

SBIR- 05.01-4100
release date 7/29/92 ✓

NAS7-1036
Final Technical Report 2168

ORIGINAL

**HOLOGRAPHIC ENHANCED
REMOTE SENSING SYSTEM**

August 1990

SCIENTIFIC AND TECHNICAL REPORT

N93-18230

Unclas

G3/43 0121179

Submitted to:

**National Aeronautics and Space Administration
Dr. Neville I. Marzwell
Contracting Officer's Technical Representative
Jet Propulsion Laboratory
Pasadena, CA**

Prepared by:

**Helene P. Iavecchia
Edwin S. Gaynor
Lloyd Huff
William T. Rhodes
Edward H. Rothenheber**

(NASA-CR-190867) HOLOGRAPHIC
ENHANCED REMOTE SENSING SYSTEM
Final Technical Report No. 2168
(Analytics) 272 p



Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle Holographic Enhanced Remote Sensing System		5. Report Date August 1990	
		6. Performing Organization Code	
7. Author(s) Helene P. Iavecchia, Edwin S. Gaynor, Lloyd Huff, William T. Rhodes and Edward H. Rothenheber		8. Performing Organization Report No. Final Technical Report 2168	
		10. Work Unit No.	
9. Performing Organization Name and Address Analytics 2500 Maryland Rd. Willow Grove, PA 19090		11. Contract or Grant No. NAS7-1036	
12. Sponsoring Agency Name and Address NASA Resident Office JPL 4800 Oak Grove Drive Pasadena, CA 91109		13. Type of Report and Period Covered Final Technical Report July 1988 - July 1990	
		14. Sponsoring Agency Code	
15. Supplementary Notes			
16. Abstract The Holographic Enhanced Remote Sensing System (HERSS) consists of three primary subsystems: (1) an Image Acquisition System (IAS); (2) a Digital Image Processing System (DIPS); and (3) a Holographic Generation System (HGS) which multiply exposes a thermoplastic recording medium with sequential 2-D depth slices that are displayed on a Spatial Light Modulator (SLM). Full-parallax holograms were successfully generated by superimposing SLM images onto the thermoplastic and photopolymer. An improved HGS configuration utilizes the phase conjugate recording configuration, the 3-SLM-stacking technique and the photopolymer. The holographic volume size is currently limited to the physical size of the SLM. A larger-format SLM is necessary to meet the desired 6 inch holographic volume. A photopolymer with an increased photospeed is required to ultimately meet a display update rate of less than 30 seconds. It is projected that the latter two technology developments will occur in the near future. While the IAS and DIPS subsystems were unable to meet NASA goals, an alternative technology is now available to perform the IAS/DIPS functions. Specifically, a laser range scanner can be utilized to build the HGS numerical database of the objects at the remote work site.			
17. Key Words (Suggested by Author(s)) Holography Spatial Light Modulator Thermoplastic Camera Remote Sensing 3-D Display		18. Distribution Statement	
19. Security Classif. (of this report) UNCLASSIFIED	20. Security Classif. (of this page) UNCLASSIFIED	21. No. of pages	22. Price

EXECUTIVE SUMMARY

This report describes the development of a prototype Holographic Enhanced Remote Sensing System (HERSS). HERSS consists of three primary subsystems: (1) an Image Acquisition System (IAS) to acquire video images of a remote worksite; (2) a Digital Image Processing System to process the image data and build a numerical base of the object surface points; and (3) a Holographic Generation System (HGS) to display a full-parallax view of the remote objects in near-real-time (i.e., in less than 30 seconds). HGS uses a Spatial Light Modulator (SLM) as the object source and thermoplastic as the recording medium. Several alternative configurations were developed and evaluated for each subsystem. For the IAS, two optical imaging designs were developed — an afocal telecentric lens system and a non-afocal-telecentric lens system. For the DIPS, three algorithms were pursued — Frieden, Hausler, and Z-contrast. For the HGS, two primary holographic recording configurations were evaluated — baseline and phase conjugate. Other HGS experimentation included use of a 3-SLM-stack exposure method as well as use of a new Du Pont photopolymer recording material. A key advantage of the stacking technique was a reduction in the time to generate a holographic frame.

Full-parallax holograms were successfully generated by superimposing SLM images onto the thermoplastic and photopolymer. Each SLM image contained the object information associated with a unique depth plane at the remote work site. The multiple exposure capacity of the thermoplastic was found to be limited. With the 3-SLM-stacking technique, the storage capacity of the thermoplastic was doubled. However, other limits of the thermoplastic precluded its use in the ultimate system. An improved HGS configuration utilizes the phase conjugate recording configuration, the 3-SLM-stacking technique, and the photopolymer. The holographic volume size is currently limited to the physical size of the SLM. A larger-format SLM is necessary to meet the desired 6.0-inch holographic volume. A photopolymer with an increased photospeed is required to ultimately meet a display update rate of less than 30 seconds. It is projected that the latter two technology developments will occur in the near future. While the IAS and DIPS subsystems were unable to meet NASA goals, an alternative technology is now available to perform the IAS/DIPS functions. Specifically, a laser range scanner can be utilized to build the HGS numerical data base of the objects at the remote work site.

The HERSS concept has potential for use as a teleoperations 3-D display device. The system can ultimately be used to perform space maintenance activities, remote operations in hazardous environments, and to handle dangerous materials.

ACKNOWLEDGEMENT

The authors would like to acknowledge the important role of Dr. Neville Marzwell of the Jet Propulsion Laboratory in monitoring and guiding this program. The authors also acknowledge the technical contributions of Dr. Kristina Johnson of the University of Colorado and Dr. B. Roy Frieden of the University of Arizona in the development of the Holographic Generation System and the restoration algorithms of the Digital Image Processing System. We are also most grateful to the Optoelectronic Computing Center at the University of Colorado at Boulder and the Electrical Engineering Department at the Georgia Institute of Technology for use of laboratory equipment in the evaluation of alternative HERSS optical configurations. The continued support of colleagues at Analytics, especially Regina Harris and William Arnold, is recognized and greatly appreciated. The authors would also like to thank Phyllis Martin, Harry Oliveri, and Heidi Langdon for their efforts in the production of this report.

TABLE OF CONTENTS

1.	INTRODUCTION	
1.1	Background and Phase I Analysis.....	1-2
1.1.1	3-D Display Technologies.....	1-2
1.1.2	Phase I Analyses.....	1-4
1.2	Organization of the Report.....	1-7
2.	PHASE II: DESIGN GOALS AND TECHNICAL APPROACH	
2.1	Phase II Design Goals.....	2-1
2.2	Phase II Approach.....	2-3
3.	IMAGE ACQUISITION SYSTEM (IAS)	
3.1	The Baseline System: Afocal-Telecentric Imaging.....	3-2
3.2	Alternative: Non-Afocal-Telecentric.....	3-7
3.3	Summary — IAS.....	3-13
4.	DIGITAL IMAGE PROCESSING SYSTEM (DIPS)	
4.1	DIPS Preprocessing for Non-AT Data Bases.....	4-3
4.2	DIPS Alternative: Frieden.....	4-4
4.3	DIPS Alternative: Z-Contrast.....	4-7
4.4	DIPS Alternative: Hausler's Algorithm.....	4-13
4.5	Performance of the Image Processing Algorithms.....	4-14
4.6	Alternative IAS/DIPS — Laser Range Scanners.....	4-16
4.7	Summary.....	4-19

TABLE OF CONTENTS
(continued)

5. HOLOGRAPHIC GENERATION SYSTEM (HGS)

- 5.1 Detailed Optical Design of the Baseline System..... 5-3
- 5.2 Review of the State-of-Technology of SLMS and
Near-Real-Time Recording Materials 5-7
- 5.3 Early Proof-of-Concept and Component Testing at the
University of Colorado5-11
 - 5.3.1 Experimentation5-11
 - 5.3.2 Results5-12
- 5.4 Baseline Development at the Analytics Laboratory5-16
 - 5.4.1 Experimentation5-17
 - 5.4.2 Results5-17
- 5.5 Phase Conjugate Configuration.....5-19
 - 5.5.1 Experimentation5-25
 - 5.5.2 Results5-26
- 5.6 Multiplane-by-Multiplane Exposure Scheme.....5-30
 - 5.6.1 Experimentation5-32
 - 5.6.2 Results5-33
- 5.7 HGS Alternative Recording Material: DuPont Photopolymer5-35
 - 5.7.1 Experimentation5-36
 - 5.7.2 Results5-37

6. CONCLUSIONS AND RECOMMENDATIONS

LIST OF FIGURES

Figure 2-1	HERSS Subsystem Functions	2-2
Figure 2-2	HERSS Technical Approach.....	2-5
Figure 3-1	Image Acquisition Scheme	3-2
Figure 3-2	Space-Invariant Region (shaded).....	3-4
Figure 3-3	IAS Baseline: Afocal-Telecentric Configuration.....	3-6
Figure 3-4	IAS Baseline: Test Object for AT Data Collection	3-8
Figure 3-5	IAS Alternative: Non-Afocal-Telemetric Configuration.....	3-11
Figure 3-6	IAS Alternative: Test Object for non-AT Data Collection.....	3-12
Figure 4-1	Sitter Coordinate Transformation Model for non-AT Data Bases	4-3
Figure 4-2	Results of Z-Contrast Algorithm Processing of the AT Data Base: (a) Shape Functions (b) Brightness Functions	4-10
Figure 4-3	Z-Contrast Results Cross-section of Shape Function Gray-Scale Image	4-9
Figure 4-4	Results of Z-Contrast Algorithm Processing of the non-AT Data Base (No Sitter Correction): (a) Shape Function, (b) Brightness Function	4-12
Figure 4-5	Results of Z-Contrast Algorithm Processing of the non-AT Data Base (with Sitter Correction): (a) Shape Function, (b) Brightness Function	4-14
Figure 5-1	Holographic Generation Scheme	5-2
Figure 5-2	HGS Baseline: Optical Configuration.....	5-4

LIST OF FIGURES
(continued)

Figure 5-3	Multiple Exposure of a Square Annulus through Depth to Create a 3-D Pyramid	5-6
Figure 5-4	Structure and Recording Steps of a Thermoplastic Material.....	5-9
Figure 5-5	HGS Alternative: Phase Conjugate Optical Configuration.....	5-21
Figure 5-6	Photodocumentation: Holograms Recorded on Thermoplastic.....	5-27
Figure 5-7	Photodocumentation: Holograms Recorded on DuPont Photopolymer.....	5-38

1. INTRODUCTION

Three-dimensional (3-D) display of a remote worksite during teleoperations can enhance the telepresence experienced by human controllers. Enhanced telepresence is a feeling of "being there" which can translate to a more accurate perception of the relative distances between objects at the worksite. Currently-employed display systems utilize multiple two-dimensional (2-D) TV camera views of the remote site require the teleoperator to form a mental map of the 3-D world. This results in slow operations. Furthermore, the user typically cannot position the end effector of the remote manipulator system closer than six or seven inches of the desired location. Three-dimensional holographic displays have the potential to offer shorter performance time, improved safety, and expansion of teleoperations to close range tasks that would normally require extra-vehicular activity (cf., Iavecchia, Arnold, Gaynor, Rhodes, and Rothenheber, 1987).

This report describes the development of a prototype Holographic Enhanced Remote Sensing System (HERSS) to:

- Acquire video images of a remote worksite;
- Process the image data to build a numerical base of the object surface points; and
- Display a view of the surfaces holographically in near real time.

The work was performed as part of a Phase II research effort awarded by the National Aeronautics and Space Administration (NASA) under the Small Business Innovation Research (SBIR) Program, contract NAS7-1036, monitored by the Jet Propulsion Laboratory.

1.1 BACKGROUND AND PHASE I ANALYSES

The objective of the Phase I effort, performed in 1987, was to investigate the feasibility of using a holographic-based system for near-real-time display of a remote worksite. Phase I technical tasks included:

- Review NASA teleoperations and operating environments,
- Review visual factors in depth perception and determine display design criteria relevant to space-based NASA teleoperations,
- Compare alternative 3-D display technologies with particular emphasis on human interface issues,
- Review the state-of-technology in holographic display generation,
- Assess the applicability of existing holographic techniques for a NASA teleoperations display system, and
- Develop a HERSS system architectural concept considering application requirements.

The review of NASA teleoperations requirements documented increases in the number and complexity of future space operations. Telerobotic servicers are planned for a gamut of tasks including spacecraft servicing and structural assembly, as well as for contingency events. These servicers are to be controlled by astronauts from the interior of a space station. A clear need for the development of displays to enhance telepresence at the telerobotic workstation was identified. Prime candidates for enhanced telepresence are 3-D display technologies. A brief comparative review of current 3-D display technologies is provided below followed by a summary of the results of the Phase I Effort.

1.1.1 3-D Display Technologies

Three-dimensional display technologies fall into three major categories: stereo pair, multiplanar, and holographic (Hodges and McAllister, 1987). **Stereo pair displays** present unique perspective views to the right and left eyes corresponding to the views that would normally be seen due to the horizontal eye separation. The disparate images presented to each eye are fused by the brain through a process known as stereopsis that results in the perception of a solid 3-D object. **Multiplanar displays** create the perception of a solid 3-D object by

rapidly and sequentially projecting flat images, each representing a specific depth plane, into a volumetric space. One multiplanar implementation uses a vibrating varifocal mirror to alter the distance at which each depth plane is projected (Sher, 1990; Traub, 1967). Another implementation uses an acousto-optic modulator to direct a laser beam across the surface of a rotating disc to address particular points within an image volume (Williams and Garcia, 1988; 1989). **Holographic displays** provide 3D images by presenting to the viewer an optical wavefront that duplicates what would be seen by an observer who is directly viewing an object volume.

Various advantages and disadvantages are associated with each of the 3-D display technologies. Although stereographic systems can provide color display of solid 3-D objects, they typically only provide a veridical perspective view from only one observation position. Stereographic systems can provide a correct perspective view for every observation position, that is, full parallax, but this requires use of sophisticated equipment and processing. For systems that utilize two cameras to gather right and left eye views at a remote worksite, full parallax is achieved with a head-tracking device that slaves right and left cameras to head movement (Cole, Pepper, and Pinz; 1981). For systems that use a computer to generate right and left eye views of an underlying numerical data base, a computationally-intensive algorithm is used to compute the correct right- and left-eye perspectives. Viewer discomfort has been reported with the use of such stereographic systems including headache and motion sickness symptoms. This could lead to an increase in the occurrence of space adaptation syndrome. Finally, 5% of the population is unable to fuse stereographic images and another 10% may have problems using the display because of stereopsis deficiencies.

Multiplanar and holographic displays can provide inherent full parallax. However, multiplanar systems are typically limited to wireframe and translucent images as well as an update rate of approximately 10Hz. The primary drawback for holographic systems is the time required to create a 3-D image. Typically, the interference pattern of an object wavefront and a reference wavefront must be recorded. In the absence of a real object to produce the object wave, some form of computer generation is necessary. Techniques have been developed to compute the interference, or diffraction, pattern associated with a particular object (Weingartner, 1983). However, the technique is computationally-intensive and time

consuming. Aerodyne Research, Inc., in an attempt to minimize the computational effort, used an interferometric system to record the exposure patterns associated with different points of the object. (Caulfield, 1983; Gaynor, Rhodes, and Caulfield, 1987). The process is fairly efficient, though still time-consuming because the interference patterns are recorded one by one for each object point. Even this holographic display system could not approach near-real-time update rates, that is, with an update in less than a minute.

1.1.2 Phase I Analyses

The Phase I design goals and technical efforts were driven by the identified NASA goals and requirements for space-based telerobotics including (cf., Iavecchia, et al., 1987):

1. A full-parallax, holographic image should be created;
2. The 3-D data base representing the object space should be determined with minimal computational effort and should be based on real and reliable data;
3. Operator safety and comfort must be maximized; and
4. An inherent capability for direct graphic overlay onto the holographic display should be provided for integrated data display.

The Phase I analysis of holographic, as well as supporting electronics and recording material technologies, indicated that near-real-time holographic display was possible. The proposed Holographic Enhanced Remote Sensing System (HERSS) uses a video camera to collect image data at the remote worksite. The video data is processed to create a 3-D numerical data base of the surfaces of the objects at the remote worksite. Prior to holographic generation, the numerical data base is "sliced" into 2-D image planes with each 2-D plane representing the surface points at a unique depth position. During holographic exposure, each 2-D image plane is sequentially transmitted to a spatial light modulator. The image on the spatial light modulator acts as the object source for the hologram. Each 2-D image plane is sequentially exposed onto a thermoplastic recording material, in a plane-by-plane fashion until the entire object volume is recorded and one holographic frame is complete.

A key innovation of this concept was the use of photosensitized thermoplastic as the recording material. Thermoplastic materials are capable of developing a holographic frame following exposure in less than a minute and are reusable. Another key feature of the design was the use of a computer-addressable spatial light modulator to hold image data for a particular depth plane while acting as the "coherent" object source for the hologram. The plane-by-plane recording scheme had three major advantages:

- (a) The potential to quicken the generation of a single holographic frame because the information was being recorded object-plane-by-object-plane instead of point-by-point as in the Aerodyne technique.
- (b) The potential to increase the amount of information contained in a single holographic frame because more information could be recorded in a single exposure. All recording materials have limits to the number of multiple exposures that can be recorded before serious image degradation occurs.
- (c) Near-real-time holographic recording would be possible.

To achieve the first goal – near-real-time display update – a spatial light modulator (SLM) was selected as the holographic image source and thermoplastic was selected as the holographic medium. Two-dimensional depth slices of the surface points of the object space could be sequentially displayed on the SLM and exposed onto the thermoplastic. The thermoplastic material is capable of rapid development and erasure of a holographic image and has the capability to store/erase at least 100 holographic frames per sheet.

