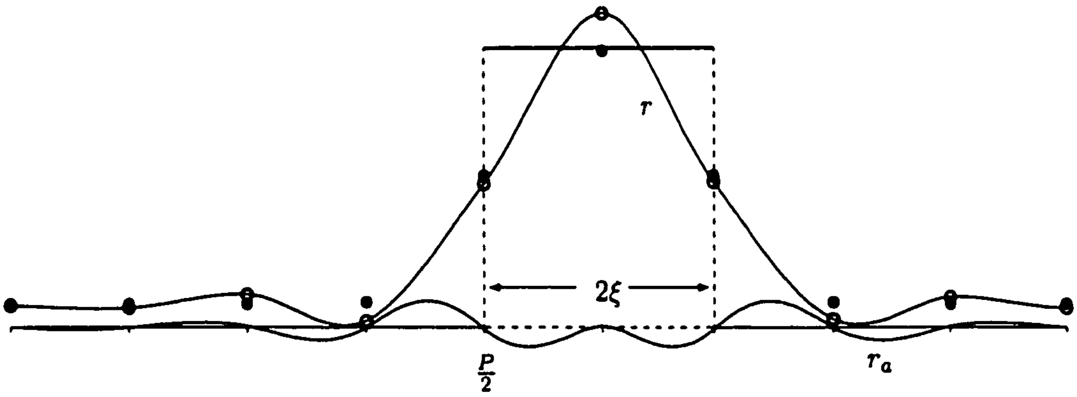


Figure 5.39: Pulse train r and r_a functions — sample-scene phase shift centered pulse on sampling point.

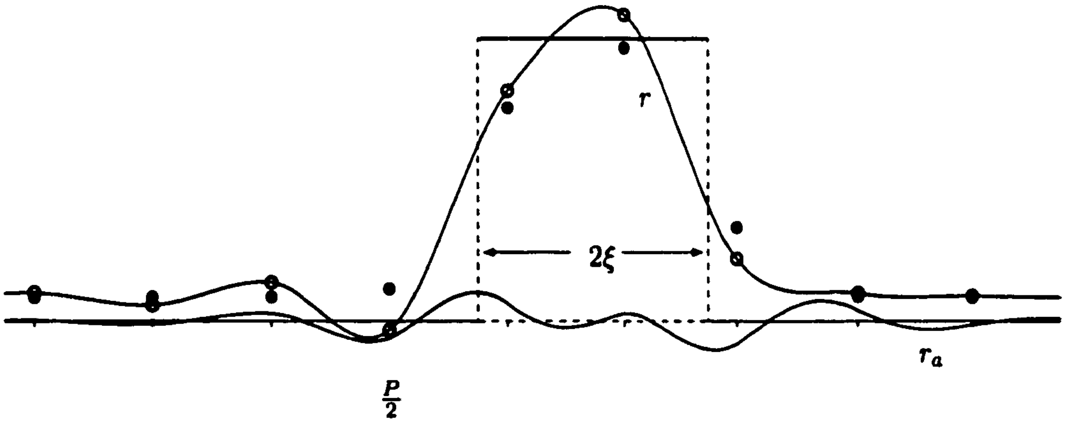


Figure 5.40: Pulse train r and r_a functions — sample-scene phase shift positioned pulse asymmetrically relative to sampling points.

provenance of the six 1-D figures. In addition, corresponding 1-D figures derived from input scene 2 *et al.* produce the vertical mirror images of Figures 5.37 and 5.40. From Figures 5.38 to 5.40, it will again be noted that no two of the q sequences are the same due to the sample-scene phase shift in the location of the 2ξ-width pulse.

Input scene number	pre-sampling 1-D functions	post-sampling 1-D functions
3,7,11,15	Figure 5.35	Figure 5.38
1,5,9,13	Figure 5.36	Figure 5.39
4,8,12,16	Figure 5.37	Figure 5.40
2,6,10,14	v.m.i. of Figure 5.37	v.m.i. of Figure 5.40

Table 5.8: Relationship between numbered input scenes and 1-D figures (v.m.i. denotes "vertical mirror image").

As specified in Table 5.1, only four horizontal and the same four vertical sample-scene phase shifts were implemented to generate the 16 2-D input scenes with a $2\xi \times 2\xi$ target. Equivalently, each pulse train function was subjected to one of the four possible sample-scene phase shifts specified in Section 5.2.1. Because each of the 2-D scenes is the product of two pulse train functions, one horizontal and one vertical, the information contained in Figures 5.38 to 5.40 can again be used to predict the characteristics of the 2-D targets in the reconstructed images.

5.2.4 $3\xi \times 3\xi$ targets

Sixteen different input scenes, each comprised of a $3\xi \times 3\xi$ white target on a black background, were synthesized in the frequency domain. The top left-hand corner of the white target was positioned in the center of the black background in the first input scene, as shown in Figure 5.41. Subsequent scenes were created as in Section 5.2.1, and all 16 input scenes

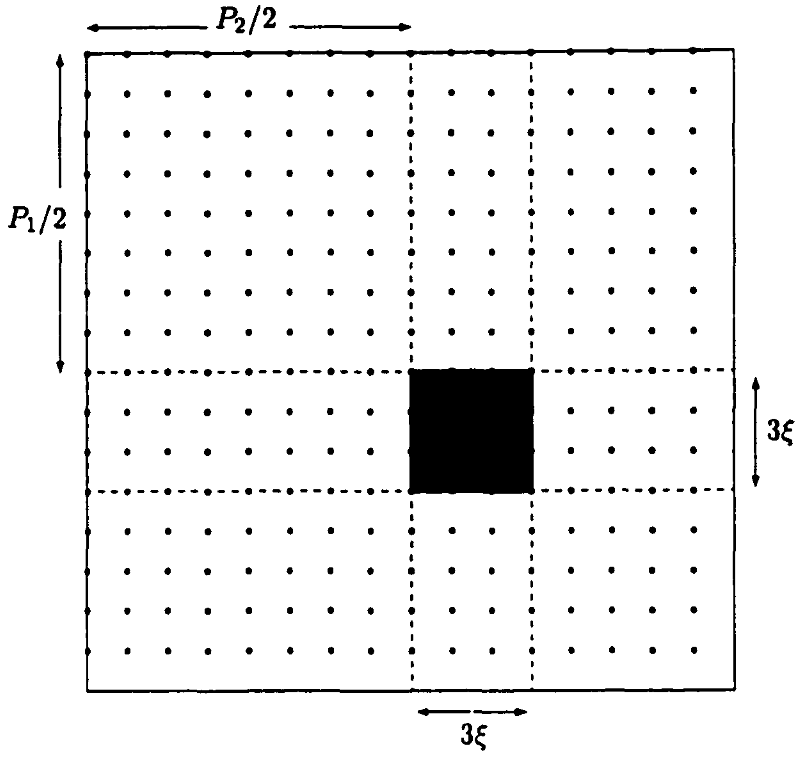


Figure 5.41: Stylized representation of $3\xi \times 3\xi$ input scene 1 (color inverted).

are again defined by Table 5.1 which identifies the sample-scene phase shift of each $3\xi \times 3\xi$ target relative to the target's position in Figure 5.41. Figure 5.42 graphically depicts the resultant position of the target in each input scene relative to the sampling grid.