Concerning the second goal – minimization of computational effort in acquiring image data – an afocal telecentric image acquisition system was proposed. An afocal configuration provides constant transverse and longitudinal magnification throughout the image volume. Thus, software algorithms to correct for image distortions caused by non-uniform magnification (as would occur with conventional imaging schemes) would not be necessary. If the system is also telecentric, the imaging operation is space-invariant in three dimensions, further simplifying the required image processing software. Further, the image acquisition and processing system design scheme avoids reliance on template matching and prediction algorithms to recognize remote object shapes. Instead, the video image slices (containing both blurred and in-focus information) are analyzed to determine true surface point locations.

With respect to the third goal – operator safety and comfort – it was determined that the holographic development process would not pose a health threat. That is, the normal thermoplastic development process would not result in the emission of uncontrolled toxins into the operator's workstation (Neville, Marzwell, personal communications, 1987).

The fourth goal – graphic overlay capability onto the holographic image – was readily achievable. Because the source of the holographic image is a numerical data base of the surface points of the remote object space, graphic data can be inserted anywhere in the volumetric data base as appropriate.

In summary, an analysis of the proposed design indicated the following:

- Three-dimensional imaging provides an opportunity for enhanced telepresence, reduced operator workload, improved mission efficiency and safety, and expansion of teleoperational activities for NASA space-based operations.
- Stereography may be unsuitable for space-based operations because of its potential to induce motion sickness symptoms.
- Holography was a promising 3-D display technology suitable for space-based environments.
- Holography was capable of providing near-real-time 3-D display of the remote worksite for NASA teleoperation tasks.
- All components for the proposed near-real-time holographic display system were commercially available.
- Optical data acquisition could be achieved with an afocal-telecentric imaging system and a detector array located at the remote worksite.
- The HERSS system design concept achieved NASA short-term goals of enhanced telepresence using "real" image data acquired at the remote worksite with 3-D display of that data.
- The system design concept was compatible with NASA long-term goals for autonomous telerobotic control.

Therefore, the HERSS concept was judged to be feasible and could meet NASA design goals for space-based teleoperations.

1.2 ORGANIZATION OF THE REPORT

Based on the Phase I analyses, a Phase II effort was warranted to develop a prototype system for experimental investigation. The Phase II effort is described in this report as shown below:

- Section 2 — Phase II design goals and technical approach;
- Section 3 — Development of the Image Acquisition System (IAS);
- Section 4 — Development of the Digital Image Processing System (DIPS);
- Section 5 — Development of the Holographic Generation System (HGS);
- Section 6 — Conclusions and recommendations.

Appendix A describes the central computing system including hardware and software components, functions, and interactions. Appendix B contains detailed optical drawing of the IAS and HGS optical configurations. An auxiliary report (Janiszewski, Iavecchia, and Mathur, 1990) contains detailed information on the use of the HERSS software as well as programmer's documentation.

2. PHASE II: DESIGN GOALS AND TECHNICAL APPROACH

The overall purpose of the Phase II effort was to construct, demonstrate, and optimize a HERSS laboratory prototype. The three primary HERSS subsystems, as illustrated in Figure 2-1, are:

- (1) Image Acquisition System (IAS) to perform the sensing function at the remote worksite. The IAS acquires video images of sequential 2-D depth planes at the remote work site. The IAS uses a custom designed optical system and a detector array to collect the video data. Each video frame represents a 2-D depth slice at the resolution limit of the imaging system. Each 2-D depth slice contains both in-focus and out-of-focus information.
- (2) Digital Image Processing System (DIPS) to perform the image processing function. The DIPS processing algorithms remove the out-of-focus elements within a single depth plane while leaving the in-focus elements.
- (3) Holographic Generation System (HGS) to perform the 3-D display function. The HGS holographically records each depth plane on a thermoplastic material in near-real-time to create a 3-D view of the remote work site. The resultant HERSS image would theoretically provide near-real-time, full-parallax, volumetric display of remote objects.

The Phase II effort covered a two-year period commencing 31 July 1988 and ending 31 July 1990.

2.1 PHASE II DESIGN GOALS

Specific goals established for the Phase II effort were:

- The holographic display should provide a full-parallax view of the remote object.
- Object field size at the remote worksite should be a six inch cube. Similarly, the holographic display volume should equal a corresponding six inch cube.

FUNCTION

SUBSYSTEM

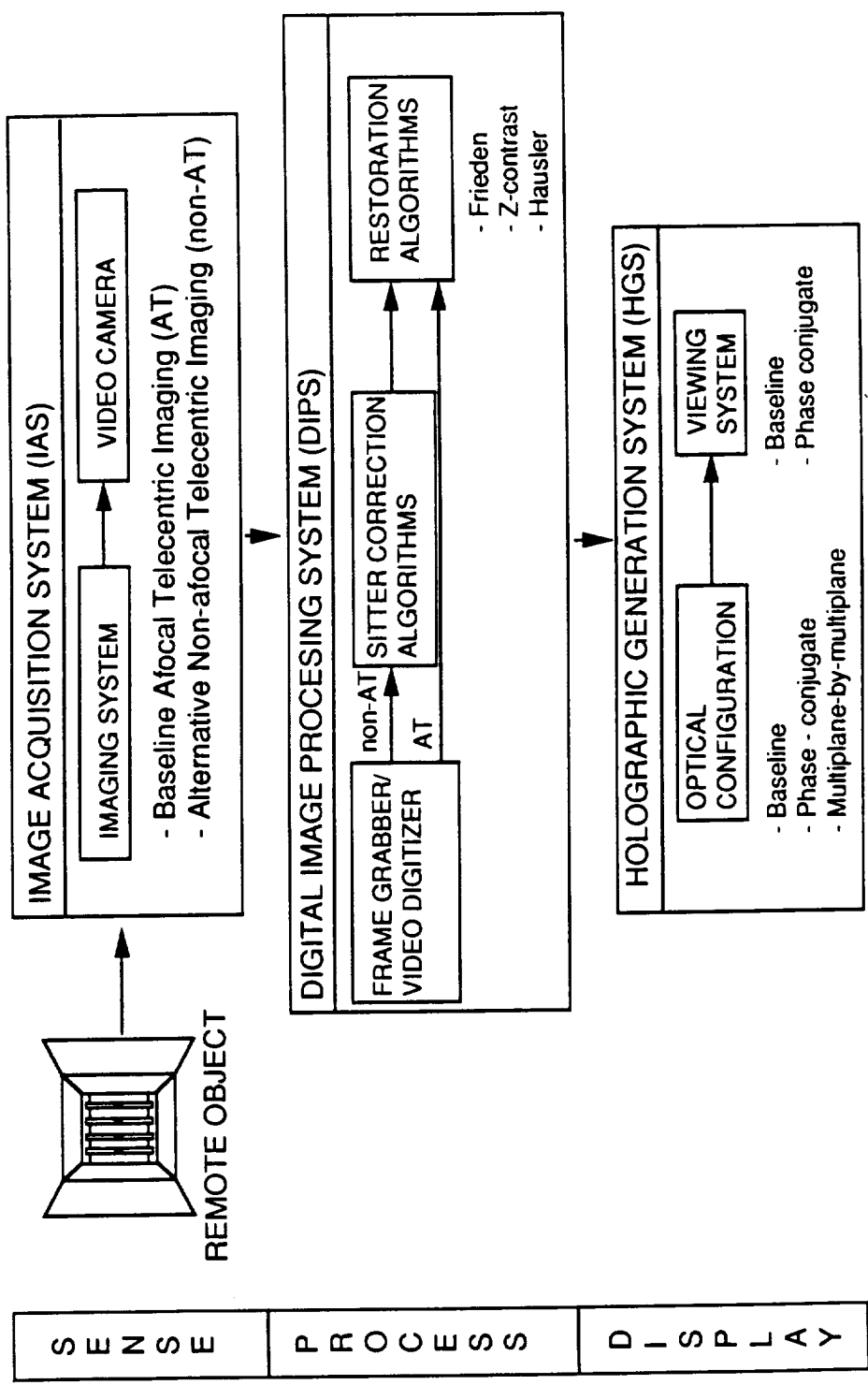


Figure 2-1. HERSS Subsystem Functions

- Object-space depth resolution should be 2 mm to 3 mm. Following conversations with NASA representatives in February, 1989, the resolution goal was reset to 0.1mm for the image acquisition process. A finer resolution would be critical to an eventual autonomous robotic system (a long term goal of NASA). Depth resolution for the holographic display remained at 2 mm to 3 mm.
- Holographic image quality should be optimized to the greatest extent possible.
- Holographic display update time should be 30 seconds or less.

2.2 PHASE II APPROACH

The Phase II effort comprised five major tasks as defined in the original Statement of Work (SOW). Three additional tasks (6 through 8) were incorporated into the SOW in 1989 (Spangenberg, 1989). These tasks included:

- Task 1. Equipment acquisition, installation, and test. This task included an initial survey of the state-of-technology of image processing equipment, SLMs, and thermoplastic materials to ensure that the most cost-effective and functional equipment would be acquired.
- Task 2. System design specification for the IAS, DIPS, and HGS as well as for the central computer hosting the entire system.
- Task 3. System integration and test.
- Task 4. System optimization. This task encompasses the establishment of baseline performance for each subsystem as well as the iterative system refinement.
- Task 5. System documentation.
- Task 6. Alternate IAS/DIPS design and test. This task was incorporated into the SOW in 1989 to develop an alternate image acquisition system that uses a single lens instead of a two-lens system in an afocal-telecentric configuration. In the report, this design alternative is referred to as the non-afocal-telecentric (non-AT) lens system. DIPS pre-processing algorithms were also developed in connection with this task to correct for magnification distortions concomitant with a single lens imaging system. These algorithms are referred to as the Sitter correction algorithms.
- Task 7. Alternate HGS optics design and test. This task was incorporated into the SOW in 1989 to allow development of an alternative hologram generation method using a phase conjugate recording configuration.

This design alternative is referred to as the phase conjugate recording configuration.

Task 8. Alternate HGS exposure design and test. This task was incorporated into the SOW in 1989 to allow development of a holographic exposure scheme that would increase the information content recorded on the thermoplastic. This design alternative is referred to as the multiplane-by-multiplane exposure scheme.

Figure 2-2 presents the approach taken in the execution of these tasks. The effort began with detailed design exercises including specification of the structure and functions of the IAS, DIPS, and HGS subsystems. A survey of the state-of-technology for the major equipment components of the system was also initiated following final design specification and functional requirements. Task 6 was added to the SOW following initial conclusions drawn from detailed design specification for the IAS. Specifically, it was determined that in order to achieve an imaging operation over a six-inch volume using an afocal-telecentric imaging system, large and prohibitively expensive lenses would be required. An alternative scheme was incorporated as Task 6 into the SOW to circumvent this limitation.

Preliminary proof-of-concept experiments were conducted early in the effort at the Optoelectronic Computing Center at the University of Colorado. A critical conclusion of the Colorado experiments was that the image quality of the hologram was severely degraded following 30 multiple exposures of the thermoplastic using the baseline plane-by-plane exposure scheme. Image aberrations were also noted. This led to the recommendation for the addition of Task 7 and 8 to the SOW.

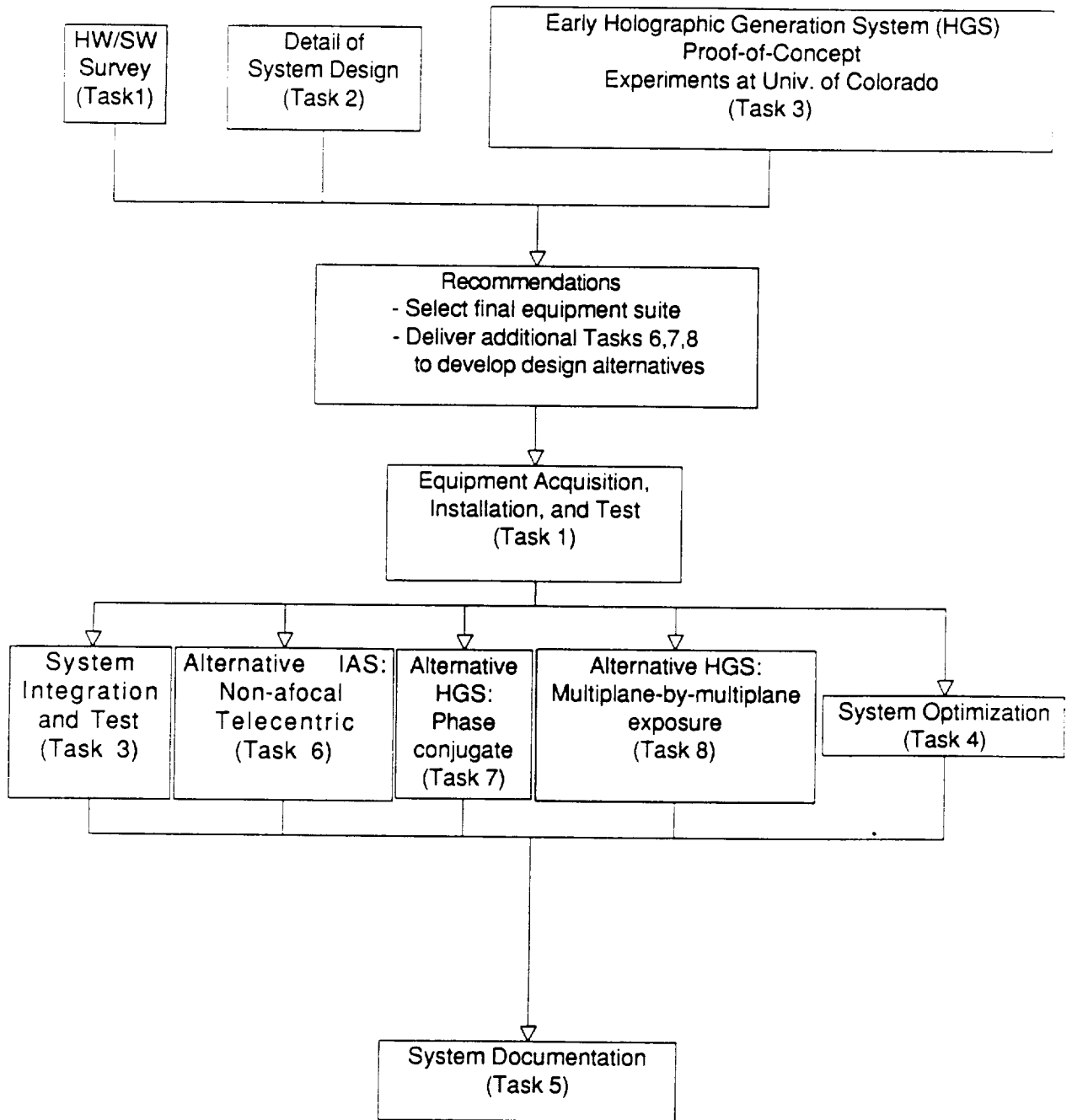


Figure 2-2. HERSS Technical Approach

In summary, in order to reduce the overall risk of the prototype development, several alternative configurations were developed for each subsystem. For the IAS, two optical imaging designs were developed — an afocal telecentric lens system and a non-afocal-telecentric lens system. For the DIPS, three algorithms were pursued — Frieden, Hausler, and Z-contrast. For the HGS, two primary configurations were evaluated — baseline and phase conjugate. For the latter configuration, further HGS experimentation included use of a multiplane-by-multiplane exposure method as well as use of a fly's eye lens. The following three sections present the results of development efforts for the IAS, DIPS, and HGS subsystems.

3. IMAGE ACQUISITION SYSTEM (IAS)

The primary goal of the IAS is image acquisition within a six-inch cubic volume with relatively fine depth resolution. The main components of the IAS are a custom imaging system and a single video camera detector array. As illustrated in Figure 3-1, the 2-D detector array, oriented perpendicular to the optical axis, is moved longitudinally in a sequence of steps through the image space. The step distance is determined by the Nyquist interval, which is the minimum stepping interval for which the sample data accurately represents the continuously distributed light distribution. The Nyquist interval in the z (depth) direction is essentially equal to the depth resolution of the imaging system. A video snapshot is taken at each depth plane. Each video image contains in-focus object information for a specific depth plane and out-of-focus information for all other depth planes. The full set of depth planes captured by the video camera are subsequently processed by the DIPS subsystem to determine the actual surface and brightness of the remote object. In this way the DIPS produces a numerical representation of the object.

During detailed system design (Task 2), it was determined that there were serious inherent limitations in the afocal-telecentric (AT) design configuration initially specified for the IAS. These limitations, not previously reported in the technical literature, are discussed below. Following discovery of the limitations, more conventional optical systems were considered for the IAS. Subsequently, Task 6 — development of an alternative IAS optical system based on non-afocal-telecentric (non-AT) imaging — was incorporated into the SOW. Non-AT optical systems produce geometrically distorted images where the magnification changes with position in image space, and a correction algorithm must be applied to the incoming image data to rectify sequential depth images. The nature of a novel algorithm used in this program is discussed in Section 4.

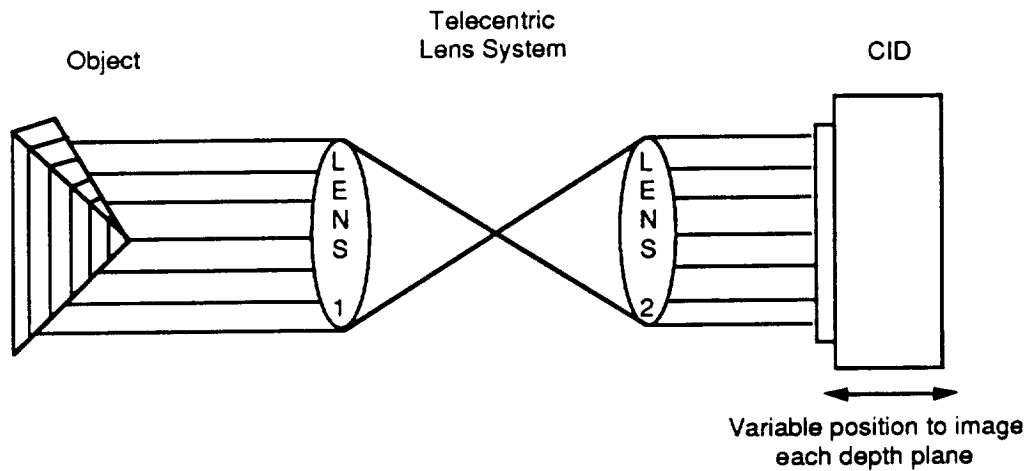


Figure 3-1. Image Acquisition Scheme

3.1 THE BASELINE SYSTEM: AFOCAL-TELECENTRIC IMAGING

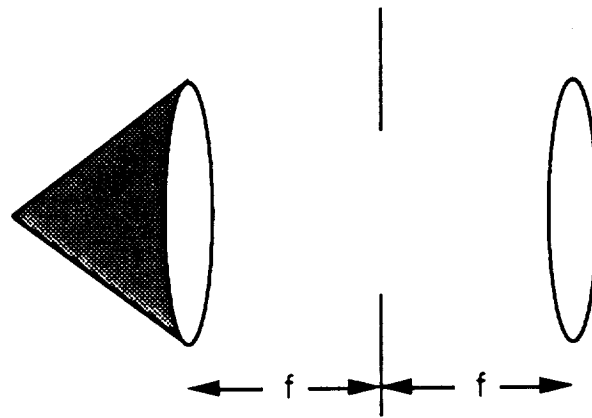
The baseline optical system consisted of a two-lens AT imaging system (Wetherell, 1987). This imaging system was proposed because of its especially attractive features for 3-D imaging. To begin with, transverse and longitudinal magnification are constant throughout the imaging volume within the operational area of the optical system. That is (at least in the absence of aberrations), no distortion is introduced other than a possible uniform compression or stretching of the image in the longitudinal and transverse directions. Moreover, and most importantly from the overall system standpoint, the imaging operation is space-invariant: each point of light in the image space is formed in the same way in three dimensions. This means that the DIPS three-dimensional deblurring operation can be applied in exactly the same way at each and every point in image space. This characteristic greatly facilitates restoration of the imagery.

During the Phase I effort, the feasibility assessment of the afocal-telecentric imaging design was based on an analysis made by Lohmann in connection with volumetric interconnections for optical computers (Lohmann, 1987). Lohmann's analysis concluded that the region of object space over which space-invariant imaging could be obtained with an AT system was in the shape of a cone whose base was at the entrance lens to the imaging system

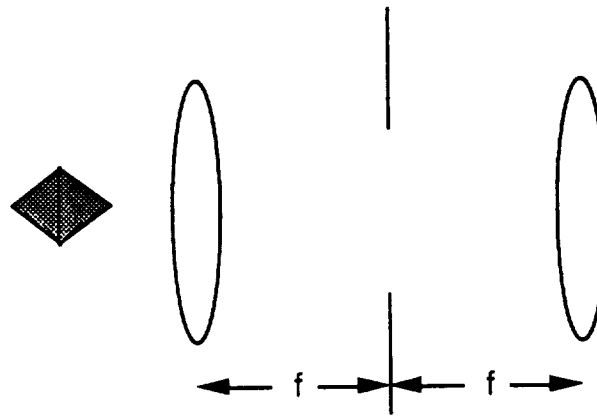
and whose apex extended out into object space. This is illustrated in Figure 3-2(a). The implication was that, at its largest extent, the space-invariant region had a diameter equal to that of the first lens. Lohmann's analysis suggested that high-numerical-aperture operation of the system was possible without reduction in the diameter of the space-invariant imaging region. An important Phase I conclusion was that good depth resolution could be achieved over relatively large-diameter objects.

In a subsequent detailed analysis of the space-invariant region (SIR) of afocal-telecentric imaging systems, Rhodes, Sitter, and Rothenheber (1989a, 1989b) found that in fact the SIR was much smaller than Lohmann had concluded, being given not by a single cone but by the intersection of two cones. This is illustrated in Figure 3.2(b). Lohmann's analysis had not accounted for how the image-space lens limits the SIR, but only for the effect of the object-space lens. In addition, the detailed analysis showed that to achieve space-invariant imaging over relatively large objects it was necessary that the system operate at a small numerical aperture. Associated with the small numerical aperture is an unacceptably large depth-of-focus, corresponding to poor longitudinal resolution for the image data base. As the depth of focus gets finer and finer, the SIR gets smaller and smaller, until ultimately it disappears.

Various AT lens configurations were evaluated to determine whether the size of the SIR could be increased without the use of much larger lenses. Unfortunately, it was determined that there is an unavoidable tradeoff between the size of the object to be imaged and the depth resolution achievable (cf., Iavecchia, Rhodes, Rothenheber, and Janiszewski, 1990a). This tradeoff must be addressed in the HERSS program because of the need to demagnify larger objects when imaging them onto the roughly half-inch-square-format conventional TV-type image detector arrays. Whereas transverse resolution scales linearly with object width, longitudinal resolution scales with the square of the object width. Thus, an object one-half inch on a side can be imaged with good resolution in both the transverse and longitudinal directions. A six-inch object, on the other hand, can be imaged only with significantly degraded depth resolution. Although transverse resolution is affected by only a factor of 12 (the required demagnification factor), longitudinal resolution is affected by a factor of 144.



(a)



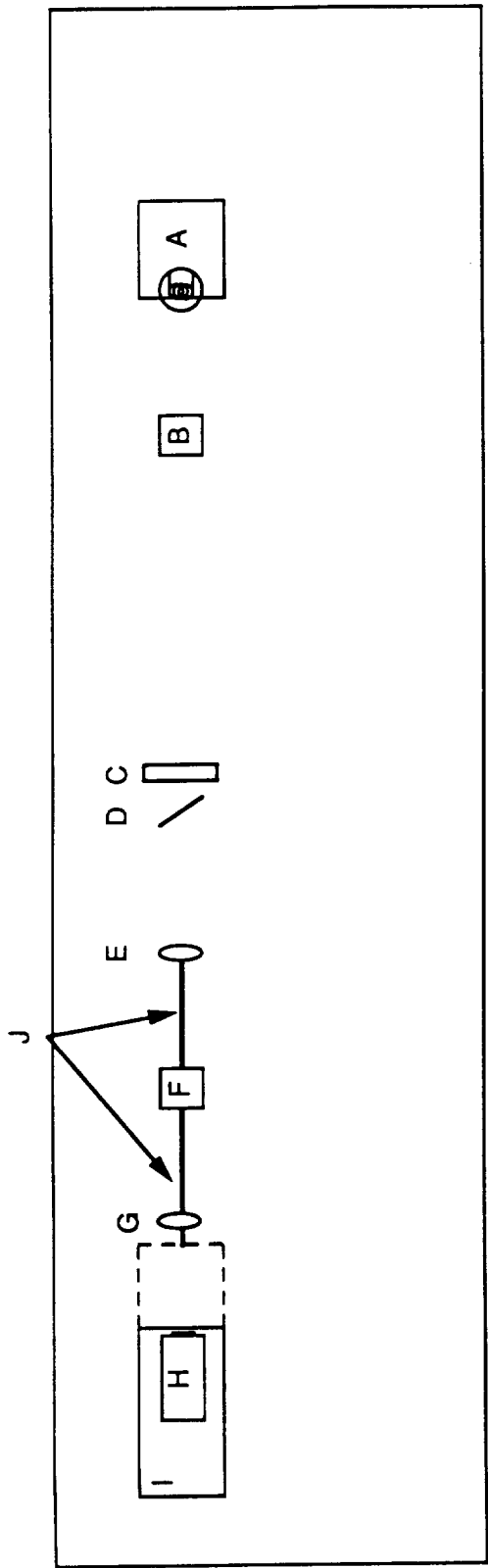
(b)

**Figure 3-2(a) and (b). Space-Invariant Region (shaded):
 (a) region obtained according to Lohmann's analysis;
 (b) much smaller region obtained by correct analysis. The
 distance f is the focal length of the lenses.**

Ultimately the limitations of the AT imaging system were overcome by using an imaging system of the non-AT type. Because of the space variance of non-AT systems in 3-D imaging, it was necessary to perform additional processing on the raw image data base. Both the nature of the non-AT systems and the additional processing performed to correct for the space variance of the imaging operation are discussed in later sections.

Despite its deficiencies, the AT system was still considered suitable for certain applications, and further analysis was performed to determine under what conditions an AT system might present a favorable tradeoff (cf., Iavecchia, et al., 1990b). Object size, depth resolution, and computational complexity were all considered. The probably the most serious limitation of the AT system is the relatively small SIR that can be imaged. Because practical AT systems must operate with demagnification factors near unity, the transverse width of the SIR in object space is limited to approximately one-half inch, the width of the TV detector array. If a large object is to be imaged, the full image can be built up in the transverse direction by a patchworking technique, in which the image is obtained in half-inch pieces. Because of the extensive data-collection operation required for the patchworking, the use of an AT configuration is effectively precluded for a display system that operates in near real. However, it is feasible that such a system could be employed by an autonomous robotic controller in cases where high-speed operation is not critical.

In the Phase II experimental program, work with the AT system focused on acquiring information over a half-inch square region with fine depth resolution. The AT data base was collected using the configuration illustrated in Figure 3-3. Two lenses, achromatic infinite-conjugate doublets with equal focal lengths (20 cm), were employed. The lenses were separated by a distance equalling twice their focal lengths. An aperture was placed in the common focal plane between the lenses, establishing the numerical aperture of the system in object and image space. Finer depth resolution over a reduced SIR was obtained with larger apertures.



- A - ILLUMINATION SOURCE
- B - 1/16 " PINHOLE
- C - DIFFUSER
- D - AIR FORCE TEST CHART
- E, G - ACHROMATIC DOUBLETS
- F - 1" APERTURE
- H - CIDTECH CAMERA
- I - TRANSLATION TABLE
- J - OPTICAL RAIL

Figure 3-3. IAS Baseline: Afocal-Telecentric Configuration

The test object for the AT experimentation was an Air Force resolution test chart in 35 mm slide format. This was placed in front of a diffuser, which was back illuminated by the beam from a slide projector. The illumination brightness was reduced to a level appropriate for operation of the CIDTECH TV camera detector array. The 35mm slide was oriented at a 45 degree angle to the optical axis so that the left portion of the slide was closer to the camera; the right side was more distant. As the camera was moved during image acquisition, each sequential depth plane contained a different narrow vertical slice of the test chart in clear focus. Other portions were out of focus. The resolution test chart was chosen as the input object because it contained fine structural detail. Figure 3-4 illustrates that portion of the test chart which was imaged. As discussed later, this aspect of the object was important to the performance of some of the DIPS algorithms.

A total of three-hundred video images were collected through 42.0 mm in image space at a stepping distance of 0.14 mm, the Nyquist distance for the system in the depth direction. Because of the 1:1 correspondence between object and image space in the unity-magnification AT design, information was therefore collected for 42 mm of object space. Resolution was also the same in both spaces. Transverse resolution, defined by the CIDTECH detector array pixel size, was 0.025 mm; longitudinal resolution, determined by the f/number of the system, was 0.1 mm. The transverse area of each image was approximately 10 mm by 10 mm as defined by the size of the TV camera detector array.

3.2 ALTERNATIVE: NON-AFOCAL-TELECENTRIC

The non-AT imaging system, which can consist of a single lens, is capable of imaging a much larger object than its AT counterpart. However, as stated above, the imaging operation is not space-invariant in three dimensions. The image is distorted geometrically because transverse and longitudinal magnification change in different ways as a function of depth plane. In contrast to the AT system, in the non-AT system light focuses in different ways to different points in image space. If the 3-D space variance is strong, it is necessary to subject the non-AT data base to algorithms that convert it to an AT data base by means of a pair of coordinate transformations (Rhodes, 1989). This point is discussed further in the DIPS section.

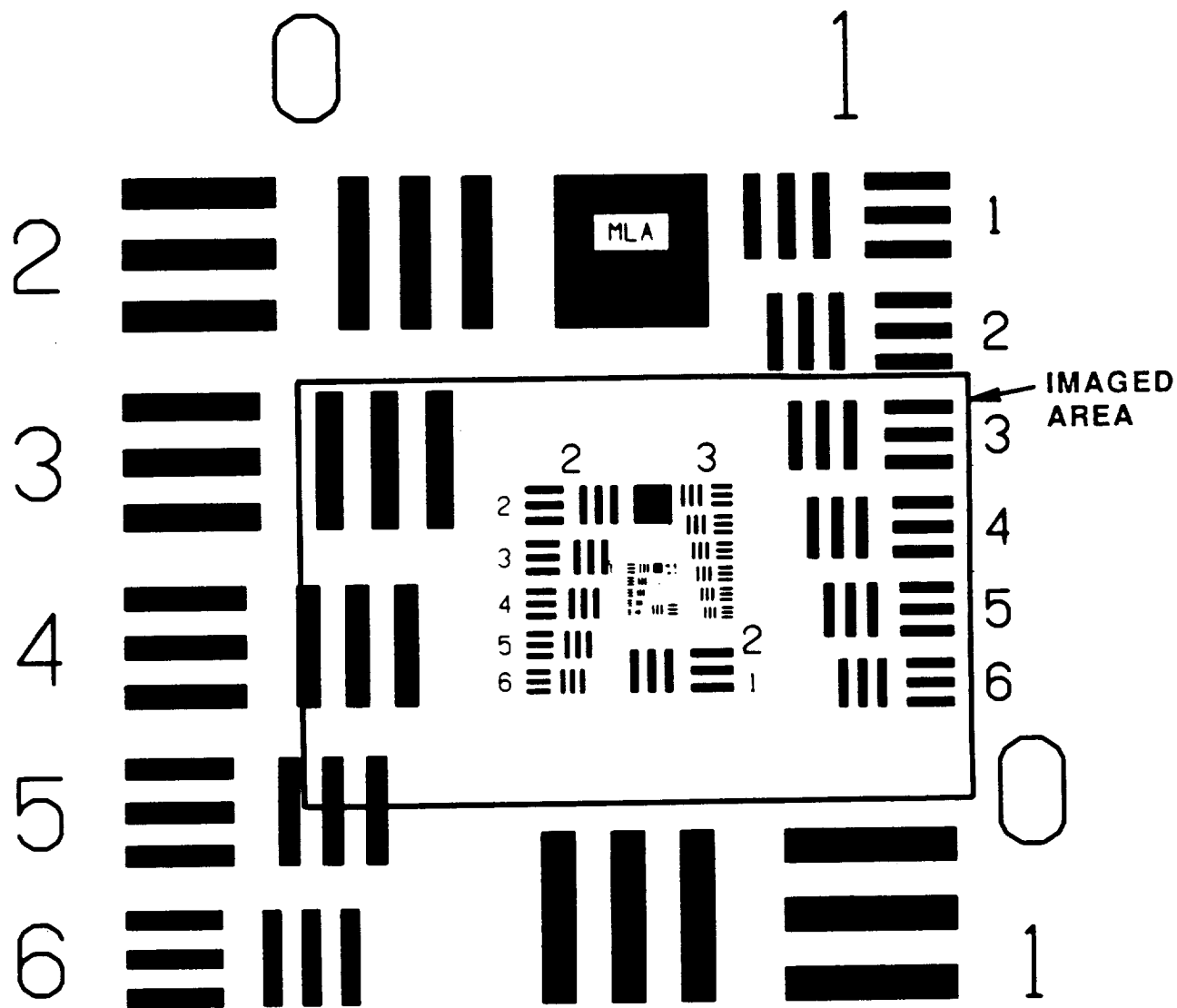


Figure 3-4. IAS Baseline: Test Object for AT Data Collection

The design of the non-AT imaging system was dictated by two primary considerations: The need to image a six-inch cube, and the need for the system to operate with an f /number of 4.0 or smaller in image space. Smaller f /numbers are more desirable, since they correspond to better depth resolution in both image and object space. However, diffraction-limited operation at numbers much below $f/4$ was deemed impractical because of prohibitive cost.

An $f/4$ MicroNikor lens was ultimately chosen as the imaging lens. This commercially-available lens, designed for use with a 35mm camera, is extremely well corrected over a wide range of magnifications and over a satisfactory field of view.

Pertinent parameters of the imaging system are as follows:

- Transverse magnification, $1/12$;
- Image space f /number, 4.0;
- Image space transverse resolution, 0.025 mm;
- Image space depth resolution, 0.1 mm;
- Object space f /number, 48;
- Object space transverse resolution, 0.3 mm;
- Object space depth resolution, 14.4 mm;
- Allowable object width, 152.4 mm;
- Lens focal length, 50 mm.