The 16 different images are combined in Figures 5.43 to 5.46, and the sample-scene phase shift relative to Figure 5.41 (input scene 1) corresponds to Table 5.1 and Figure 5.42. Figures 5.43 to 5.46 display the 16 images at the same three stages of the end-to-end system as are used in Section 5.2.1.

Once again, prior to sampling (Figures 5.43 and 5.44), the image of the target is identical in all 16 cases independent of sample-scene phase. After sampling and reconstruction (Figures 5.45 and 5.46), the representation of the target varies much less significantly with sample-scene phase than in the cases where the dimension of the target is comparable to

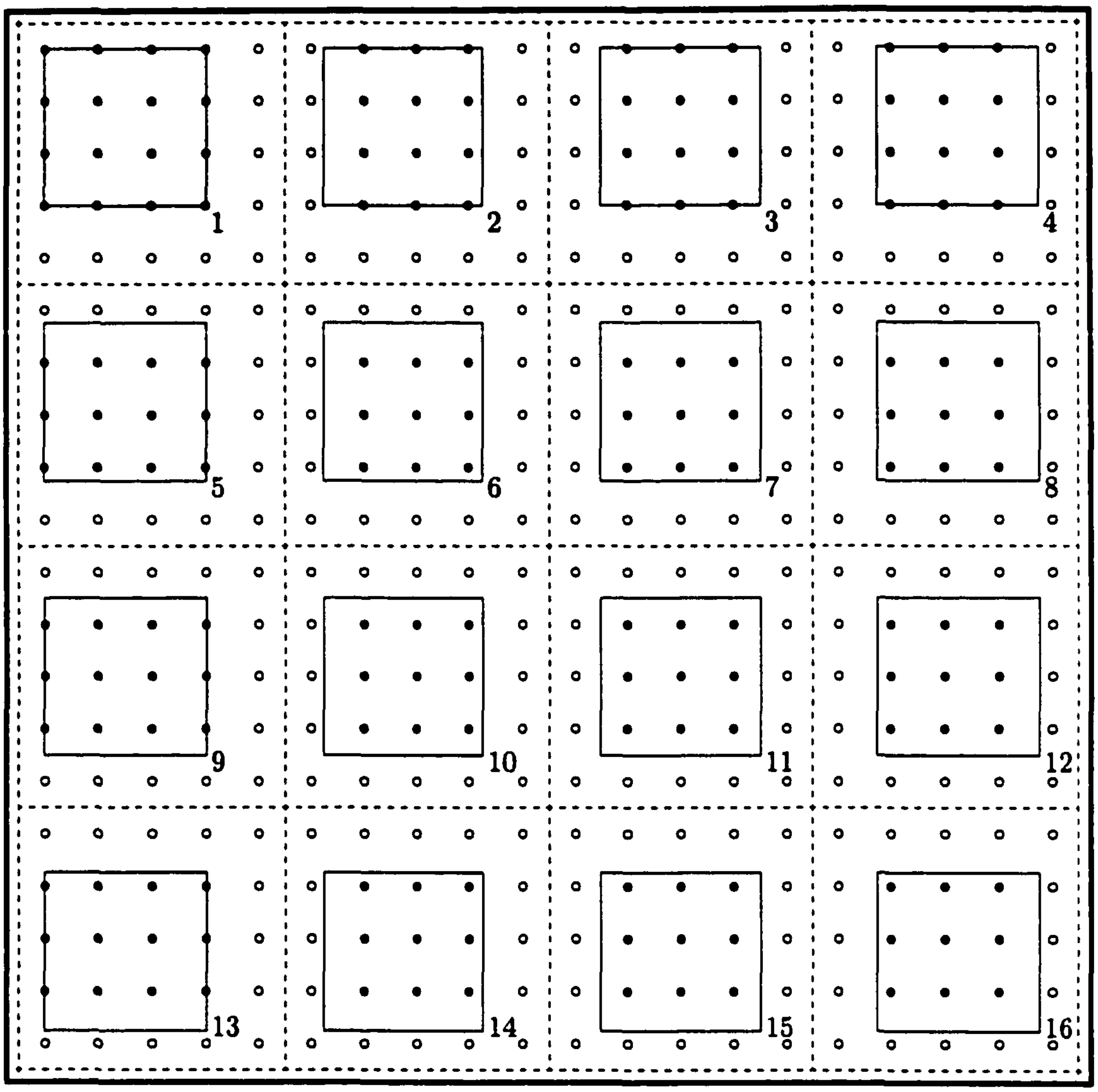


Figure 5.42: Location of $3\xi \times 3\xi$ targets on sampling grid.

or less than a unit area of the sampling grid. Table 5.9 summarizes the observations which can be made by comparing and contrasting corresponding $\xi \times \xi$ and $3\xi \times 3\xi$ target images.

Figures 5.47 to 5.49 show a 1-D horizontal cross-section through the $3\xi \times 3\xi$ target of a 2-D input scene having the same components as in Section 5.2.1.

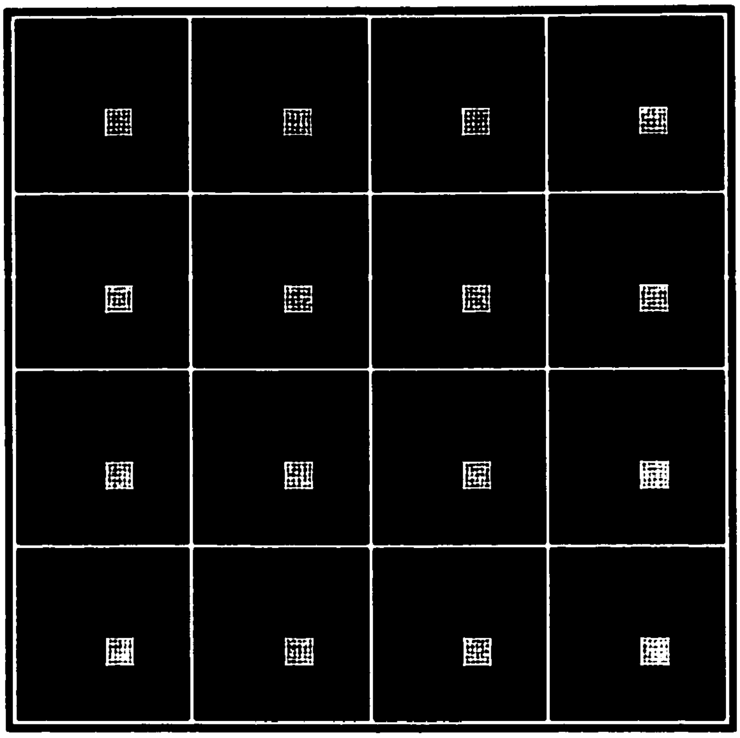


Figure 5.43: Input scenes with $3\xi \times 3\xi$ targets.