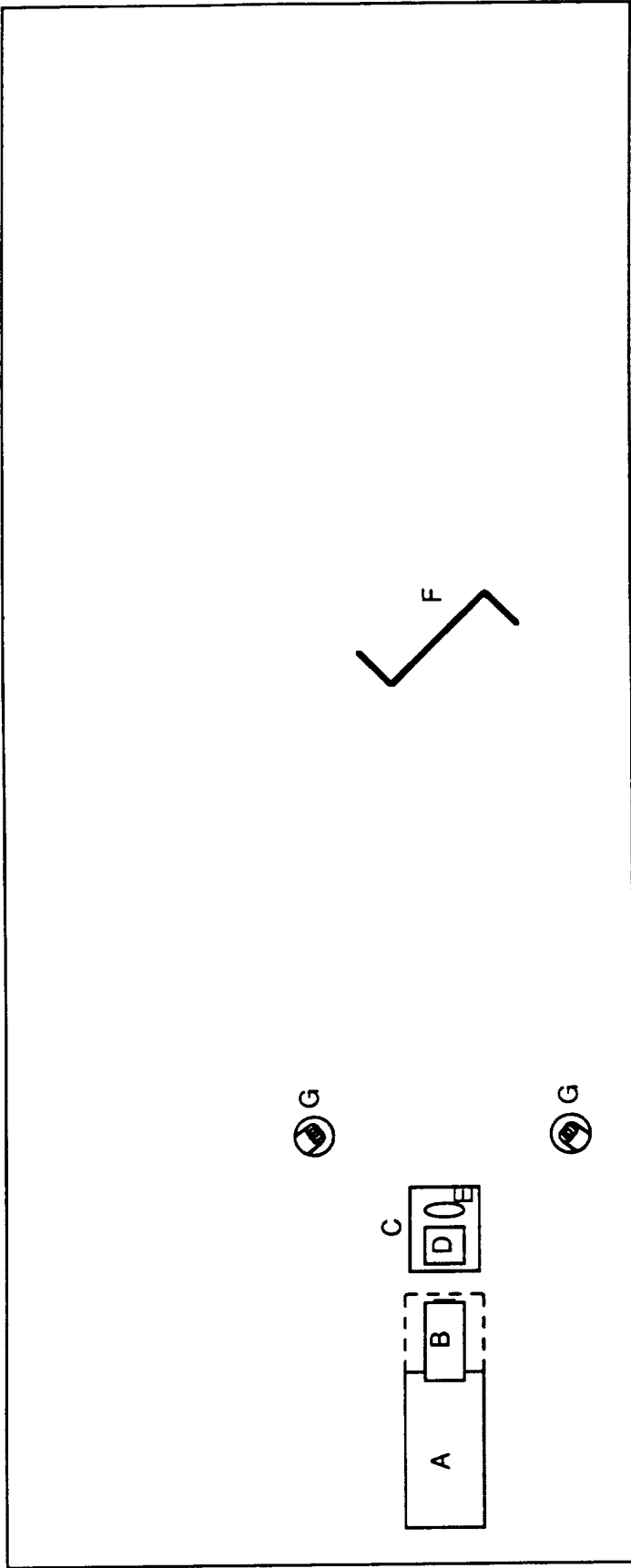
The resolution values given above are nominal and apply only to the midplane of the object and image space, respectively. For example, consider the nominal 14.4 mm object space depth resolution. In moving from the midplane of the six-inch-cube object space to the near plane in object space, depth resolution improves to 11 mm. In moving from the midplane to the far plane, depth resolution degrades to 18 mm. With a nominal image plane step distance corresponding to 14.4 mm in object space, only 10 to 11 slices are needed over the six-inch deep object to obtain the object brightness distribution.

The imaging system was set up so that a six-inch cube was imaged completely onto the TV camera. The non-AT optical configuration is shown in Figure 3-5. Since the locations of the principal planes and entrance and exit pupils for the MicroNikor lens were not available, step parameters may have been slightly in error. They were chosen, however, so as to assure that image-plane spacing never exceeded the Nyquist sampling distance. Transverse resolution was in all cases determined by the TV camera detector element size and did not present a problem so far as Nyquist intervals are concerned. For convenience, the mid-plane of the cube was positioned for 12:1 demagnification. Demagnification was greater or less than that for other object planes because of the image distortion introduced by the AT system.

The test object was a metal plate spray-painted with a white on black speckle pattern. As illustrated in Figure 3-6, various samples of text and speckle patterns were attached to the plate. The transverse resolution of the patterns was consistent with the resolution required to read a printed page and chosen to be near the transverse resolution limit of the imaging system. The plate was 12 inches wide but was bent at two locations creating a center flap 8 inches wide and two outer flaps 2 inches wide. The plate was oriented at a 45 degree angle during image acquisition. The plate was illuminated from the front with two incandescent bulbs.

A total of two-hundred images were collected through 4.0 mm in image space at a 0.02 mm stepping distance. This represented 2.8 mm of depth in object space. The transverse area of each image represented 140 mm in object space, or 5.5 inches. The image depth planes were spaced by a distance much smaller than the Nyquist interval. This is not necessary. However, finer sampling in the z direction would allow for testing of DIPS performance as a function of sample density.

Following data collection, the distortion-correction algorithm was executed to convert the acquired raw data base to an AT data base format. Only 20 image slices were used in this conversion to reduce the computational load. Subsequent DIPS image processing to acquire the shape function was applied to both the original uncorrected data base and to the 20 slices of the Sitter-corrected data base. Furthermore, to reduce the computational load, only the top left quadrant of the image data was converted.



- A - TRANSLATION TABLE
- B - CIDTECH CAMERA
- C - NIKON LENS ASSEMBLY
- D - NIKON SHUTTER
- E - MICRO-NICOR 55mm LENS
- F - OBJECT
- G - ILLUMINATION SOURCES (INCANDESCENT BULBS)

Figure 3-5. IAS Alternative: Non-Afocal Telecentric Configuration

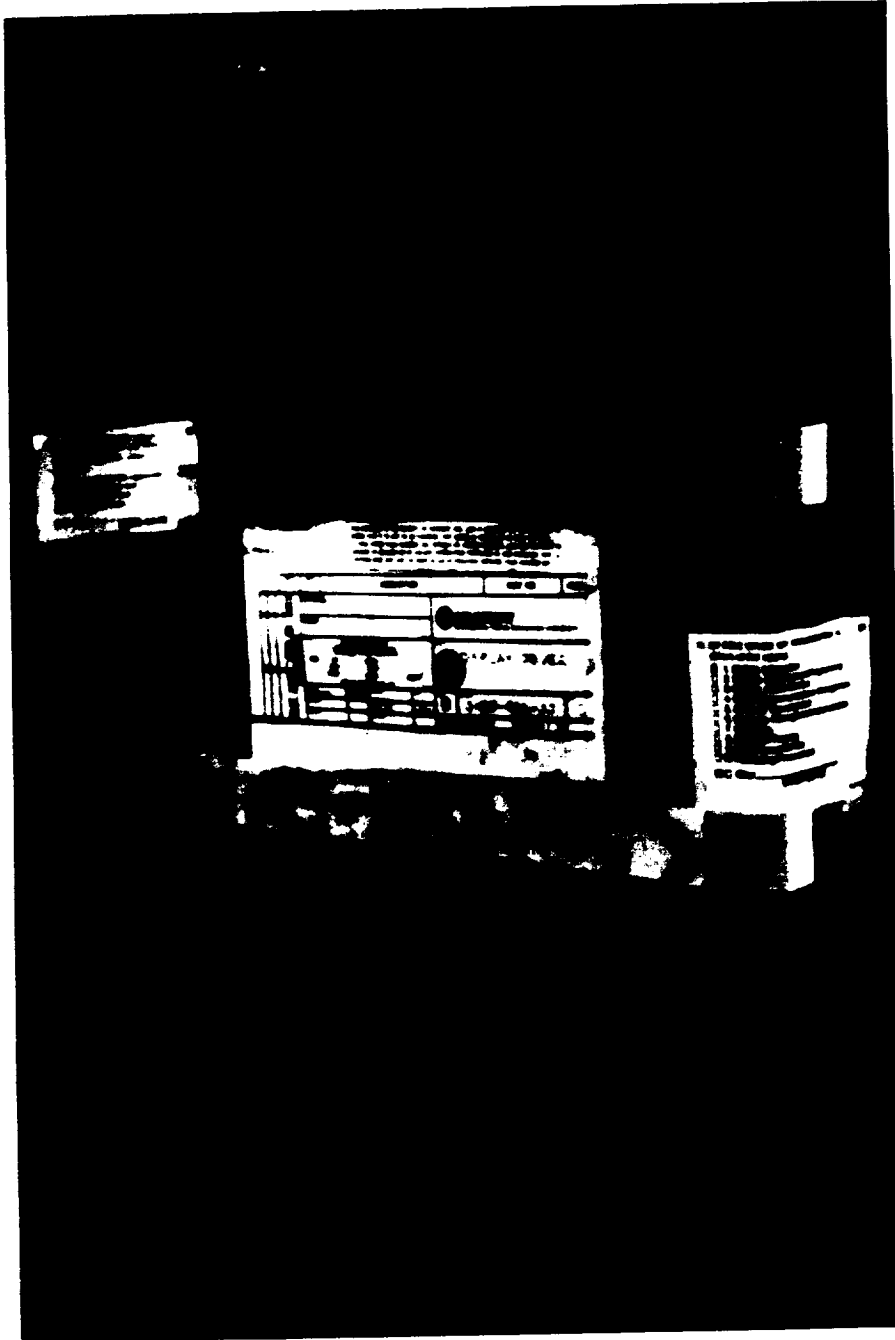


Figure 3-6. IAS Alternative: Test Object for Non-AT Data Collection

3.3 SUMMARY

The purpose of the IAS is collection of a 3-D image data base at the remote worksite using a customized optical configuration that operates with a single video camera (as opposed to a pair of cameras for stereo image pairs). Two-dimensional images are collected at small increments along the longitudinal depth axis in the 3-D image space. Each 2-D image, corresponding to a different depth plane of the object, contains both in-focus and out-of-focus information. The depth-plane images are processed by the DIPS to determine what in-focus information should remain in each 2-D slice. The DIPS algorithms use the known optical properties of the IAS to generate a numerical representation of the surfaces of the objects at the remote worksite.

The original IAS design scheme called for the use of an afocal-telecentric (AT) imaging system. The key advantage of an AT system is the shift invariance of the imaging operation: every point on the image of the object surface is imaged in exactly the same way, so long as the point was in the space-invariant region (SIR) of the imaging system. Furthermore, there is no distortion of image space with an AT system. The technical literature had reported that the SIR of an AT system formed a relatively large cone. However, during detailed design specification it was determined that the SIR region was in fact typically quite small. There was a critical tradeoff between the size of the SIR and the transverse and longitudinal resolution achievable with the system. A fine resolution could only be achieved over a very small area. For example, to achieve the desired 0.1 mm depth resolution, the SIR would only cover a 0.5 inch patch. The HERSS IAS design goal for 0.1 mm imaging could thus be met but only for a very limited object space area. In order to image a six inch cubic volume, a patchworking image acquisition technique can be implemented, but at the cost of both acquisition time and system complexity.

A non-AT system was subsequently proposed. This system can meet the IAS design goal and image a six-inch cubic area but cannot meet the depth resolution goal. With reasonably conventional optics, a six-inch cubic volume can be imaged at a depth resolution of about 15 mm. Because the non-AT system images are space variant, the non-AT system also requires use of a special correction algorithm (the Sitter correction algorithm) to transform the non-AT data base into an AT data base. This conversion is necessary because the DIPS

restoration algorithms assume an AT data base. Thus, a computationally-intensive correction algorithm must be applied to the data base prior to DIPS processing.

Despite the limitations of each of these IAS configurations, both an AT and a non-AT data base were collected and used as input to the DIPS restoration algorithms. An AT data base was collected of a one-half inch square patch of the Air Force resolution test chart at a fine depth resolution. A non-AT data base was collected of a metal plate with relatively crude depth resolution. The plate was spray painted with a white-on-black speckle pattern. It also contained various samples of text and speckle patterns. As discussed below, it was predicted that the DIPS restoration algorithms would work best when the speckle pattern was at the resolution limit of the imaging system.

4. DIGITAL IMAGE PROCESSING SYSTEM (DIPS)

The purpose of the Digital Image Processing System is to analyze the volumetric image data collected by the IAS and to determine the location and intensity of surfaces points within that volume. The DIPS algorithms result in two primary outputs, a shape function and a brightness function. Together these specify the object distribution. Each of these functions is specified by an x-y array, the length and width of which correspond to the dimensions of the CIDTECH detector array: 480 x 360. The shape-function array contains the z-value (depth plane number) for the corresponding x-y point on the object surface. The brightness array contains the corresponding surface brightness value.

Three alternative algorithms were tested for determining the surface and brightness functions:

- Frieden image restoration algorithm,
- Z-contrast algorithm,
- Häusler-based restoration.

Application of these algorithms required that the following assumptions be satisfied:

- Image data is sampled in the longitudinal and transverse directions at the Nyquist interval or better.
- The imaging system remains at a fixed distance from the object. (The imaging system is not moved in the longitudinal distance as the 3-D image-space is sampled. The only movement is of the detector array in image space).
- Each point on the object is defined by a position or shape function and a radiance or brightness function.
- The 3-D imaging operation is shift invariant.

The last assumption has two critical implications:

- 1) All points on the object surface are seen by the entrance pupil of the imaging system. There can be no hidden points.
- 2) The imaging system is both afocal and telecentric. If a non-AT system is used, the raw IAS data must be converted to AT format.

Two primary IAS data bases were collected and used as input to the DIPS algorithms. One data base was collected using an AT system with an Air Force test pattern. Figure 3-4 shows the test pattern, highlighting the portion that was actually imaged. The second data base was collected using a non-AT configuration and the metal plate described in the IAS section. The metal plate is illustrated in Figure 3-6. The raw IAS data bases were used to evaluate algorithm performance. In addition, for the Frieden algorithm, simulated data bases were also in the evaluation.

As noted earlier, afocal-telecentric imaging systems were chosen for baseline image acquisition work because of a key feature unique to them: they image all points of the object in precisely the same way. In mathematical terms this means that the imaging operation can be described by a 3-D convolution. This characteristic of AT imaging systems greatly simplifies the image processing operation that must be performed by the DIPS subsystems. Non-AT systems produce distorted images in three dimensions, different points on the object being imaged in different ways. If the DIPS algorithms were to take the space variance into account, the computational load would be enormous. To avoid such complexity, an algorithm was recently developed to convert a non-AT data base to an AT data base before being input to a DIPS algorithm.

Section 4-1 describes a preprocessing algorithm for converting a non-AT data base to an AT format. Sections 4-2 through 4-4 present research efforts for the three alternative DIPS algorithms — Frieden, z-contrast, and Häusler. The overall performance of the algorithms is evaluated in Section 4-5. An alternative technique for acquiring the numerical data base of the remote object surfaces is presented in Section 4-6. A summary of the research results is provided in Section 4-7.

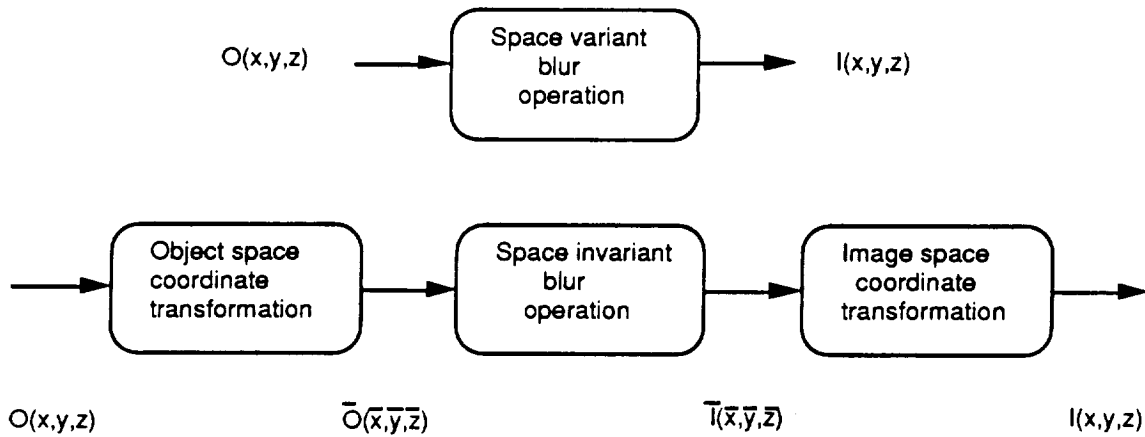


Figure 4-1. Sitter Coordinate Transformation Model for non-AT Data Bases

4.1 DIPS PREPROCESSING FOR NON-AT DATA BASES

Recently, Sitter developed a simplified method for processing non-AT 3-D imagery to convert it to an AT format (Sitter and Rhodes, 1990). Sitter took the integral expression representing the 3-D imaging operation and rewrote it in terms of redefined object- and image-space coordinates. The effect, illustrated in Figure 4-1, is to replace a general shift-variant superposition integral with a shift-invariant one, i.e., with a 3-D convolution integral. The convolution operation is bracketed by object- and image-space coordinate transformations of the form

$$x' = \frac{x}{d + z} \quad (1)$$

$$y' = \frac{y}{d + z} \quad (2)$$

$$z' = \frac{z}{d + z} \quad (3)$$

where d is a parameter of the imaging system. If the Sitter coordinate transformations are applied correctly to the raw non-AT 3-D image data base, the result is a data base of the same form as one acquired by an AT imaging system. The shape-and brightness-finding algorithms can then be applied just as though the data were acquired with an AT imaging system.

The coordinate transformations affect both the transverse coordinates x and y and the longitudinal coordinate z . The longitudinal coordinate transformation can be affected either by changing the spacing between image sample planes (slices) in accord with Eq. (3), or by appropriate numerical interpolation in the longitudinal direction. The necessary transverse scaling was effected by means of a simple linear interpolation algorithm applied to the raw data base. To make the linear interpolation results more accurate, the raw image sample data (which was taken at a rate only slightly above the minimum Nyquist sampling rate) is preprocessing using a 4x sinc function interpolation.

4.2 DIPS ALTERNATIVE: FRIEDEN

The Frieden image restoration technique uses the theory of the 3-D optical transfer function to determine the 3-D object from measured image-space light distribution. The method is similar in many regards to a deconvolution: The effect of the imaging system is modeled mathematically, and the measured image-space data is, in effect, moved back through that model. The result is the original object space distribution, to within the limits allowed by the information actually passed by the imaging system. (Some information is irretrievably lost, e.g., because of the finite resolution of the imaging system.)

Frieden's algorithm requires that several additional assumptions be satisfied beyond those listed earlier:

- The imaging operation is diffraction-limited.
- The image data is related to the object shape and brightness functions through the 3-D optical transfer function associated with the imaging system.

The Frieden algorithm first finds an estimate of the object brightness function. This is done by taking the sum (projection) of all the depth-plane measurements in the z -direction and subjecting the resultant distribution to a deblurring operation. The deblurring is performed in the spatial frequency domain by application of a pseudo-inverse filter to the projection data. The surface brightness function estimate is then used in an elegant algorithm, based on a Fourier-transform-based description of the image distribution, to obtain an estimate of the object shape function (cf., Frieden, 1988a; 1988b).