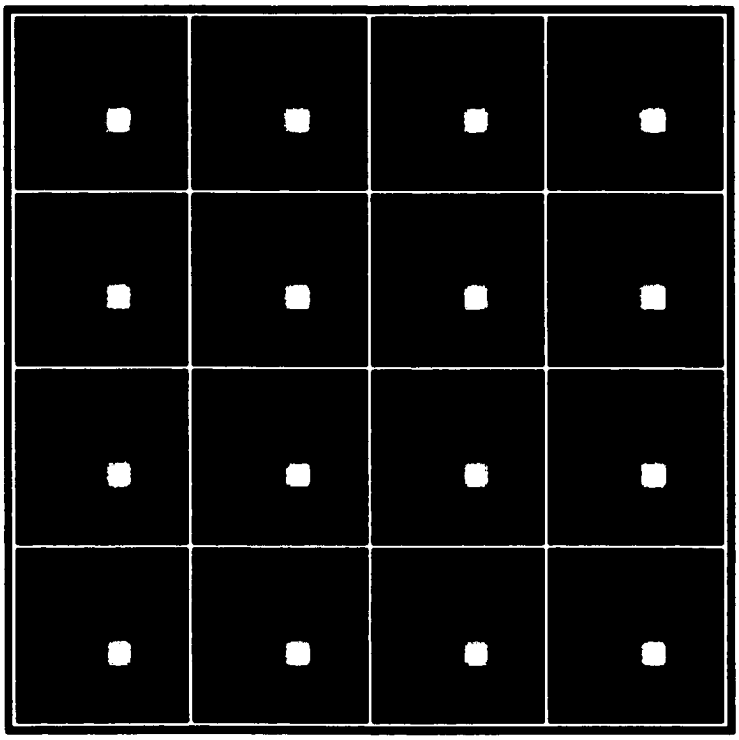


Figure 5.44: Blurred input scenes with $3\xi \times 3\xi$ targets.

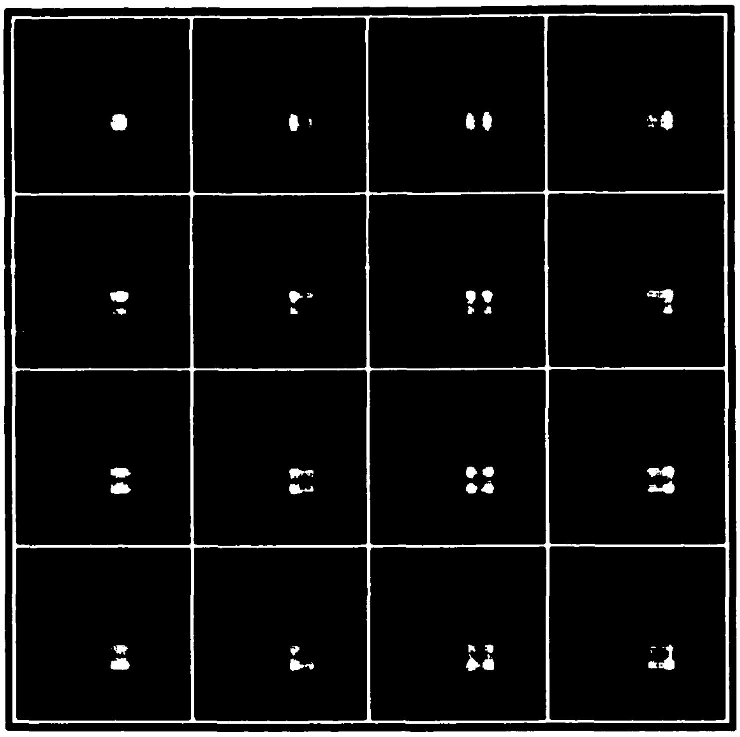


Figure 5.45: Reconstructed 3ξ × 3ξ targets.

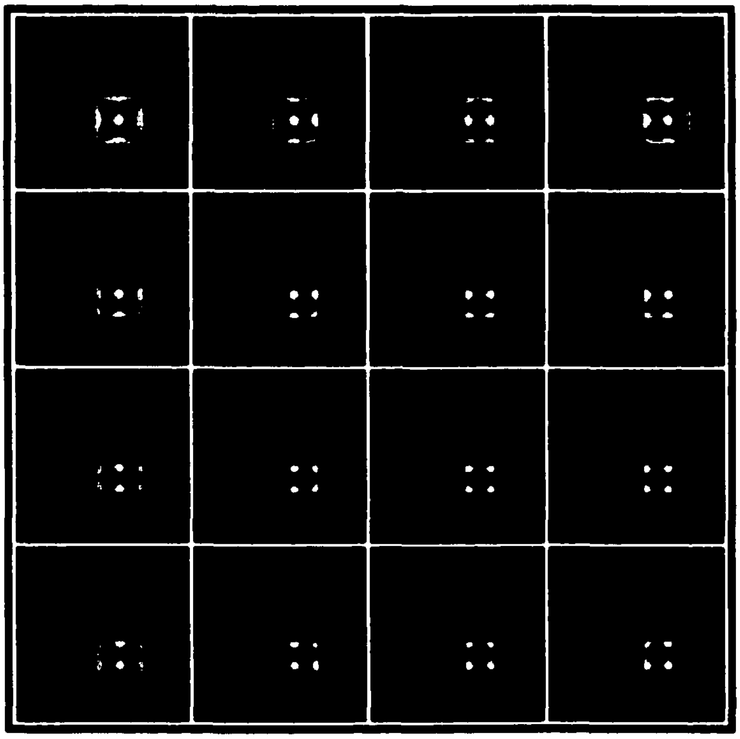


Figure 5.46: Aliased component of reconstructed 3ξ × 3ξ targets.

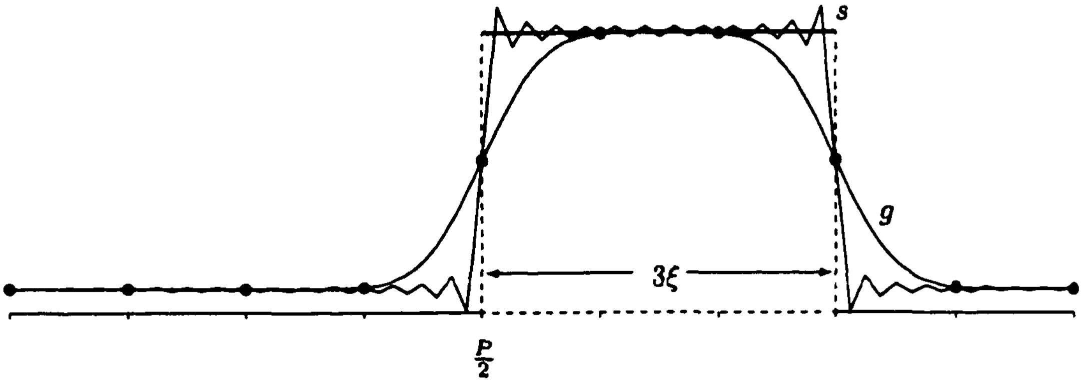


Figure 5.47: Pulse train *s* and *g* functions — sample-scene phase shift centered pulse between 2 sampling points.

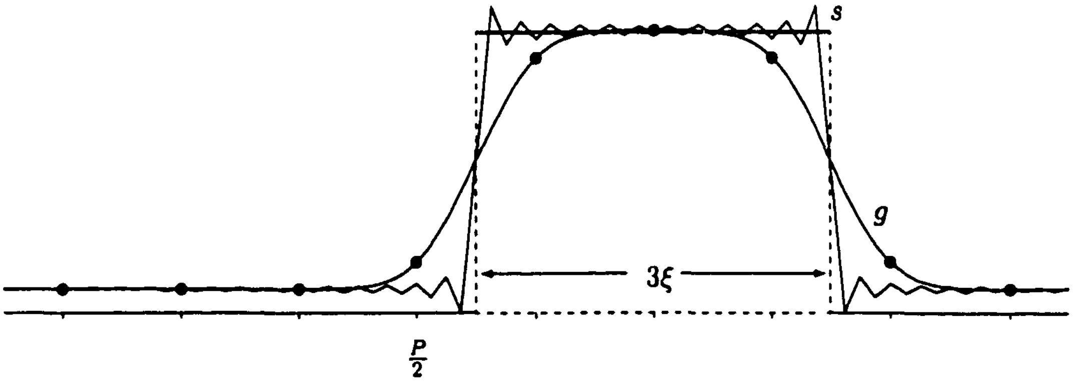


Figure 5.48: Pulse train *s* and *g* functions — sample-scene phase shift centered pulse on sampling point.

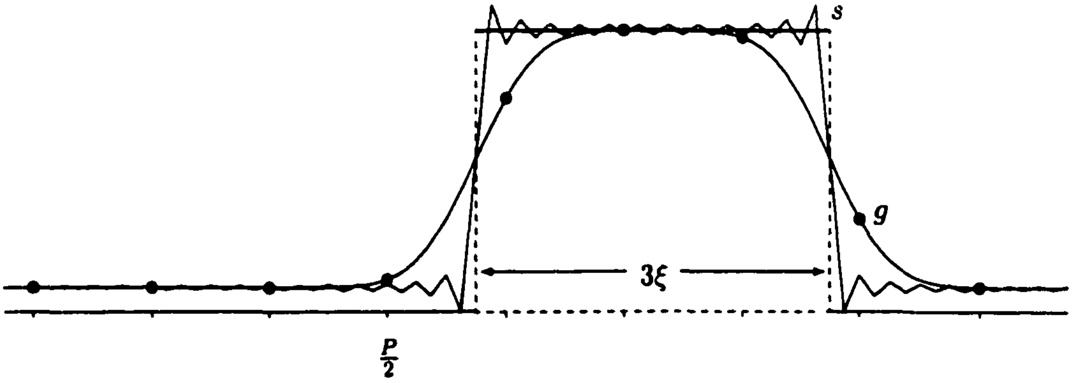


Figure 5.49: Pulse train *s* and *g* functions — sample-scene phase shift positioned pulse asymmetrically relative to sampling points.

$\xi \times \xi$ Figure	$3\xi \times 3\xi$ Figure	Observations
5.4	5.43	All 16 images are identical in both cases and the ratio of the area covered by the representation of the $\xi \times \xi$ target to the area covered by the representation of the $3\xi \times 3\xi$ target is approximately 1:9, as might be expected.
5.5	5.44	All 16 images are identical in both cases and the ratio of the area covered by the representation of the $\xi \times \xi$ target to the area covered by the representation of the $3\xi \times 3\xi$ target is approximately 1:9.
5.6	5.45	While displaying some of the same characteristics relative to sample-scene phase shift as were exhibited in Figure 5.6, all 16 images in Figure 5.45 are similar to each other and similar in shape and size to the original input scene.
5.7	5.46	While still displaying the same characteristics relative to sample-scene phase shift as were exhibited in Figure 5.7, all 16 images in Figure 5.46 are similar to each other in shape and size.

Table 5.9: Comparison of scenes with $\xi \times \xi$ and $3\xi \times 3\xi$ targets.

Again, no two of the p' sequences are the same due to the sample-scene phase shift in the location of the 3ξ-width pulse.

Figures 5.50 to 5.52 show that the aliased component, r_a, is again significantly affected by sample-scene phase shift and is responsible for the artifacts visible in Figure 5.45. Table 5.10 identifies the provenance of the six 1-D figures. In addition, corresponding 1-D figures derived from input scene 2 *et al.* produce the vertical mirror images of Figures 5.49 and 5.52. From Figures 5.50 to 5.52, it will again be noted that no two of the q sequences are the same due to the sample-scene phase shift in the location of the 3ξ-width pulse.

As specified in Table 5.1, only four horizontal and the same four vertical sample-scene phase shifts were implemented to generate the 16 2-D input scenes with a $3\xi \times 3\xi$ target.

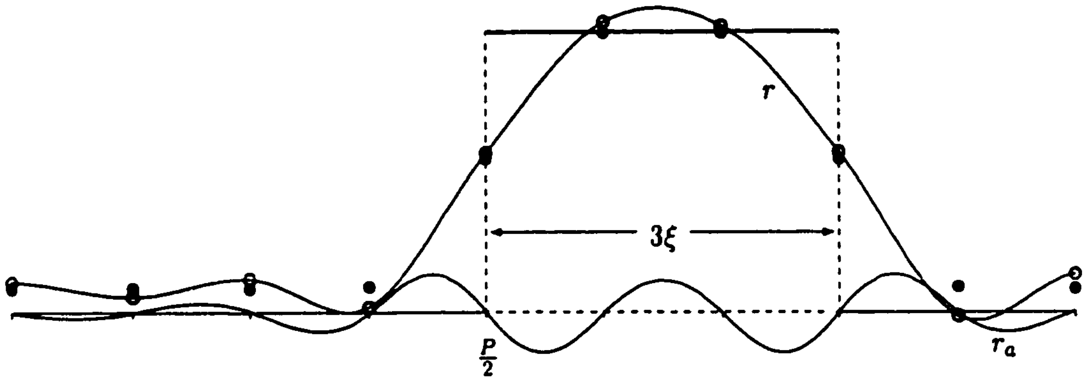


Figure 5.50: Pulse train r and r_a functions — sample-scene phase shift centered pulse between 2 sampling points.