The Frieden algorithm was tested in four phases:

- Simulation testing at the University of Arizona on a CDC Cyber 175. The simulated image data base had dimensions of 32 x 32 x 32. A 32-bit word length was used.
- Simulation testing at the University of Arizona on a CDC Cyber 175 using 64 x 64 x 20 simulated data bases (32-bit words).
- Simulation testing at Analytics on the Macintosh IIX using 32 x 32 x 32 simulated data bases (8-bit words).
- Testing at Analytics on the Macintosh IIX using the AT experimental data base collected with the AF resolution test chart as test object.

In the early simulation testing, image test data were generated using a software program developed by Frieden (1988c). This program simulated a 2-D disk and a 2-D ring located within a 32 x 32 x 32 image volume. The disc and the ring were located in unique depth planes. The on-axis was located in plane position 20 and an on-axis ring was located in plane position 23. During this early work, it was determined that the original Frieden technique for estimating the shape function was not performing as expected. Although Frieden's method for estimating the shape function worked well for some cases, in other cases it was unable to determine the object shape distribution with satisfactory accuracy. For example, although the z-position of the ring was well estimated, the central region of the disc was not found. Only the rim of the disc was well estimated.

Numerous attempts were made to determine why the shape algorithm was failing to perform better, including the following:

- Rechecking the theoretical development and programming of the theory,
- Tapering the image endpoints with a raised cosine Hanning filter to reduce possible aliasing,
- Equalizing the planes of energy by a simple scale-factor multiplication of each plane, and
- Varying specific algorithm parameters in the event that the original values were too fine or too coarse for successful numerical differentiation.

None of these attempts was successful.

Two major characteristics of Frieden's algorithm are thought to have contributed to its poor performance. First, the algorithm is based on the numerical calculation of the derivative of the estimated object spectrum making it quite sensitive to noise in the measured image plane values. Secondly, the algorithm requires knowledge of the prior-estimated object brightness function, which itself contains noise.

In an attempt to overcome these limitations, a simpler algorithm was developed for finding the shape function. The approach scans the brightness values along the z-axis for each x,y transverse position. The plane where the maximum brightness is found is taken to be the depth location associated with the surface point. For the 32 x 32 x 32 data base described above, the method was successful at locating the disc in plane position 20 and the ring in plane position 23.

Further testing at the University of Arizona focused on determining the effect of noise on algorithm performance. A more complicated data base was generated on the Cyber 175 using a program named PIECE.TOG (Frieden, 1989). This program generates a data base 64 x 64 x 32 which can contain several disc and rings that are not on axis. A data base was created with a central disc that was placed at plane 6. Four rings were placed at locations surrounding the disc in planes 4, 5, 7, and 8. Without noise added to the data, the local maximum shape function algorithm was able to predict the locations of the objects rather well. When 5% noise was added, the errors were more numerous. With 10% noise, the algorithm performance was considerably degraded.

The source code developed at the University of Arizona was used at the Analytics laboratory facility in the next development phase. The code was first converted from Cyber Fortran 77 to Macintosh Fortran 77. A smaller (8-bit) word length was utilized to more realistically simulate the gray scale resolution achieved with a standard video camera. The results were much noisier than those achieved with the Cyber-run test cases. The difference is attributed to the reduction of precision in the data base.

The AT data base of the AF resolution test chart collected with the IAS was also input to the revised Frieden shape algorithm. To assist in analysis of the quality of the results, the numerical values corresponding to predicted depth values were converted to eight-bit numbers for gray-scale or pseudo-color display. Values equal to zero were white in the gray-scale displays and represented image data points closest to the camera. Values at 256 were black and represented image data points farthest from the camera. The gray-scale display of the tilted Air Force resolution test target--a planar object placed at an angle with respect to the optical axis--should thus be uniform in the vertical direction but get progressively darker in moving from one vertical edge to the other.

The shape function obtained provided only a crude estimate of the location of the surface points in the depth planes. The bars of the test chart varied in depth from the left to the right portion of the image. This would be expected due to the 45 degree tilt of the 35 mm slide containing the test chart. However, the background was more accurately located in regions where object information (i.e., a vertical or horizontal bar) was located. In regions where the background was relatively distant from object information, the algorithm rather arbitrarily assigned the depth of the background to be either in the front or back plane.

The modified shape-finding algorithm can only be expected to work on isolated bright objects in what is otherwise a void. These are just the conditions of the original University of Arizona simulation testing. For typical real-world cases, however, the background is not a void, and the object does not consist strictly of bright pieces in relative isolation. In such cases, the modified Frieden shape-finding algorithm cannot be expected to perform well. The tilted test target represents an intermediate case between the simulation test case and a typical real-world scene.

4.3 DIPS ALTERNATIVE: Z-CONTRAST

Improved shape function estimation was achieved by a method referred to as the z-contrast method. As described below, this method is based on a measure of local image contrast.

The z-contrast method for determining the object shape function is based on a local image contrast criterion. In this scheme the contrast in a small, local (e.g., 3x3-pixel), detail-containing region of the image irradiance distribution in a given plane is measured. If that small region is in focus, the contrast will be maximum. Any degree of defocusing will reduce the contrast measured for that small local region. The contrast was determined by measuring the variance over the 3x3-pixel patch. The brightness at that point defined the image surface brightness distribution. The center point of the 3x3 patch was assigned the z-value of the plane in which contrast was maximum. A logic flow diagram for the z-contrast algorithm is presented in the auxiliary software guide (Janiszewski, Iavecchia, and Mathur, 1990).

The z-contrast method can be expected to perform best on regions of the object that contain high contrast structural detail at the resolution limit of the imaging system. Just as a human operator of a microscope needs structural details on which to focus, so too does the contrast-maximization computer algorithm. In addition, there are ways in which the z-contrast algorithm can be fooled into selecting the incorrect depth plane. Assume, for example, that a 3x3-pixel patch of the object is uniformly bright but is surrounded by dark pixels. As the detector array moves through focus, the contrast over that central 3x3 patch can actually fall from finite values to zero. Because the variance goes to zero, the algorithm will as a consequence yield the incorrect depth plane. In order to avoid this problem, it may be necessary in some cases to supply structural detail (e.g., by splatter-painting) to object surfaces otherwise devoid of resolution-level patterns.

The Z-contrast algorithm was applied to three IAS data bases:

- AT data base for the AF resolution test chart
- Non-AT data base of the metal baffle with 20 Sitter-corrected image slices.
- Non-AT data base of the metal baffle with the 200 non-Sitter-corrected image slices.

It was first run on a data base that was collected with the afocal telecentric imaging system. The test object was, as noted earlier, a portion of an Air Force resolution test chart placed at approximately a 45 degree angle to the optical axis. The portion of the test chart that was used contained dark bars on a bright background. The bars were oriented both vertically and horizontally and were of different widths and lengths.

When the derived shape function was displayed as a digital gray-scale image, as shown in Figure 4-2(a), it was apparent that the algorithm was not estimating the correct shape function over all portions of the object. Ideally, the gray-scale representation would exhibit a linear progression in gray value from the left to the right edge. This would corresponded to a planar object placed at an angle with respect to the optical axis. However, as is clear from the figure, those portions of the image corresponding to the background in the test object appeared to be randomly located throughout the entire volume. This kind of behavior is particularly evident in the cross-sectional scan, shown in Figure 4-3. There is a clear average slope to the scan values, corresponding to the tilt of the planar test object. However, there is also considerable noise in the depth positions determined by the algorithm.

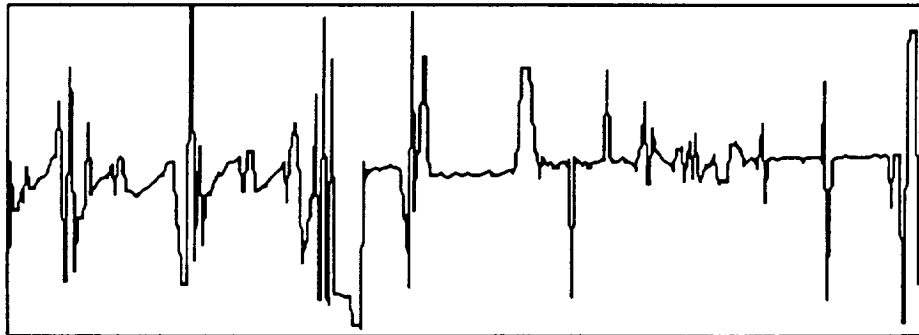
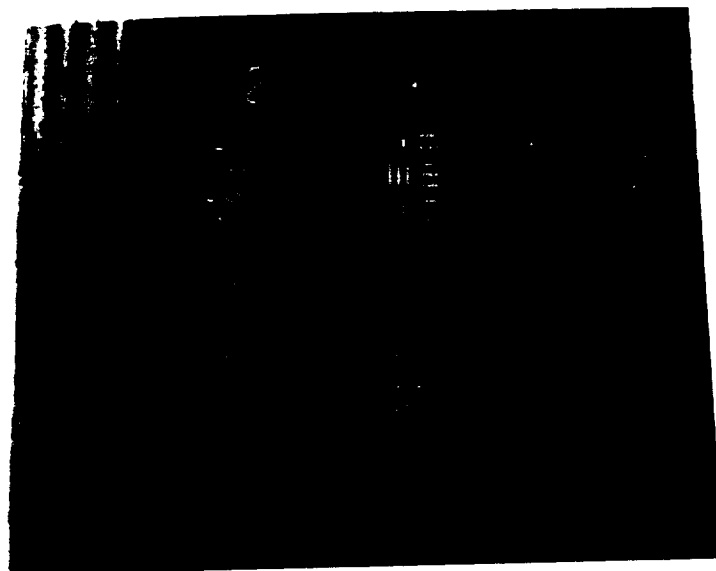


Figure 4-3. Cross-section of Shape-Function Gray-Scale Image



(a)



(b)

Figure 4-2. Results of Z-Contrast Algorithm Processing of the AT Data Base:
(A) Shape Function, (B) Brightness Function

The most likely explanation of this behavior is that the measured contrast in the background regions is so small that the noise generated by the imaging system predominates. The algorithm performed more accurately on sections of the test object that contained bars and edges, though even in those regions the depth function was sometimes incorrectly assigned. Specifically, the regions that contained the horizontal bars were located in approximately the correct planes, but invariably the algorithm located the upper half of a bar in one plane and the bottom half of the bar in a different plane. Additionally, the algorithm located the center line of each bar to be in a different plane than either the upper or lower half of the bar and generally the center line was placed in a plane that was significantly farther away than either of the other two sections. No explanation for this behavior of the algorithm has been found.

The brightness function found by the z-contrast algorithm function, shown in Figure 4-2(b), has much the expected appearance. However, because the surface function is incorrectly assigned as certain x-y points in the image, the brightness values are also incorrectly assigned at those points. The brightness function thus has a rather noisy appearance.

In an effort to optimize the performance of the z-contrast method, the algorithm was applied to the same data base using a 5x5 kernel in place of the 3x3 kernel. Results obtained were comparable, though a little noisier. This is reasonable considering the constant tilt of the object. Because the focus varies more over the 5x5 patch than over a 3x3 patch, contrast will be less and, hence, the determination of maximum contrast will be more subject to the influence of noise.

Results obtained with the non-AT data base were consistent with those obtained with the AT data base. Figure 4-4 shows the results obtained using a 3x3 kernel and no Sitter-algorithm correction to the data base. The shape algorithm performed best for those regions of the object containing structural detail at the resolution limit of the imaging system. This included regions of text as well as portions of the speckle pattern. The brightness distribution was blurred for some regions of the test pattern. Performance was good for only those regions where the test structure was close to the resolution limit of the imaging system.

ORIGINAL COPY IS
OF POOR QUALITY



(a)



(b)

Figure 4-4. Results of Z-Contrast Algorithm Processing of the Non-AT Data Base (No Sitter Correction): (a) Shape Function, (b) Brightness Function.

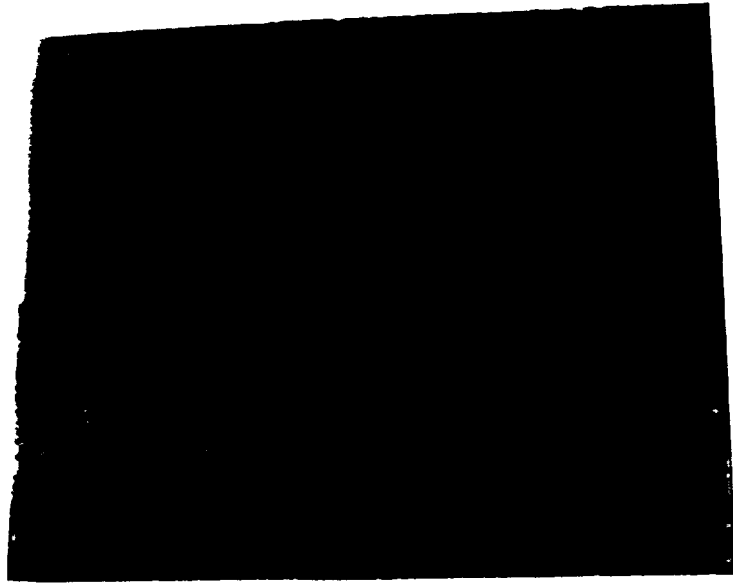
The z-contrast algorithm was also applied to the distortion-corrected data base output produced by the Sitter algorithm. Results are shown in Figure 4-5. By correcting for space variance in the 3-D image data base, the Sitter algorithm yielded slightly improved results, though the differences are minor. More significant differences would be expected if a more strongly space variant data base were used as input. (Because the demagnification factor was so large for the non-AT imaging operation, the effects of the Sitter correction algorithm were, in fact, not very great).

4.4 DIPS ALTERNATIVE: HÄUSLER'S ALGORITHM

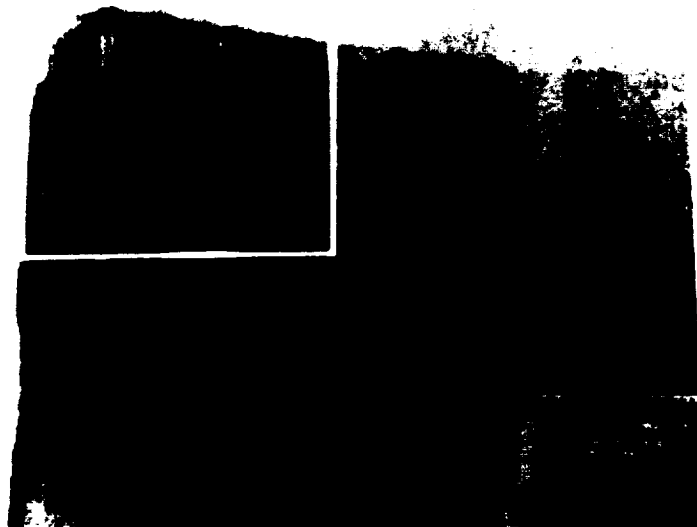
The z-contrast method fails if the patch over which the contrast is measured has uniform brightness. When this is the case and when the patch is in focus, the contrast is actually zero. An alternative method, developed by Häusler for establishing the best focus for planar objects, will often work better than z-contrast for such objects.

Häusler's algorithm (Häusler and Korner, 1984) is based on the following observation: if an image is in focus at a given detector element, the output of that element (voltage or current) is usually at a local extremum--i.e., a local maximum or a local minimum. (This will always be the case with binary, or dark-bright objects. In that case when the object is in best focus, the darks are darkest, the brights brightest. With gray-scale objects, cases arise when the best focus does not produce an extremum. Even in those cases, however, the slope of pixel intensity, as a function of focusing position, should be zero.) The best focus for a planar surface can thus be found by monitoring the outputs from a number of pixel detector elements and looking for a condition when they all simultaneously reach a local extremum.

As developed for the DIPS subsystem, Häusler's algorithm tests whether the output of a given detector element changes as z is changed by one step. If so, it is judged that that element cannot be in focus. In the DIPS implementation, a 3x3 pixel patch is deemed to be in focus if none of the elements in that patch is judged to be out of focus. Some allowance must be made for noise in the measurements, e.g., in the form of a threshold slope value above which the detector element is taken to be out of focus.



(a)



(b)

Figure 4-5. Results of Z-Contrast Algorithm Processing of the Non-AT Data Base (With Sitter Correction): (a) Shape Function, (b) Brightness Function.

Häusler's algorithm can be applied successfully to a wide variety of textured objects that are planar and oriented normal to the z axis. For non-planar objects, however, and even for planar objects that are tilted out of the x - y plane, it will often fail to work, since under such conditions simultaneous slope zeros will not normally be measured even by detector elements in adjacent resolution cells. Despite this limitation, it was thought appropriate to test the Häusler algorithm because of its potential ability to determine in-focus conditions when the z -contrast method is certain to fail, e.g., when the in-focus 3×3 patch of concern has uniform brightness (zero contrast).

Häusler's algorithm was coded in Fortran and tested on the AT Air Force test target data base. Although it is not certain why, the algorithm produced an output shape function given by $z = 0$. In other words, the estimated shape, instead of corresponding to a tilted planar surface, was given as a flat surface at the extreme end of the longitudinal scan range. The reason for this result is not completely clear. However, as suggested in the preceding paragraph, the algorithm is not expected to work well on a tilted surface. In the absence of noise, and with a tilted surface containing considerable detail, the results obtained are probably correct: the in-focus condition will never be satisfied over the 3×3 -pixel patch, and ultimately the shape-function array will be filled with the z -value (image slice number) of the extreme back or front of the scanned volume.

4.5 PERFORMANCE OF THE IMAGE PROCESSING ALGORITHMS

As discussed earlier, only the z -contrast algorithm performed even moderately well, and it is subject to being fooled by objects with inherent low contrast at the resolution limit of the imaging system. The best object, so far as the z -contrast algorithm is concerned, has a surface that is rich in high-contrast, high-resolution structural detail. Patterns like those obtained with spatter paint might work well, as would random patterns similar in appearance to laser speckle. Such texture patterns could be applied to cooperative objects of interest to NASA in connection with telerobotics operations.