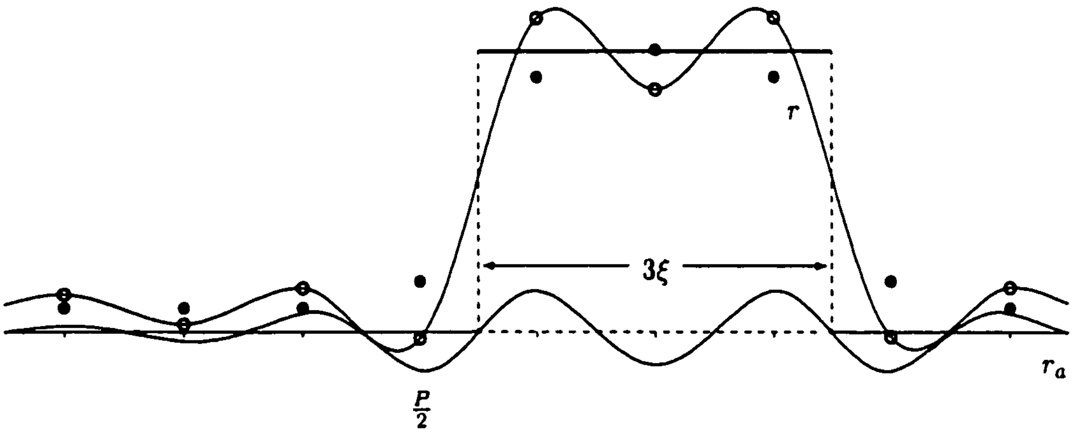


Figure 5.51: Pulse train r and r_a functions — sample-scene phase shift centered pulse on sampling point.

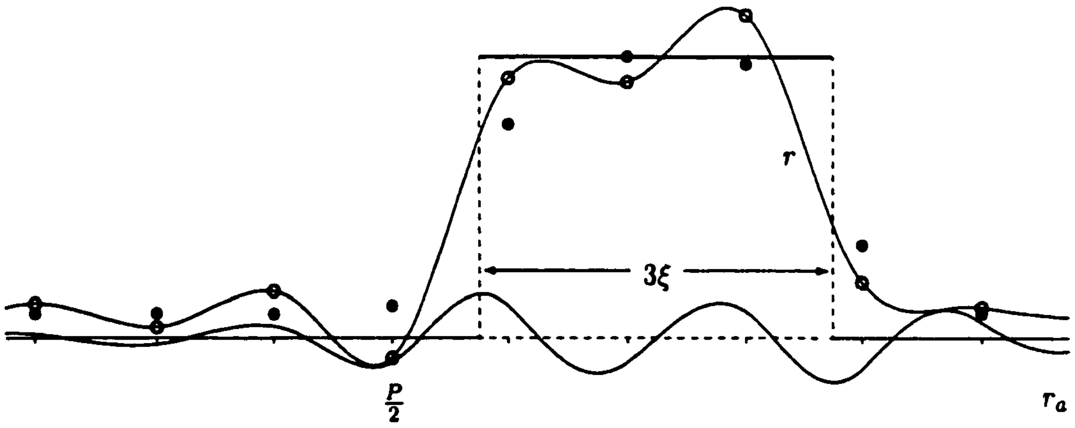


Figure 5.52: Pulse train r and r_a functions — sample-scene phase shift positioned pulse asymmetrically relative to sampling points.

Input scene number	pre-sampling 1-D functions	post-sampling 1-D functions
1,5,9,13	Figure 5.47	Figure 5.50
3,7,11,15	Figure 5.48	Figure 5.51
4,8,12,16	Figure 5.49	Figure 5.52
2,6,10,14	v.m.i. of Figure 5.49	v.m.i. of Figure 5.52

Table 5.10: Relationship between numbered input scenes and 1-D figures (v.m.i. denotes "vertical mirror image").

Equivalently, each pulse train function was subjected to one of the four possible sample-scene phase shifts specified in Section 5.2.1. Because each of the 2-D scenes is the product of two pulse train functions, one horizontal and one vertical, the information contained in Figures 5.50 to 5.52 can again be used to predict the characteristics of the 2-D targets in the reconstructed images.

Chapter 6

Summary and Conclusions

6.1 Summary

The 2-D continuous/discrete/continuous (c/d/c) imaging system model described in detail in Chapter 2 was used as the foundation on which a 2-D end-to-end digital imaging system design environment was built. The c/d/c model is more comprehensive than other common, but incomplete, models (as is discussed in the introduction to Chapter 2). The research of existing environments with similar functionality described in Section 1.3 confirmed that none of the environments is based on the 2-D c/d/c model.

In terms of lines of code, the environment is comprised of approximately 10,000 instructions, estimated by a semi-colon count in the .c modules. The capabilities required of the interactive simulation environment and the objectives it was hoped to achieve are specified in the preface to Chapter 1. All of the minimum requirements were met, as detailed in Sections 6.1.1 to 6.1.9, and the extent to which the objectives were accomplished is discussed in Sections 6.1.10 to 6.1.15.

6.1.1 Input

Some forty images (input scenes) currently reside in the environment's library, and a user-friendly pre-processing option described in Section 2.2.3.1 allows any image which can be read by *xv* [3] to be easily reformatted and added to the library, where it becomes immediately available for future use. An additional feature allows the user to synthesize an input scene directly in the frequency domain as discussed in Section 2.2.3.2.

6.1.2 Image display

A matrix representation (image) of the scene at the input and output of all c/d/c system components can be readily displayed in either the spatial domain or the frequency domain. The scene to be displayed is selected by clicking at the appropriate location on the schematic diagram of the end-to-end c/d/c system model. Toggling between the two domains is achieved simply by clicking in the image window. The input scene can be displayed simultaneously in the currently-selected domain for comparison with the processed scene from a later stage in the end-to-end system.

6.1.3 Interactive modification of system filters

A menu of possible filter types is provided for the input scene filter, the acquisition filter, the digital filter, and the reconstruction filter. (See Sections 2.2.3.4, 2.2.4.1, 2.2.7.1, and 2.2.8.1 respectively for details.) When a particular filter type is selected by the user, the facility to alter the horizontal and/or vertical parameter(s) pertaining to that specific filter type becomes available to the user. The frequency responses of the horizontal or vertical components of the currently implemented filters can be viewed by clicking on any filter box in the c/d/c schematic window, at which time each filter box is replaced by a response window.

6.1.4 Interactive modification of 2-D sampling density

The 2-D sampling density can be modified at any time from the sampling menu by altering the number of horizontal and/or vertical samples per period. The inter-sample distance of the sampling grid can also be adjusted from the sampling menu.

6.1.5 Image acquisition noise simulation

Only additive noise is simulated, as defined in Section 2.2.6.

6.1.6 Visualization of the aliasing effects due to spatial sampling

In both 1-D and 2-D modes, the environment can operate in one of four *component* modes. As described in Section 2.2.11, the reconstructed output is the sum of three separate components. That is, in the frequency domain, the output from the sampling box can be separated into the cascaded component and the error attributable to frequency folding, with the result that the input to the digital filter, and consequently the reconstructed output, can be separated into three components — the cascaded component, the aliased component, and the noise component. The environment has the capability to visualize composite scenes or any one of these three component scenes, at any appropriate location in the end-to-end system. No matter which component mode is selected, composite scenes are displayed if the display point selected precedes the sampling box in the end-to-end system.

6.1.7 1-D operation

Once a 2-D input scene has been loaded, the environment can be toggled into 1-D mode. Most changes implemented while in the 1-D mode do not affect the displayed input scene until the 1-D mode has been exited. However, altering the number of samples per period immediately causes the image window to be resized and the input scene to be reprocessed from raw data. The following functionality is available in 1-D mode.

6.1.7.1 selecting input

Any row or column from the 2-D input scene can be selected and used as the 1-D input to the c/d/c system by clicking in the image window to specify the row and column. Alternatively, a 1-D input can be selected from the environment's library.

6.1.7.2 vertical and horizontal modes

In 1-D operation, either vertical or horizontal mode must necessarily be selected. As in 2-D operation, changes made to filter parameters in vertical mode do not affect the horizontal filter components, and vice versa. However, any changes made to either vertical or horizontal filter parameters during 1-D operation remain in effect on reverting to 2-D operation.

6.1.7.3 1-D input display

The amplitude and phase of the vector representation of the 1-D input can be viewed at any stage of the end-to-end system, and visualization of the vector representation can be toggled between the frequency domain and the spatial domain. The vector representation of the processed input from any stage of the c/d/c system can be overlaid on the vector representation of the input for visual comparison, or the overlay facility can be switched off. A frequency domain range of 2× or 4× the Nyquist frequency can be selected.

6.1.7.4 spatial domain zoom facility

The spatial domain display provides a zoom-in, zoom-out feature which allows successive 2× magnification of a region of interest (ROI) to a maximum magnification factor of 32×. An additional feature provides continuous scrolling through the periodically replicated vector representation at any magnification factor greater than 1.

6.1.8 System snapshots

Save and retrieve capabilities are provided for system snapshots. The current definitions of all system components and parameter values are captured with a save command and restored with a retrieve command. This facility is available in both 1-D and 2-D.

6.1.9 Performance analysis

In either 1-D or 2-D mode, fidelity metrics for quantitative end-to-end system performance evaluation can be applied to composite or cascaded scenes that have been processed through the c/d/c system, as described in Section 2.2.10. A comparable fidelity metric is provided to measure the aliased and error components at the output from the end-to-end system in both 1-D and 2-D modes. The latter 2-D metric is defined in Equation 4.2.

6.1.10 Support for system design and performance analysis

This goal was achieved, as specified in Sections 6.1.1 to 6.1.9. The verification and validation of the implementation of the c/d/c model are described in detail in Chapter 5.

6.1.11 Acceptable response time

As discussed in Section 4.1.3, the calculations of 2-D Fourier transforms and 2-D spatial domain convolutions are CPU-time intensive. The use of these two computational techniques was considered carefully, therefore, and resulted in the convention that Fourier representations are the standard format of all data supplied to the environment, and subsequent processing occurring exclusively in the frequency domain. Also investigated was the operation of the environment in two distinct modes, one a 2-D mode in which the matrix representation of the input scene was processed through the c/d/c system model, the other a 1-D mode in which a vector representation was processed through the end-to-end system model, and in which the user had the capability to redesign the system by modifying the c/d/c system model parameters. In practice, this two-mode implementation proved frustrating to the user and was rejected, as described in Section 4.1.4. A redesign adapted the philosophy of this approach, adding the design capabilities of the 1-D mode to the 2-D mode. By simply selecting the input scene for display when tentative filter parameter changes are being made in 2-D mode, the user can easily avoid the lengthier system response time associated with 2-D display after reprocessing.

6.1.12 Sampling grids at least as large as 512 × 512

In addition to response time, the amount of memory required by the environment was carefully considered, as discussed in Section 4.2. To minimize memory requirements, input scenes are cropped, if necessary, while being pre-processed for inclusion in the environment's library. Possible values of user-selected parameters such as $\tau_1 \times \tau_2$, $N_1 \times N_2$, and $\xi_1 \times \xi_2$ are limited. Chapter 5 details the justification for restrictions on $\tau_1 \times \tau_2$.

In the default setting, the reconstruction passband is equivalent to the representation passband, and, in addition, the display size is never allowed to exceed 512 × 512. Only frequency domain matrix representations are stored, and two-quadrant complex conjugacy is utilized, making it necessary to store only half of each matrix representation. The aperiodic (infinite) matrix representations of the aliased components of $\hat{p}', \hat{p}$, and $\hat{q}$ are stored as cropped data structures, the sizes of which are defined by $\tau_1 \times \tau_2$ as opposed to $\tau_{r_1} \times \tau_{r_2}$. With these precautions in place, use of a 512 × 512 sampling grid with representation and reconstruction passbands of $2N_1 \times 2N_2$ produces an acceptable (though not spectacular) response time on a 500 MHz CPU and 0.5GB RAM machine. Use of a sampling grid that size is also possible on a system with only 64MB of RAM, provided $\tau_1 \times \tau_2 = N_1 \times N_2$. If the size of the input scene matrix representation in the environment's library is small (64 × 64), and $\tau_1 \times \tau_2 < 3N_1 \times 3N_2$, use of a 1024 × 1024 sampling grid is technically possible, even on a machine with less RAM, but it is not very useful, because large, zero-padded data structures are created unnecessarily.

The difficulty with larger matrix representations from the environment's library is that the library data is stored in memory to speed the system response time when the user changes the value of $\tau_1 \times \tau_2$, $N_1 \times N_2$, or $\xi_1 \times \xi_2$. This would not be necessary, of course, if it could be guaranteed the computer system utilized had a fast local disk. The difficulty is in maintaining a balance which will allow the environment to be useful on a range of hardware specifications. The response time, therefore, and the limits at which the environment grinds to a halt because it can no longer allocate sufficient memory, vary from system to system. However, in general, use of a 512 × 512 sampling grid can be achieved on contemporary machines.

6.1.13 Good graphical user interface design

Every effort was made to make the environment user-friendly and intuitive. A menu-bar with the expected functionality is provided, as well as many point-and-click shortcuts. A Help menu explains some of the less-obvious operational features, but the contents would undoubtedly be considered inadequate were this a commercial offering. Consultation on the human computer interface (HCI) aspect of the GUI suggested that topic should have been considered prior to software development, and the user interface developed independently of, but in parallel with, the c/d/c model software. The subject was addressed at too late a stage to have significant impact on the basic design of the environment, but a redesign to incorporate features that would not be transparent to the user was undertaken.