In some cases a hybrid algorithm combining, for example, the z -contrast method with Häusler's method, might do a better job of determining the location of an object surface. When one method fails, it is often the case that the other will work, and vice versa. However, even a

combined method is not expected to work satisfactorily for most NASA telerobotics applications. Ultimately, if algorithms of this kind are to perform adequately, they must be combined with some higher-level intelligence in the processing software. Consider, for example, the case of determining the location of a smooth, featureless wall. With passive distance-finding methods of the kind considered in this program, it is impossible to determine its distance because there are no features on the surface. The distance could be inferred from the locations of the edges and corners--something that requires processing at a higher level.

Active distance-finding methods (e.g., laser range-finding or laser-triangulation methods) are not subject to the limitations of the passive methods considered in this project, and, as noted below, they may in fact represent the best possible solution to the problem of 3-D data base acquisition.

4.6 ALTERNATIVE IAS/DIPS — LASER RANGE SCANNERS

Many range finding technologies have emerged recently that are suitable for use as alternatives to the IAS / DIPS system originally proposed in 1986. These technologies include laser radar, triangulation, holographic interferometry, and others. Although many of these alternatives could be adapted for use in the HERSS project, triangulation is a clear solution because of cost effectiveness and resolution advantages. This technology became commercially available in 1987 (after the original IAS/DIPS system was proposed) and therefore was not previously considered.

Commercially-available laser range finders employ triangulation techniques to determine object surface points. A laser projects a point of light onto the object at a particular angle to the x-axis. That point of light is reflected from the object and is imaged onto a CCD camera. The output of the CCD, combined with the current projection angle, is then used to determine (x,y,z) points on the object surface. At some cost to acquisition time, most scanners use linear CCD arrays and thus only operate on a single line of points at a time. Thus, for linear arrays to obtain a full (x,y,z) data base of an object, it is necessary to traverse the scanner's line of sight across the object. This can be done by the use of either a positioning table or a scanning mirror. Some commercially available systems include the necessary positioners to acquire a full (x,y,z) representation of the object. While area CCD arrays could

circumvent the time cost associated with linear CCD data acquisition, area CCDs have their own limitations of reduced resolution.

A number of currently available range finders were surveyed to determine their capabilities and applicability to the NASA application. Key factors that were considered included:

- **Standoff Distance** - closest distance between the object and the imaging system.
- **Z Resolution** - smallest distance the system can resolve in z.
- **Lateral Resolution** - smallest distance the system can resolve in x or y.
- **Depth of field** - total z space the system can measure.
- **Field of view** - the x or the x-by-y area that the scanner can image (according to whether the unit is a line or area scanner).
- **Positioners required** - indicates what types of positioners would be required to scan a full 6"x6"x6" object.
- **Interface** - how the camera is interfaced to the main computer system.
- **Acquisition time** - the manufacturers rating for how many points of data can be acquired in a second. Note that this time does not include any additional time necessary to move the camera via positioners.
- **Physical dimensions** - Actual dimensions of the scanner itself, not including any external hardware necessary to interface the camera to the computer system.

Table 4-1 presents the results of the survey conducted across five commercially-available range finders.

The most critical factors for the NASA telerobotic application include standoff distance, resolution, and speed. Considering these requirements, Servo Robot's Jupiter Camera (1987) was found to be the optimum candidate. A key advantage of the Jupiter system over the other alternatives is that it has a 1005 mm depth of field and a depth resolution of 1 mm over a six-inch cubic area. Because the Jupiter is a line scanner, collecting a data base requires the use of either a one-axis positioning table or a scanning mirror. Other systems,

Table 4-1. Laser Range Scanner Survey

	Synthetic Vision Systems	Servo Robot	HyMarc	Selcom	Cyber Optics
Unit	SMT Inspection System	Jupiter Camera	HyScan	S001 Laser Track Sensor	LRS 400-1200
Standoff	62.5mm	165mm	150mm	190mm	30mm
Z Resolution/Accuracy	0.000625mm	0.029mm	0.025mm	0.02mm	0.01mm
Lateral Resolution/Accuracy	0.0025mm	0.15mm	0.025mm	0.5mm	0.03mm
Z Depth of field	6.25mm	1005mm	100mm	72mm	4mm
Field of view	6400mmx6425mm	108mm - 579mm (line)	100mm	50mm	2mm
Positioners required	None	One positioner required	None	One positioner required	One positioner required
Computer Interface	Full Workstation	Std. Controller	Full Workstation	Std. Controller	IBM PC Card (only)
Acquisition time	700,000 pts/sec	28,000 pts/sec	10,000 pts/sec	5,000 pts/sec	500 pts/sec
Dimensions	150x150x200mm	114x179x95mm	260x112x70mm	330x113x50mm	150x160x45mm
Price	\$130,000	\$30,000	\$100,000	\$28,000	\$10,000

however, require the use of a two-axis positioner which is a less desirable option for NASA. That is, minimizing scanner movement is an important goal in a remote object-space where the area may be cluttered with various objects and/or space debris. Furthermore, the Jupiter system can acquire a 6"x6"x6" data base with a resolution of 1 mm in approximately 36 seconds. If a 2 mm resolution is adequate, the system can acquire the data in only 18 seconds.

To demonstrate the ease of integrating a laser scanner data base with the HGS subsystem, two laser scanner data bases were acquired from Servo-Robot and integrated with the HERSS software. One data base contained a numerical representation of a pair of pliers resting on a table; the other was a frontal scan of a sculpted face. The resolution of each data base was 0.3 mm in x and 0.1 mm in y and z. The data was converted to the standard HERSS data base file format for use by the HGS system and a hologram of the pliers data base was successfully recorded.

4.7 SUMMARY

The z-contrast, Frieden, and Häusler restoration algorithms were used to process the AT data base of the AF test chart. As speculated, the z-contrast maximization technique was better at predicting the location (i.e., shape) of the surface for those regions of the test chart that contained the finest detail. Performance was poor for areas with little or no fine structure because the algorithm could not find the small variations in contrast required for successful operation. Thus, a necessary criterion is that the patch must contain detail (e.g., white and black speckles). A patch with uniform brightness (e.g., a white wall) will not be successfully located because there are no intensity differences within the patch and hence no contrast variations. Cooperative objects can be prepared for satisfactory imaging and processing by the z-contrast method by providing them with high-contrast, high-resolution detail, such as splatter-painted patterns.

In an effort to determine the optimum performance of the z-contrast algorithm, the patch size was varied from a 3 x 3 kernel to a 5 x 5 kernel. Performance was somewhat a little worse using the larger kernel. The degradation in performance is attributed to the tilt of the object through depth because the 5 x 5 kernel size extended over more depth slices than the 3 x 3 kernel.

The Frieden algorithm was less accurate than the z-contrast algorithm in the prediction of surface point locations. For each x,y transverse position, the Frieden shape function searches for a maximum intensity value along the z- axis. The plane where the maximum value is found is then assumed to be the z location for that x,y position. This technique was developed for bright objects against a dark, distant background similar to the test case objects used during the early simulation studies of the algorithm.

The results using Häusler's surface-finding technique were poor. The algorithm was not able to find any planes to be in focus, and the resultant shape function contained all zeros. It is believed that this behavior of the algorithm is related to the tilt of the test object. The algorithm was really designed to locate planar objects that are normal to the camera axis. Häusler's algorithm examines a 3 x 3 patch at each z-depth plane. When every element of the patch contains an extremum (minimum or maximum) the patch is assumed to be in best focus. However, if the patch is oriented at a 45 degree angle, part of the patch may be in focus in one depth plane and another part of the patch in another depth plane. If this is the case, it is not possible to meet the criteria for common extreme in every patch position.

For the non-AT data base obtained with the bent planar test object, the Sitter algorithm was used to convert the data from a non-AT to an AT format. The Sitter processing algorithm performed as expected, though because of the large demagnification factor of the IAS configuration, the difference between the Sitter-corrected data base and the original data base was fairly minor.

When the z-contrast algorithm was applied to the non-AT data base, the brightness distribution was blurred. This result is consistent with expectations because much of the test structure was close to the resolution limit of the non-AT imaging system. Performance of the shape-finding algorithm depended on the structural scale of the noise patterns present in the test object. Those portions where structural details closely matched the resolution limit of the imaging system yielded the best performance.

Considerable research is necessary to determine how object characteristics (e.g., shape, surface structural detail, orientation, illumination) affect algorithm performance. Ultimately it is thought that an algorithm could be developed to restore an object that had very specific characteristics. However, it is unlikely that a generic algorithm can be developed that would apply to all object types without the incorporation of some higher-level intelligence.

The relatively crude performance of the DIPS algorithms, as well as the limitation of the IAS to scan a six inch cubic area with high depth resolution without a patchworking technique (cf., Iavecchia, Rhodes, Rothenheber, and Janiszewski, 1990), was the impetus to consider other current technologies for collecting the numeric data base of the remote objects. Developments in technology indicate that laser scanning devices can acquire an image data base with a depth resolution significantly less than 1 mm. Laser scanning devices are not generally dependent on specific object characteristics as are the DIPS image restoration algorithms. Therefore, the most practical recommendation for collection of a numerical data base for the NASA teleoperation application is with the use of a laser scanning device. To demonstrate the ease of integrating a laser scanner data base with HERSS, a data base was acquired from Servo-Robot of Canada. The data was readily converted to the standard HERSS data base file format and a holographic view of scanned image was generated using the HGS subsystem.

5. HOLOGRAPHIC GENERATION SYSTEM (HGS)

The purpose of the HGS is to generate a holographic view of the remote object using the data produced by the DIPS. Several goals were established for the HGS development:

- Map image data representing a six inch cubic volume at the remote work site onto a holographic volume of the same size;
- Produce a near-real-time display with an initial frame update rate of 30 seconds;
- Provide a depth resolution of two to three millimeters;
- Develop a display system that maximizes operator safety and comfort; and
- Provide a capability to overlay graphics onto the hologram.

In generating a single holographic frame, the HGS uses the shape and brightness functions computed by the DIPS subsystem. As previously discussed these functions define the location and brightness of points contained within each longitudinal depth plane. The maximum number of depth planes available depends on the longitudinal sampling distance of the IAS. For example, if the IAS longitudinal stepping distance, or resolution, were 3 mm, then 51 depth planes would be obtained. If the IAS depth resolution were 2 mm, 76 depth planes would be obtained. As the number of image planes increases, the final holographic image becomes more similar to the actual surface.

To generate a holographic view of the remote object, the proposed HGS design concept requires multiple, sequential exposure of the depth plane images onto a recording material. The recording material selected for HGS was thermoplastic because of its near-real-time capability. A SLM serves as the object source for an individual depth plane because its 2-D pixel matrix is computer addressable. It can also emit a coherent light pattern which is a requirement for holographic generation. As shown in Figure 5-1, the SLM is mounted on a sliding table with a six inch travel. For the first exposure, the table is positioned at one

extreme. At that position, the SLM displays the image points for the first depth plane. This SLM image is exposed onto the thermoplastic. When the first exposure is complete, the table is moved to the next depth plane location (e.g., 2 mm more distant from the thermoplastic camera). The SLM displays the image points for the next depth plane and the thermoplastic is exposed again. This move-display-expose cycle is repeated until the entire object volume is exposed onto the recording material. When the exposure process is complete, the material is developed and a single holographic frame is viewable.

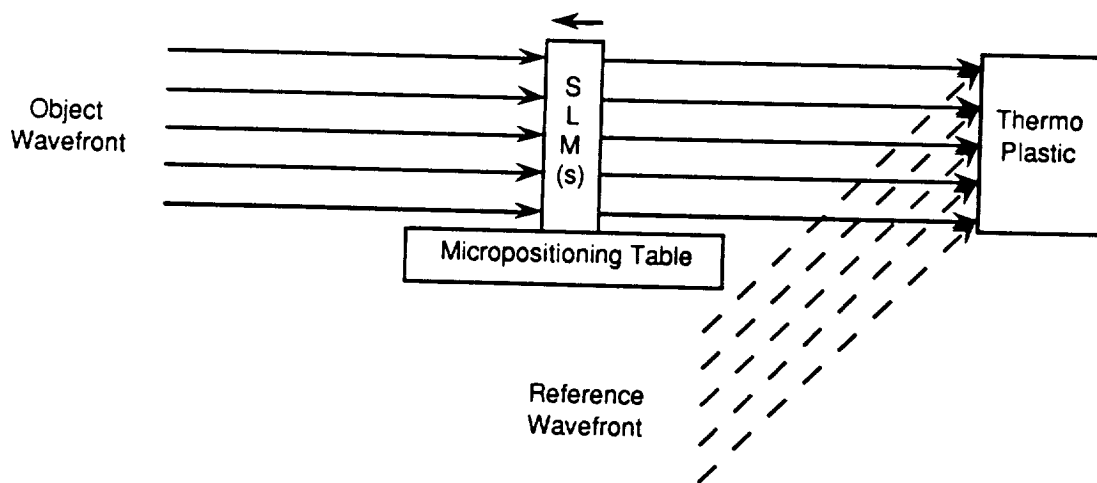


Figure 5-1. Holographic Generation Scheme

HGS development occurred over several phases:

- Perform detailed optical design for the proposed baseline concept.
- Review the state-of-technology for SLMs and near-real-time holographic recording materials to determine recent technological advances in these areas prior to equipment purchase .
- Perform an early proof-of-concept demonstration at the University of Colorado and test the performance of key components.
- Build and test a baseline system at the Analytics laboratory;
- Develop and test an alternative holographic recording configuration, the phase conjugate optical system;

- Develop and test an alternative HGS exposure method using a multiplane-by-multiplane recording technique; and
- Perform testing with an alternative HGS recording material, the Du Pont Series 700 holographic recording film.

The detailed optical design for the baseline system is described in Section 5.1. A review of the state-of-technology of SLMs and holographic recording materials is presented in Section 5.2. Early proof-of-concept studies at the University of Colorado are described in Section 5.3. These first three tasks were critical in defining the path for the remainder of the effort. As a result of these initial tasks, recommendations were made to and approved by NASA to purchase equipment components differing from that originally proposed. Additional tasks were also approved to develop alternative configurations to circumvent the limitations of the original design identified during early testing at the University of Colorado. Initial baseline development and system integration tasks performed at the Analytics HERSS laboratory are discussed in Section 5.4. Development work associated with the alternative phase conjugate recording configuration are discussed in Section 5.5. Development work associated with the alternative multiplane-by-multiplane exposure method are presented in Section 5.6. In Section 5.7, experimental results using a newly released Du Pont photopolymer (Weber, et al., 1990) as the holographic recording material are presented. Experimentation and optimization efforts for each configuration are presented within each subsection. An overall analysis of HGS performance is presented in Section 5.8. A summary is provided in Section 5.9.

5.1 DETAILED OPTICAL DESIGN OF THE BASELINE SYSTEM

Dr. Lloyd Huff developed the detailed optical design for the baseline HGS subsystem as illustrated in Figure 5-2. In the figure, the heavy solid line depicts the laser beam before it is split into the object and reference beam wavefronts. The medium weight line represents the reference beam path; the lightest line represents the object beam path. The major components, as labeled in the figure, and their functions are:

- A. Beam riser — raises the beam to the desired height of the optical system (approximately 11 inches above the surface of the table).
- B. Electronic shutter — controls the time for each exposure. The shutter can be set to an exposure as short as 0.0001 second.