6.1.14 Independence from proprietary applications

The environment was developed in ANSI C as an X Windows application for UNIX platforms. The implementation incorporates raw X commands [13, 14] with no other software restrictions on the operation of the environment. However, the pre-processor described in

Section 2.2.3.1 requires that a file submitted for inclusion in the environment's library be in PGM (ascii) format. Because *xv* [3] is currently supplied with most Unix-based operating systems and is capable of displaying many different image formats on all X displays known to the *xv* author, the pre-processor instructs users on how to utilize this application to convert their files to the required format. Reinventing an inferior wheel did not appear justified, in this instance, to satisfy a condition originally intended to maintain versatility.

6.1.15 Portability among UNIX/Linux operating systems

The environment runs without software modification on Linux, Irix, and Sun operating systems. The only problems encountered in switching among machines and/or operating systems are differences in font availability and display size. The visual display is less aesthetically pleasing when the software is forced to revert to a default font instead of one it has been designed to utilize, or when the specified font size increases or decreases relative to the number of pixels provided by the display.

6.2 Conclusions

The difficulties inherent in extending the 1-D c/d/c model to 2-D in an interactive environment require careful resolution of the computer-age-old time versus memory conundrum. Chapter 4 describes the process by which a balance between the two conflicting criteria was sought and attained. In particular, the results of a systematic study of passband limits are described in detail in Section 4.2.1.1, leading to the conclusion that the aliasing contribution from frequencies beyond $\tau_1 \times \tau_2 = 2N_1 \times 2N_2$ is invariably negligible. This makes a strong case for eliminating larger values of $\tau_1 \times \tau_2$ from the implementation, thus imposing an upper limit on memory requirements.

6.2.1 Towards improving time efficiency

From repeated operation of the environment, it became evident that the time savings which would prove most significant to the operating response time experienced by the user (when that response time is within bounds acceptable to the user of an interactive environment). would be any which would improve the efficiency of the FFT (i.e., a faster algorithm). Because a generic 2-D FFT is used in the environment, the possibility exists that a highly-optimized algorithm could improve the system response time when utilizing large sampling grids. However, no focus was concentrated in this area because, when the sampling grid is larger than 512 × 512, the response time of all the algorithms operating on the matrix representations becomes unacceptable.

6.2.2 Towards reducing memory requirements

Operation of the environment also suggests that storing the matrix representation of only the input scene might be a feasible alternative, when the sampling grid is not too large (i.e., in the cases where response time is currently acceptable). Storing only the input scene matrix representation may have little impact on operating response time in those cases, because the FFT that is always used to display a scene in the spatial domain is more CPU-time intensive than the processing algorithms. However, once again, when the sampling grid is larger than 512 × 512 and unacceptable response times are encountered, the effects of this change on system response time would merely compound an already bad situation, as the operation of all the processing algorithms slows to unacceptable limits.

6.3 Future Development

In the course of developing the environment, modifications which would improve its utility as a tool for end-to-end digital imaging system design continually became apparent. Many were implemented — and immediately suggested other desirable changes. The current state of development is one which in most cases fulfills, and in some cases surpasses, the original specifications, but is by no means definitive.

The currently identified possibilities for future development fall into three categories:

1. functionality extensions;

2. redesign;

3. library expansion.

This section addresses features not implemented and changes of a recurring nature.

6.3.1 Functionality extensions

The functionality of the environment would be enhanced by adding or improving some features.

- The environment's menu of synthesized input scenes could be extended.

- The software could be modified to accept filters which are not separable. This would, however, preclude the operation of 1-D mode while a non-separable filter type was selected. Non-separable filters would also have a negative impact on memory requirements and/or response time, because currently only the horizontal and vertical components of each filter are stored, and the 2-D filter coefficients are computed as required for processing, then freed. The 2-D coefficients of a non-separable filter would either have to be stored (impacting memory requirements) or recalculated each time they are required for processing (impacting system response time).

- The capability to cascade several filters as the contents of a single processing box could be considered, and the digital filter box could provide an option that would allow the user to define a small kernel in the spatial domain as an alternative to making a selection from the menu of filters predefined in the frequency domain.

- Additional noise models could be added.

- A 2-D "zoom" feature could be implemented, allowing the user to magnify a region of interest (ROI) in the displayed scene. It is projected that operation would be comparable to the "zoom" feature in the 1-D case. (See Section 6.1.7.4.)

- Two "levels" of Save could be created. The top level Save would retain the current functionality described in Section 6.1.8 and assign a .sav filename to a stored system state for long-term retrieval via the Retrieve command. A secondary level Save would allow the user LIFO access to a most-recent-Saves list via the ↑ and ↓ keys.

- A pop-up note could appear on-screen, when the cursor crosses into an active area, to describe the area's functionality to the user.

6.3.2 Redesign

The Microsoft Windows operating system is currently so readily available to computer users worldwide that a version of the environment capable of running under MS Windows is worthy of consideration. Relative to that, it should be noted that the environment utilizes approximately 50 raw Xlib functions and 10 Xlib structures which would require replacement or conversion. Other than that, the environment is written strictly in ANSI C and might port easily to the MS Windows OS. However, this avenue has not been explored.

If the human computer interface aspect of developing a GUI had been considered prior to software development, the user interface might have been developed independently of. but in parallel with, the c/d/c model software. HCI design techniques [26] could be employed to reevaluate the GUI. In particular, revamping the software into object-oriented components which would separate operation of the interface from c/d/c model processing would be desirable. The possibility of allowing the simple addition of modular functions by the user would be an additional useful feature.

6.3.3 Library expansion

The expansion of the environment's library is included here as a change of a recurring nature. It is envisaged that users would add input scenes, as required, by means of the pre-processor which adapts digitized images to the environment's library format. Input scenes of any size are accepted by the pre-processor. The PGM (ascii) input data is converted to a frequency domain matrix representation with dimensions which are powers of 2, and the cropped data structure is saved in a file in the environment's library. The user can choose to clip, zero-pad, pad with the image mean, periodically replicate, or resample the original image, if resizing is necessary in the pre-processor, to achieve the appropriate dimensions.

Bibliography

[1] BANHAM, M. R. AND A. K. KATSAGELLOS. Digital Image Restoration. *Signal Processing*, Volume 14, No. 2, pages 24-41, 1997.

[2] BIBERMAN, L. M. (ED.) *Perception of Displayed Information.* Plenum Press, New York, 1973.

[3] BRADLEY, J. *xv* — Interactive Image Display for the X Window System. University of Pennsylvania, 1994.

[4] COOLEY, J. W., AND J. W. TUKEY. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, Volume 19, pages 297–301, 1965.

[5] FALES, C. L., F. O. HUCK, R. ALTER-GARTENBERG, AND Z. RAHMAN Image Gathering and Digital Restoration. *Philos. Trans. R. Soc. London*, Ser. A 354, pages 2193–2248, 1996.