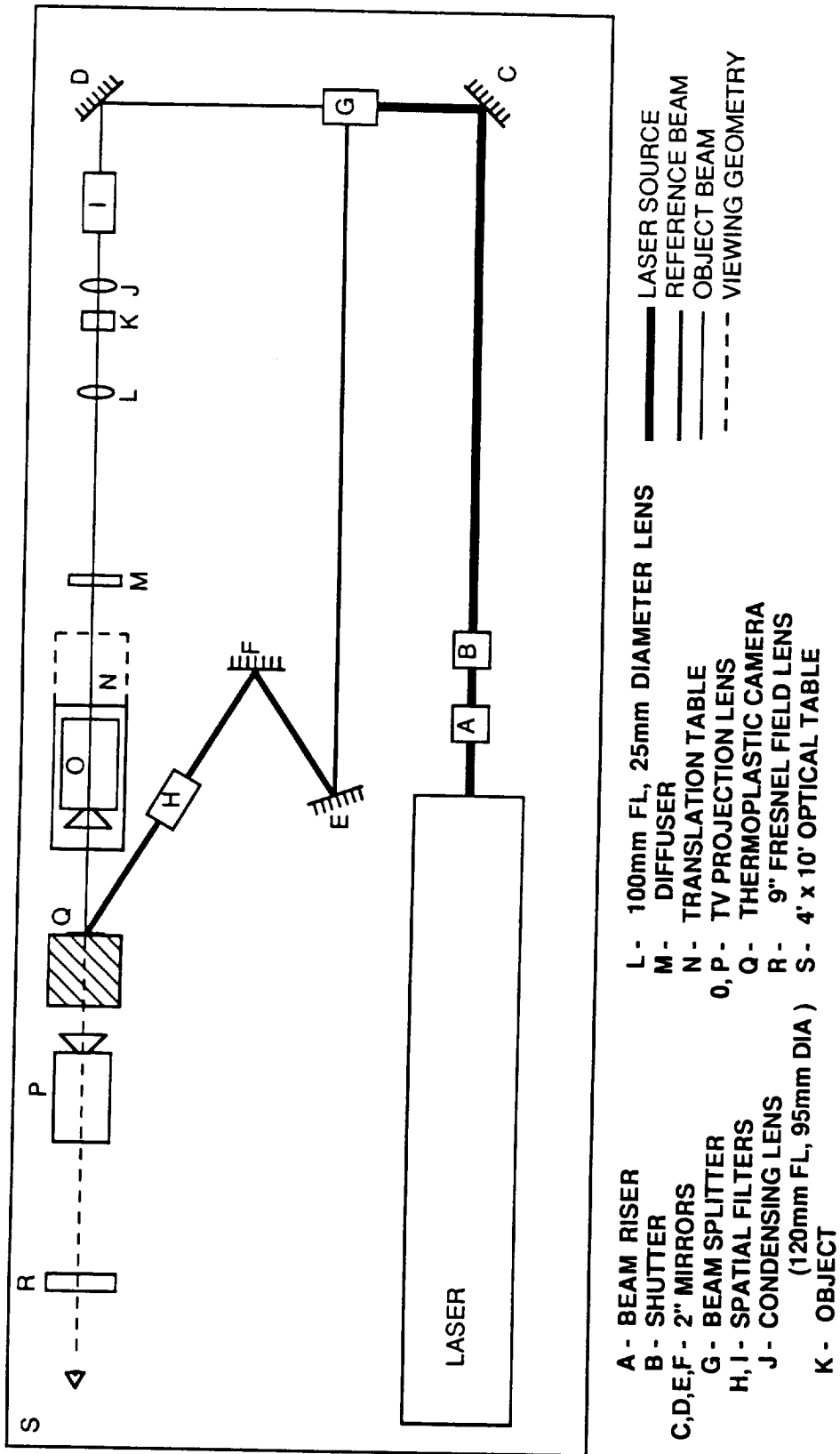


Figure 5-2. HGS Baseline: Optical Configuration

- C. 2" mirror — directs the beam to a variable polarizing beam splitter.
- G. Variable polarizing beam splitter. Splits the beam into a pair of mutually perpendicular beams. One beam forms the object wavefront and travels in the same direction as the input beam. The other beam forms the reference wavefront and is redirected 90° to the left. The optics in the beam splitter ensures that the output beams have the same polarization. The relative intensities of the two beams is controlled by a knob on the beam splitter. By adjusting this knob, it is possible to vary the object/ reference beam intensity ratio, also known as the *K* ratio.
- E,F. 2" mirrors — Directs the reference beam to a spatial filter and extends the path length of the reference beam so that the object and reference beam path lengths are approximately equal.
- H. Spatial filter — Directs the reference beam through a 10 μm pinhole and a 20X objective. The pinhole produces a point source illumination. The pinhole is aligned with the 20x objective so that a uniform cone of expanding light is output to fully illuminate the thermoplastic camera (Q) with the reference wavefront.
- D. 2" mirror — Redirects the object beam to a spatial filter.
- I. Spatial filter — Directs the object beam through a 10 μm pinhole and a 20X objective to produce a uniform cone of expanding light.
- J. Condensing lens — 120 mm focal length lens that condenses the light to fully illuminate the object.
- K. Object — 2-D transmissive object that serves as the coherent light source for the hologram. The object can be a 35 mm slide, a glass plate, or a transmissive SLM.
- L. Projection lens — 100mm focal length lens to project the image of the test object onto a diffuser.
- M. Diffuser — Localizes the image of the object.
- N. Micropositioning table — Varies the location of the TV projection lens (O) over a six inch travel.
- O. TV Projection lens — Relays and minifies the image reaching the one-inch square thermoplastic recording plate.
- Q. Thermoplastic camera — Electro-mechanical device for erasing and developing the holographic image stored on a thermoplastic plate.

- P. TV Projection lens — Projects and magnifies the holographic image in the direction of the viewer.
- R. Fresnel field lens — Relays and magnifies the holographic image to the viewer's eye.
- S. Optical table — Provides vibration isolation during holographic exposure.

When generating a holographic image, 2-D object planes (K) are sequentially exposed onto the thermoplastic plate (Q). The location of each 2-D depth plane in the holographic space is a direct function of the position of the relay lens (O). Before exposing a 2-D plane, the position of the relay lens is varied by directing the micropositioning table to a specific location along its six-inch travel. As the relay lens moves, the size and the location of the diffuser image at the thermoplastic plate changes. If a square annulus on a glass plate is exposed, a 2-D hologram of the single square annulus is recorded. But with multiple exposures and stepping of the relay lens between exposure, a 3-D pyramidal shape is recorded. That is, subsequent images of the square within the hologram are displaced in depth and are of increasing size as illustrated in Figure 5-3.

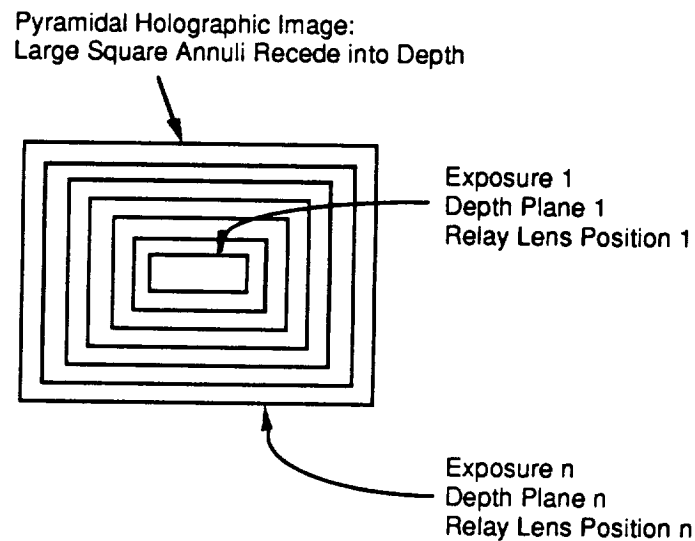


Figure 5-3. Multiple Exposure of a Square Annulus through Depth to Create a 3-D Pyramid

After the thermoplastic has been multiply exposed, the thermoplastic is developed to generate the hologram. To view the hologram, the object beam is blocked and the thermoplastic plate is illuminated by the reference beam only. The observer sits directly behind the thermoplastic camera. Between the observer and the thermoplastic camera is a TV projection lens and a Fresnel field lens with a 9 inch focal length. The viewing optics was designed to magnify the holographic image stored on the relatively small thermoplastic plate. The thermoplastic plate has an active area of 1.5 square inches. The system is capable of magnifying an image approximately threefold. This produces an image volume corresponding to an approximate 4.5" x 4.5" x 6" depth area.

In the baseline HGS configuration, use of a relatively low-powered helium neon laser was anticipated. The eventual HGS configuration required use of a 2-Watt argon laser as described in Section 5.3 and 5.4.

5.2 REVIEW OF THE STATE-OF-TECHNOLOGY OF SLMs AND NEAR-REAL-TIME RECORDING MATERIALS

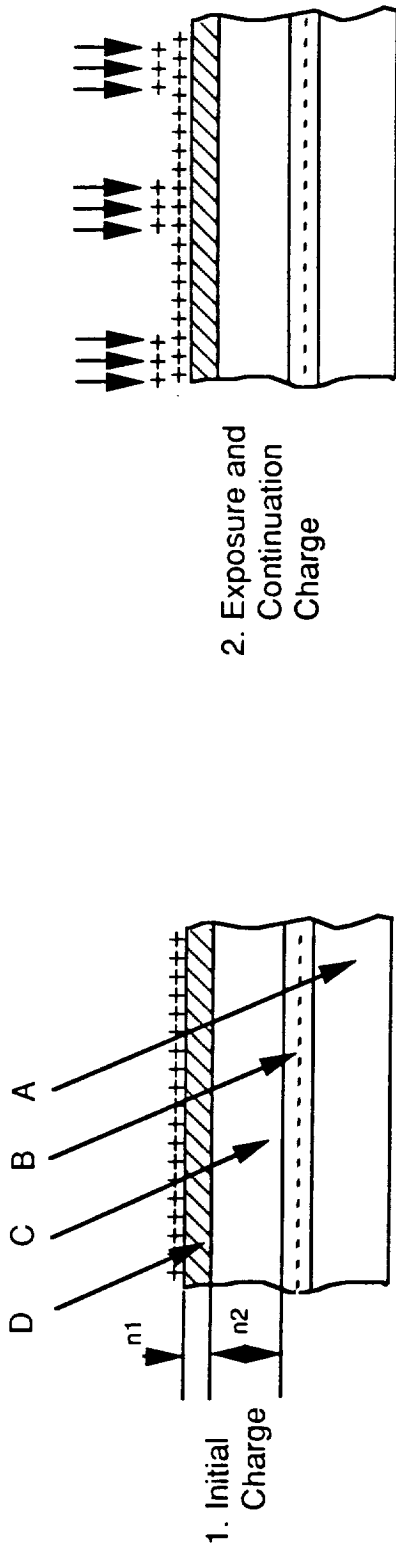
In 1987, when the Phase II proposal was written, the Hughes liquid crystal light valve (LCLV) was selected as the SLM component for the HERSS prototype. The Hughes LCLV had been under development for several years and was widely used as an incoherent to coherent light converter in Fourier optical systems. The Hughes LCLV, and other such research grade SLMs, were the only devices available for impressing imagery on a coherent beam at the time the proposal was written. However, the cost was high; the LCLV and an associated driver were \$40K. In early 1989, low cost SLMs (less than \$2K) were being mass-produced by various Japanese companies and installed as displays in hand-held portable liquid crystal televisions as well as in portable computers. These alternative SLMs were investigated in order to assess relative cost/performance tradeoffs with the Hughes system. In particular, a Seiko-Epson, Kodak Data Show, and nView SLM were evaluated.

Based on a survey of these devices, including optical performance, cost, flexibility, and the experiences of other researchers, it was concluded that the cost of the Hughes device was not justified. In fact, Andrews et al. (1988), using a Kodak Data Show as the holographic object source, produced holographic stereograms on photographic film. Furthermore, some

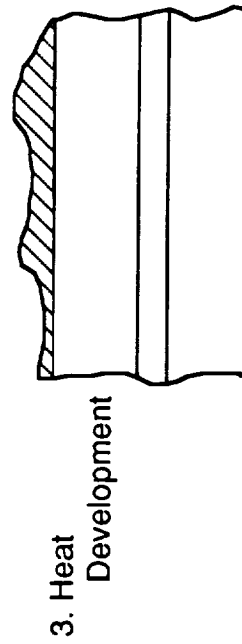
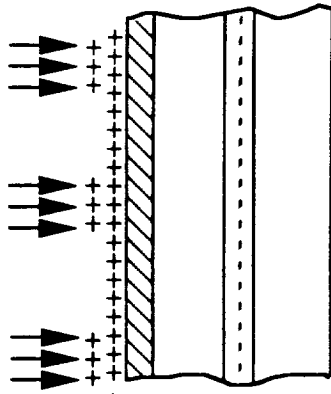
clear disadvantages were found with the Hughes LCLV as compared with the newly available alternatives. Specifically, the Hughes device was non-uniform; the performance of each liquid crystal varied throughout the display area. The Hughes device was also only a reflective device, a limiting factor in potential HGS design configurations. The newer alternatives were uniform and provided the same or better information content than the Hughes device. While the Hughes system provided the highest contrast (600:1 for the best performing regions of the display), the cost could still not be justified considering the other performance factors. It was determined that the nView SLM provided the best contrast (20:1) among the low cost SLMs at that time.

Alternative commercially-available thermoplastic materials were also investigated. Two primary sources were identified -- Newport (USA) and Micraudel (FR). The Newport holographic camera consists of a camera unit and a controller. The thermoplastic is mounted on a photoconducting substrate that is applied to a glass plate. The glass plate is approximately 1.5 square inches and is held by a plastic frame that slides into ridges in the camera for easy replacement. The 1.5 inch format places a severe constraint on the performance requirements of the HGS display optics system. That is, a one to four magnification ratio is necessary in order to obtain a six inch viewing cube. The Micraudel system uses a thermoplastic film similar to the Newport system but the film is mounted on a transparent flexible plastic strip. The camera advances the film strip for each hologram by rolling the film from one post to a second post. The magnification requirement is cut in half by the Micraudel system which has a 2 x 2.8 inch format. Performance of both the Newport and Micraudel thermoplastic were evaluated during experimentation at the University of Colorado lab in late February, 1989.

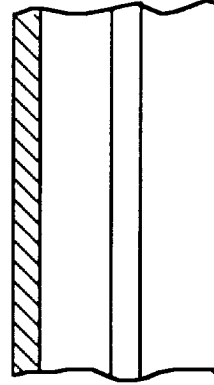
Figure 5-4 illustrates the structure and recording steps of a thermoplastic device. Before a hologram is recorded, the thermoplastic is heated to erase any existing deformations in the thermoplastic. After erasure, a positive charge is applied to the thermoplastic. When the plate is subsequently exposed to light, the charges in the more intensely illuminated regions migrate through to the conductive layer and are dissipated. The result is a surface charge distribution whose intensity varies inversely to the incident light exposure. Following exposure, the hologram development process is initiated by first applying a brief but intense electrical current through the conductive layer. This results in a heating and corresponding



2. Exposure and Continuation Charge



4. Heat Exposure



- A - Substrate
- B - Transparent Electrode
- C - Photoconductive Layer
- D - Thermoplastic Layer

Figure 5-4. Structure and Recording Steps of a Thermoplastic Material

softening of the thermoplastic. Upon cooling, the surface deformations form a relief phase grating. The hologram can be viewed when incident reference waves reconstruct the holographic object wave. The plate can be reused by heating the plate and erasing the deformations. The erasing and exposing procedure takes approximately 30 seconds and the developing procedure also takes approximately 30 seconds. Optimum exposure energy varies between 5 and 7 $\mu\text{J}/\text{cm}^2$.

In 1988, Du Pont was in the early stages of developing a new photopolymer material (HRF-352) for use as a holographic recording medium. Du Pont supplied a sample of the photopolymer to Analytix. The photopolymer was mounted on 8 1/2 by 11 inch sheets of Mylar. The material could be cut to any size and then sandwiched between two pieces of glass. As the name implies the photopolymer is a light sensitive complex chemical compound. It consists of several ingredients including polymeric binder, photoinitiator system, polymerizable monomers, and sensitizing dye (Weber, et al., 1990). When the photopolymer is exposed, assuming there is a sufficient activation energy, monomer polymerization starts and proceeds in the exposed regions. In these regions, the monomer is converted to a photopolymer. Monomers diffuse from adjacent regions creating density gradients. As exposure continues, the originally viscous material hardens and further hologram recording is stopped. Final illumination of the material with ultra-violet light polymerizes the remaining monomer and fixes the image. Further archival quality processing requires an additional baking step. This step is not considered necessary for a near-real-time holographic recording process. Optimum exposure energy varies between 15 to 80 mJ/cm^2 depending on holographic recording configurations with less energy required for reflection holograms compared to transmission holograms.

Finally, photorefractive crystals were also evaluated as a possible medium for volumetric display. However, the assessment indicated that the approach was impractical due primarily to the limited crystal size. The crystal size, limited to two to five mm, would place an even greater burden on the HGS display optics to magnify the image to a reasonable size.

5.3 EARLY PROOF-OF-CONCEPT AND COMPONENT TESTING AT THE UNIVERSITY OF COLORADO

Dr. Kristina Johnson of the Electrooptical Computing Center at the University of Colorado provided use of laboratory facilities to the HERSS development team in February, 1989, for early proof-of-concept testing and component evaluation. Several key technical questions involving the baseline HGS optical design were to be investigated including:

- Would generation of a hologram be possible if one of the optical components were mounted on a motor-driven micropositioning device? The concern here was that if the motor was continuously on, vibrations generated by the motor would transfer to the optical component mounted on the table. This could disturb the light wave pattern being transmitted to the thermoplastic and could possibly lead to a failed hologram.
- What holographic image quality is possible using the proposed optical configuration?
- How many multiple exposures are achievable with the Newport thermoplastic camera compared to the Micraudel camera before image quality degrades? With the Du Pont photopolymer?
- Will holographic image quality vary if a helium neon laser is used versus an argon ion laser?

5.3.1 EXPERIMENTATION

Experimentation with the baseline configuration included:

- Single exposure of the Newport thermoplastic with the motor of the stepper positioning table turned on and off.
- Single and multiple exposure of the Newport thermoplastic using a glass plate square annulus as the holographic object source.
- Single and multiple exposure of the Micraudel thermoplastic using a glass plate square annulus as the holographic object source.
- Single and multiple exposure of the Du Pont photopolymer (HRF-352 Series) using a glass plate square annulus as the holographic object source.
- Exposure of the Newport thermoplastic using an argon-ion laser in place of the helium-neon laser used in the previous experimentation to determine if laser type significantly affected image quality.

5.3.2 RESULTS

The baseline optical configuration presented in Figure 5-1 was setup on a 4' x 10' optical table at the University of Colorado laboratory using a Spectra-Physics 120S helium-neon laser. A 35 mm slide of an Air Force resolution test chart was used as the object source and the Newport camera was used as the holographic recording medium. There were some initial difficulties in obtaining a good hologram with the Newport system. Eventually the problem was traced to improper settings of the corona and development voltages. Once these were adjusted to match the equipment specifications, it was possible to generate holograms with relatively little background noise. The K ratio (the ratio of reference beam power divided by object beam power) and total exposure energy at the plate were varied until a sharp, bright, low noise hologram was generated.

Prior to experimentation, several holograms were generated to determine the optimum K ratio and exposure energy for the Newport thermoplastic. A 35 mm slide of the Air Force resolution test chart was used as the object source. The optimum results were obtained with a K ratio of approximately 2.5 and an exposure intensity of $70 \mu\text{J}/\text{cm}^2$.

To determine the effect, if any, of vibration emanating from the micropositioner motor on the holographic generation process, a hologram was created with the motor power turned on and off. A 35mm slide of the Air Force resolution test chart was used as the object source. To minimize the vibration with the motor on, a command was sent to the micropositioner to reduce the motor power to the lowest level possible following completion of a table move. With the motor power on or off, the holographic image was found to be of equal quality.