[6] HADAR, O., AND G. D. BOREMAN. Oversampling Requirements for Pixelated-Imager Systems. *Optical Engineering*, Volume 38, No. 5, pages 782–785, May 1999.

[7] HALFORD, C. E., K. A. KRAPELS, R. G. DRIGGERS, AND E. E. BURROUGHS, JR. Developing Operational Performance Metrics Using Image Comparison Metrics and the Concept of Degradation Space. *Optical Engineering*, Volume 38, No. 5, pages 836–844, May 1999.

[8] HAZRA, R. *Constrained Least-Squares Digital Image Restoration.* PhD thesis, College of William & Mary, Williamsburg VA, 1995.

[9] HUCK, F. O., R. ALTER-GARTENBERG, S. K. PARK, AND Z. RAHMAN Information-theoretic assessment of sampled imaging systems. *Optical Engineering*, Volume 38, No. 5, pages 742–762, May 1999.

[10] IDEMA, M. *Sub-Pixel Techniques to Enhance Spatial Resolution.* PhD thesis, College of William & Mary, Williamsburg VA, in progress.

[11] JACOBS, E. L., AND T. C. EDWARDS. Sampling Criteria for Sensor Simulation. *Optical Engineering*, Volume 38, No. 5, pages 827–835, May 1999.

[12] JAIN, A. K. *Fundamentals of Digital Image Processing.* Prentice-Hall, Englewood Cliffs NJ, 1989.

[13] NYE, A. *Xlib Programming Manual for Version 11 of the X Window System.* The Definitive Guides to the X Window System, Volume 1, Third Edition. O'Reilly & Associates, Inc., U.S.A., 1995.

[14] NYE, A. (ED.) *Xlib Reference Manual.* Third Edition for X11, Release 4 and Release 5; The Definitive Guides to the X Window System, Volume 2, Third Edition. O'Reilly & Associates, Inc., U.S.A., 1993.

[15] ODEH, AND J. EVANS. A very accurate approximation of the Normal idf. *Applied Statistics*, Volume 23, pages 96–97, 1974.

[16] PARK, S. K. Image Gathering, Interpolation and Restoration: A Fidelity Analysis. *SPIE Proceedings*, Volume 1705, pages 134–144, April 1992.

[17] PARK, S. K. FLIR Range Enhancement by Constrained Least-Squares Restoration. Department of Computer Science, College of William & Mary, Williamsburg VA, for U.S. Army Research Office, 1995.

[18] PARK, S. K. Lecture Notes on DIGITAL IMAGE PROCESSING, Version 2.4. Department of Computer Science, College of William & Mary, Williamsburg VA, 1995.

[19] PARK, S. K. Lecture Notes on FOURIER METHODS: Modeling, Simulation and Performance Analysis for Discrete Linear Systems, Version 4.1. Department of Computer Science, College of William & Mary, Williamsburg VA, 1996.

[20] PARK, S. K., AND D. GEYER. ANSI C library for random number generation. Department of Computer Science, College of William & Mary, Williamsburg VA, 1996.

[21] PARK, S. K., AND R. HAZRA. Image Restoration Versus Aliased Noise Enhancement. *SPIE Proceedings*, Volume 2239, pages 52–62, April 1994.

[22] PARK, S. K., AND R. HAZRA. Incomplete System Models Can Cause Image Restoration Failures. *SPIE*, Volume 2488, pages 22–33, 1995.

[23] PARK, S. K., AND K. MILLER. Random Number Generators: Good Ones Are Hard To Find. *Communications of the ACM*, Volume 31, No. 10, pages 1192–1201, October 1988.

[24] PARK, S. K., AND Z. RAHMAN. Fidelity Analysis of Sampled Imaging Systems. *Optical Engineering*, Volume 38, No. 5, pages 786–800, May 1999.

[25] REICHENBACH, S. E., S. K. PARK, R. ALTER-GARTENBERG, AND Z. RAHMAN Artificial Scenes and Simulated Imaging. *Stochastic and Neural Methods in Signal Processing, Image Processing, and Computer Vision: Proceedings of the International Society for Optical Engineering*, Volume 1569, pages 422–433, San Diego CA, July 1991.

[26] RUDISILL, M., L. CLAYTON, P. B. POLSON, AND T. D. MCKAY (EDS.) *Human-Computer Interface Design.* Morgan Kaufmann Publishers, Inc., San Francisco CA, 1996.

[27] SCHREIBER, W. F. *Fundamentals of Electronic Imaging Systems: Some Aspects of Image Processing.* Springer-Verlag, New York, 1986.

[28] VOLLMERHAUSEN, R., R. G. DRIGGERS, AND B. L. O'KANE Influence of Sampling on Target Recognition and Identification. *Optical Engineering,* Volume 38, No. 5, pages 763–772, May 1999.

[29] AFDPLUS. Webb Laboratories, Brookfield WI, 1997.
about@webblabs.com, http://www.webblabs.com/afdplus.htm

[30] Image Toolbox with Khoros Pro 2001. Khoral Research Inc., Albuquerque NM, 1999.
info@khoral.com http://www.khoral.com

[31] PV-WAVE Extreme Advantage 7.0. Visual Numerics Inc., Boulder CO, 1999.
info@boulder.vni.com http://www.boulder.vni.com/products/wave

[32] SAL (Scientific Applications on Linux). a web page of free software administered by Kachina Technologies.
sal@kachinatech.com http://sal.kachinatech.com

[33] System Image Analyzer, Version 1. JCD Publishing, Winter Park FL, 1996.

[34] SystemView. ELANIX Inc., Westlake Village CA, 1997.
systemview@elanix.com http://www.elanix.com/software_nf.html

VITA

Moira Joyce Turner (née Duffy) was born in Edinburgh, Scotland on January 18th, 1949. After graduating in 1970 with a B.Sc. (Hons.) degree in Computer Science from one of the first such degree programs ever offered in Scotland, she went to work for Marconi Space and Defence Systems as a Simulation Software Design Engineer, writing software for nuclear-powered submarine control room simulators and cockpit trainers. Her work in flight simulation led to involvement in the establishment of the A300B training program at Aeroformation, the training division of Aerospatiale in Toulouse, France. At the end of that two-year assignment, Ms. Turner returned to the United Kingdom where she worked first for Singer/Link UK Ltd. and subsequently for Ferranti Ltd., designing software for both fixed wing and helicopter simulators, mainly for military customers. In 1977, she emigrated to Canada to work for CAE (formerly Canadian Aviation Electronics) in Montreal. Three years later, she was appointed group leader of a 35-man team that designed the secondary systems for nuclear power plant control room simulators. In 1983, Ms. Turner succumbed to an abrupt career change from Design Engineering Group Leader to full-time mother and simultaneously emigrated once more, this time to the United States. In 1993, she determined to earn a Masters degree. Ms. Turner obtained an M.S. in Computer Science in 1995 but then chose to continue the research on which she had embarked at the College of William and Mary. She lives with her husband and two children in Chesterfield, Virginia.