To perform the multiple exposure testing, a square annulus was used as the test object. The object was created by first covering a glass plate with black tape. The shape of a 1.5" square annulus was then cut out of the background with a line width of approximately 0.125 inch. In effect, the object was a bright annulus on a black background as the laser light was transmitted from the rear of the glass plate and moved in a direction toward the thermoplastic camera. As previously mentioned, multiple exposure of a square annulus using the baseline configuration generates a hologram of a 3-D pyramid. Specifically, as the relay

lens is moved farther away from the thermoplastic camera, the location of the image in the holographic volume is displaced relative to the original image. In addition, the magnification of the image increases as the relay lens moves away from the thermoplastic camera. As illustrated in Figure 5-3, with each consecutive exposure, the image of the square becomes larger and more distant within the holographic volume. Also, as the number of depth slices increases, the pyramid appears to look more and more like a solid object.

Multiple exposure holograms were generated using the Newport camera and the glass square annulus as the test object. Five, 10, 20, and 29 multiple exposures were attempted. Good holograms were generated with 5 and 10 exposures. At 20 exposures, there was a noticeable degradation in the hologram, but the image was still presentable. At 29 exposures, the holographic image was visible, but the noise level was approaching an unacceptable limit.

Similar experiments were conducted with the Micraudel Camera. For 5 and 10 multiple exposures, the results were very similar to those obtained with the Newport system. However, for 20 and 29 exposures we were unable to obtain any holograms. Based primarily on the hardware difficulties encountered with the Micraudel system and the inability of the system to generate holograms at the higher level of exposures, this system was judged unsuitable for use in HGS.

A limited set of multiple exposure experiments were also conducted with the Du Pont photopolymer. The photopolymer exhibited superior multiple exposure performance compared to the thermoplastic in terms of image quality. However, considerably longer exposure times were required. The recommended photospeed of the photopolymer was specified in millijoules of power per area compared to the microjoule specification for the thermoplastic. Thus, long exposure times were necessary with the photopolymer especially with use of the relatively low powered helium-neon laser. Another disadvantage of the photopolymer was that the material was still experimental and contained potentially hazardous substances. Specifically, one of the ingredients of the photopolymer tested positively for mutagenicity (Harrington, 1989). Unsafe exposure could occur in the unlikely event that the material contacted the skin directly or if a fire caused hazardous gases and vapors to be produced.

A comparative analysis of the results using each recording material was conducted. The Newport thermoplastic was selected as the preferred recording material for the HERSS near-real-time application. The Newport camera could produce good quality holograms in the shortest time frame. Furthermore, it was a proven, commercially-available system. The major drawbacks to Newport's system are the time required by the camera for the erase-expose-develop cycle (50 seconds) and the limited size of the thermoplastic plates (1.5 inch square). However, it was concluded that these disadvantages could be overcome if the Phase II development efforts indicated that use of the material in the eventual HERSS system was warranted. Lee, Marzwell, Schmidt, and Tufte (1978), working at Honeywell, reported use of a thermoplastic material with development and erasure durations of 4 - 100 msec. The Honeywell material was actually the prototype thermoplastic used in the Newport camera. The faster erase and develop cycles in the Honeywell system were attributed to a faster heating process of the thermoplastic (Tufte, personal communications, 1990).

It was also concluded that the Du Pont material provided a viable backup as an alternate holographic recording material because of the superior quality of the hologram following multiple exposure. If at a latter stage, the Newport camera's performance was judged unacceptable for the teleoperation application, then further research would be performed using a Du Pont photopolymer material. In fact, subsequent research was performed with a Du Pont material (Series HRF-700) that became available in 1990 as reported in Section 5.7.

Another major goal of the experimentation conducted in Colorado was to evaluate the performance of the baseline optical design. Overall the quality of the holograms produced by the baseline system was good and was comparable to other holographic configurations. However certain limitations of the baseline system were noted including:

- Limited number of multiple exposures. If a depth resolution of 2 mm (or 3 mm) was to be met, then 76 (or 51) depth planes needed to be stored on the thermoplastic. The image generated on the thermoplastic seriously degraded following multiple exposure of 30 depth planes.
- Some distortion of the image was noted resulting from defects in the HGS optics.
- The size of an image plane varied from one depth to the next. In the eventual system, a software preprocessor would be necessary to adjust the image size prior

to exposure to counter the magnification changes induced by the relay lens. The additional software processing would slow the frame update rate. Furthermore, special requirements would be placed on the SLM. The SLM would need to be capable of displaying both a large and a small image. A fine pixel resolution would also be needed on the SLM so that pixel density could be roughly equated across for small and large image slices.

- The field-of-view of the holographic image was limited to approximately 20°.

To circumvent the multiple exposure limitation, a multiplane-by-multiplane exposure method was devised as described in Section 5.6. Basically, this method uses a stack of SLMs to expose several depth planes simultaneously instead of exposing only one depth plane at a time. Each SLM in the stack is slightly separated in the axial plane to represent the real distance between image slices. The multiplane-by-multiplane exposure method had the potential to at least double the information content, or depth resolution in the final hologram, by doubling the number of depth planes stored in the six inch display volume.

In order to enlarge the viewing angle, investigation into the use of a fly's eye lens (FEL) would also be undertaken. A FEL is composed of a matrix of small lenslets. Each lenslet, representing the view from a unique perspective, could relay the object image to the thermoplastic. With a greater spectrum of angles incident on the thermoplastic, it was conceivable that the field-of-view could increase. Furthermore, if a quality image could be formed with a FEL and if the multiple SLM stack concept did not perform to expectations, another configuration using the FEL would be considered to increase the information stored on the thermoplastic. Specifically, a FEL could be used in conjunction with a spatial multiplexing technique to reduce the exposure per recording area and its associated bias buildup. With each sequential exposure on the thermoplastic plate, a moving mask would allow only one quadrant of the thermoplastic to receive the exposure.

To circumvent the image distortion exhibited in the baseline system and to avoid the requirement to adjust the image size in software before exposure, a phase conjugate optical configuration was devised. This configuration is based on the principles of phase conjugation and reverse ray tracing. When generating the hologram, the object image passes through a series of lenses that inevitably distort the image recorded on the thermoplastic. However, when viewing the hologram, the object image is reconstructed by transmitting the light back

through the optics in a reverse path. This has the effect of eliminating the distortions induced by the optics during hologram generation including minification or magnification changes. The detailed design effort for this configuration encompassed use of the multiplane-by-multiplane exposure method and the FEL testing. Section 5.5 describes the phase conjugate recording configuration as well as the FEL testing. Section 5.6 describes testing the multiplane-by-multiplane exposure method.

Finally, an analysis of the laser power requirements for the alternate configurations revealed that power requirements would increase. The multiplane-by-multiplane exposure method, in particular, resulted in a considerable loss of energy as the light was transmitted through the SLM stack. To verify that image quality would not vary with use of a higher power laser, holograms were generated using the thermoplastic camera in conjunction with an argon ion laser. The results indicated that holograms of comparable quality could be generated with either the argon ion or the helium neon laser as a light source. In both cases, bright, clear, sharp holograms were obtained. Use of a 2-Watt argon ion laser was subsequently proposed and approved.

5.4 BASELINE DEVELOPMENT AT THE ANALYTICS LABORATORY

Following equipment acquisition and component testing, the baseline configuration was reestablished at the Analytics laboratory in Willow Grove, Pennsylvania. Three experimental goals were set for testing with the baseline:

- (1) Duplicate the results obtained during the preliminary testing in Colorado to ensure that the equipment acquired for the HERSS laboratory was functioning properly;
- (2) Characterize the optimum performance of the thermoplastic for single and multiple exposure; and
- (3) Perform preliminary experiments focusing on use of an SLM as the object source for the hologram.

5.4.1 EXPERIMENTATION

Experimentation with the baseline configuration included:

- Single and multiple exposure of a bright square annulus on a black glass plate to repeat the Colorado testing and to characterize the optimum performance of the Newport camera,
- Single exposure of an nView SLM on the thermoplastic to determine the baseline performance associated with use of that SLM as the holographic object source.

5.4.2 RESULTS

The baseline configuration, as illustrated in Figure 5-2, was set up at the Analytics laboratory. Prior to experimentation, the performance of key optical components were independently tested. This preliminary testing focused on checking the performance of the variable polarizing beam splitter as well as the thermoplastic camera. For the beam splitter, the polarization of the object and reference beams were monitored and found to be correct throughout the range of possible values. Some difficulty was experienced with use of the thermoplastic camera. In particular, it was necessary to adjust both the corona and development voltages to a minimum value to avoid a fog on the plate which would create a very noisy image. Eventually, both voltages were adjusted to appropriate settings, but throughout experimentation at the Analytics laboratory, the settings for these voltages would drift and would need to be repeatedly adjusted.

Following initial setup and component testing, experimentation to characterize the multiple exposure capability of the thermoplastic was conducted. Initial testing focused on repeating and extending the Colorado testing of the Newport camera performance. An open square annulus on a black glass plate was used as a test object. Single and multiple exposures were attempted for K ratios ranging between 0.4 and 60.0 and for energy exposures ranging between 6 and 30 $\mu\text{J}/\text{cm}^2$. For multiple exposures, the location of the depth planes were made to be equidistant within the six inch volume of the hologram.

For the single exposure case, clear, bright, sharp holograms of the test object were found for K ratios less than 2.5 and with exposure energy between 6 $\mu\text{J}/\text{cm}^2$ and 8 $\mu\text{J}/\text{cm}^2$.

Table 5-1. Single and Multiple Exposure of the Thermoplastic Camera Using Bright Square Annulus on a Black Background

	# of Exposures	K Ratio	Energy	Hologram Results
Series A	1	2.5	7.8 μ J/cm ²	Clear, Sharp, No Noise.
	10	2.5	7.8 μ J/cm ²	Clear, Sharp, Slight Noise Buildup.
	15	2.5	7.8 μ J/cm ²	Good Hologram. Noise is increasing
	20	2.5	7.8 μ J/cm ²	Good Hologram. All Depth Planes Visible Noise is Evident. Hologram is Dimmer.
Series B	1	5	7.0 μ J/cm ²	Clear, Sharp, No Noise.
	5	5	7.0 μ J/cm ²	Clear, Sharp, Slight Noise Buildup.
	10	5	7.0 μ J/cm ²	Sharp, Slightly Noisier.
	20	5	7.0 μ J/cm ²	Considerable Noise. Depth Planes are Indistinct and Blurry.
	20	2.8	7.0 μ J/cm ²	Less Noisy. Some of the Depth Planes are Visible.
	20	1	7.0 μ J/cm ²	Less Noise. All Depth Planes are Visible.
	20	0.54	7.0 μ J/cm ²	Similar to Previous Result.

For multiple exposures, the optimum K ratio and power was dependent on the number of multiple exposures. Table 5-1 illustrates results for two series of multiple exposure trials. In Series A, plate exposures varied between 1 and 20 while the K ratio and energy were held constant at 2.5 and 7.8 μ J/cm², respectively. While noise buildup increased with the number of exposures, all depth planes were still distinctly visible at 20 exposures. In Series B, the K ratio was fixed at 5 and the total energy at the plate was held at 7.0 μ J/cm². The results indicated that an unacceptable level of noise occurred at 20 exposures: the depth planes were indistinct and blurry. When the K ratio was subsequently reduced to 1, the noise at 20 exposures was minimized. There was converging evidence from these as well as other series of trials that the optimum K ratio for multiple exposures was 1 and that the optimum energy at the plate was 7.0 μ J/cm².

These results indicated that 20 multiple exposures could be obtained on the thermoplastic without serious degradation. It was concluded that the baseline configuration using a plane-by-plane exposure method could not meet the depth resolution goal of between 51 and 76 image slices. It was necessary to pursue the multiplane-by-multiplane exposure method as described in Section 5.6.

Initial testing using an SLM as the holographic object source was also conducted. The nView had an approximate 6 x 6 inch active area and operated in the transmissive mode. The nView SLM was readily addressable by a video card installed in the Macintosh-IIx. The image of a square annulus, generated by the central computer software, was transmitted to the SLM before exposure onto the thermoplastic. With only one exposure of the SLM onto the thermoplastic, a poor holographic image were generated. This was attributed to the low level of contrast that was available using this SLM. While the glass plate test object had an approximate contrast ratio of 1000:1, the nView SLM had a contrast ratio of 20:1. This low contrast ratio is attributed to considerable light leakage through cell areas that are closed.

It was concluded that the performance of the nView SLM was unacceptable for use in HGS. Further testing using an SLM as the object source would be conducted using the SLM components of the SharpVision video projection system.

A Sharp video projection system containing three SLMs became available on the market in early 1990. It was equivalent to the cost of the nView but had a higher contrast ratio of approximately 30:1. Each SLM has a 384 x 240 pixel resolution within an active area about 2.4 inches wide by 1.8 inches high. Further testing with the Sharp SLM was conducted using the phase conjugate recording system as reported in the following section.

5.5 PHASE CONJUGATE CONFIGURATION

Dr. Lloyd Huff developed the detailed optical design for the phase conjugate recording system. As previously discussed, this configuration is based on the principles of phase conjugation and reverse ray tracing. This configuration was designed to improve the performance of the HGS by removing optical distortions from the holograms and eliminating the need for complicated correction algorithms for the SLM imagery. Any image distortions

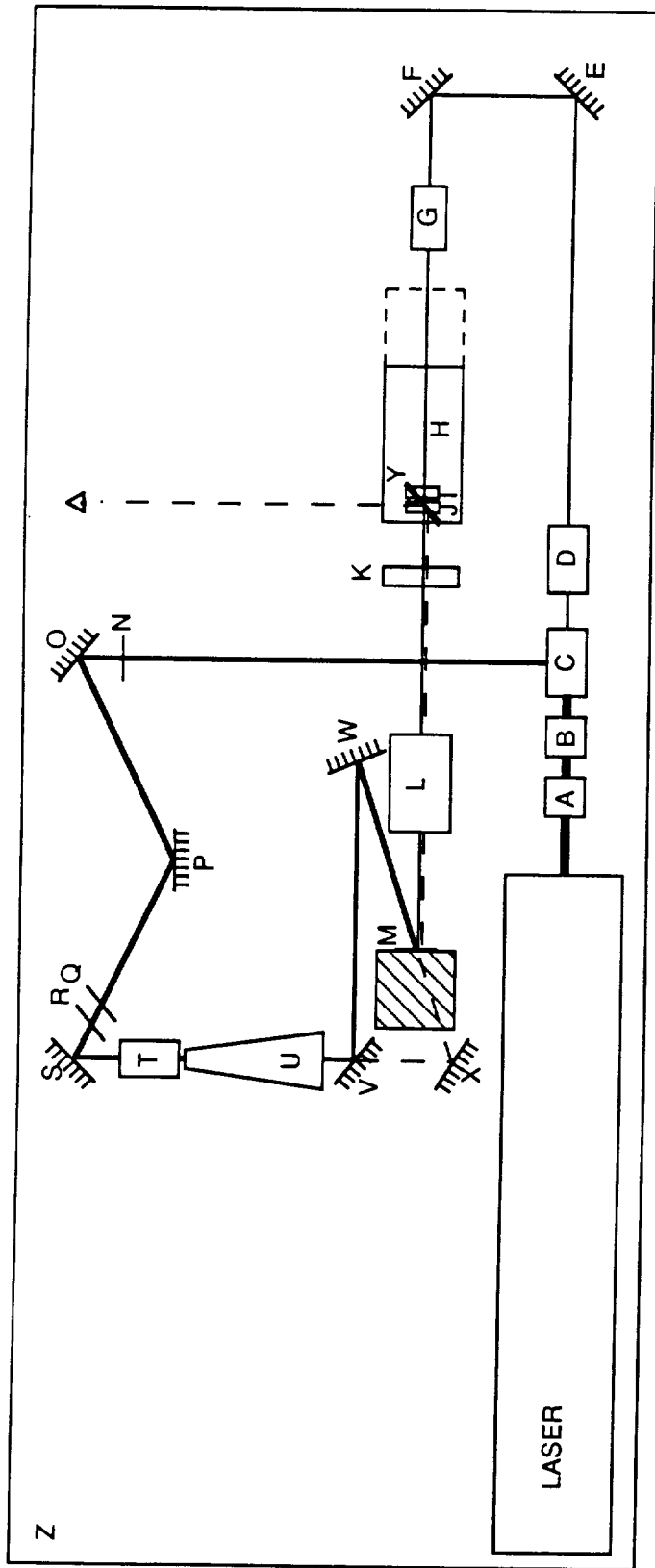
induced by the optics during hologram generation are removed during the phase conjugate reconstruction. Furthermore, the images within the hologram are presented at the same scale as the original object, that is, without magnification or minification.

Figure 5-5 illustrates the components of the system. The configuration is slightly modified depending on whether a hologram is being generated or is being viewed by the observer. In the generation mode, the configuration is arranged so that the object and reference beam wavefronts interfere at the front face of the thermoplastic camera, as is required for generation a transmission hologram. In the viewing mode, the reference wavefront is transmitted through the rear of the thermoplastic plate to reconstruct the original object wavefront. The reconstructed wave then travels in a reverse path through the optical system. Use of this technique in reconstruction provides for the elimination of any image distortions induced by the optics during hologram generation.

Another important factor in the design of this system focused on use of the SharpVision SLM units as the holographic object source. The surface of the Sharp SLM is coated with a polarizer such that transmitted light is polarized at a 45° angle. Thus, to maximize the energy transmitted through the plate, the object beam polarization was set to 45° before striking the SLM. Prior to interference of the object and reference beams at the front face of the thermoplastic, it was necessary to adjust the polarization of the reference beam so that it matched the polarization of the object beam.

The system uses a Coherent 2-Watt argon laser to generate the object and reference beams. A note is provided in Figure 5-5 for those components that are uniquely used in either the generate or viewing mode. The components, as labeled in the figure, and their functions are:

- A. Beam riser — raises the beam to the desired height of the optical system (approximately 11 inches above the surface of the table).
- B. Electronic shutter — controls the time for each exposure. The shutter can be set to an exposure as short as 0.0001 second.



- | | |
|------------------------------|---|
| A - BEAM RISER | O,P,S - 2" MIRROR |
| B - SHUTTER | Q - POLARIZER |
| C - BEAM SPLITTER | R - ATTENUATOR |
| D - ROTATIONAL POLARIZER | T - SPATIAL FILTER |
| E,F - 2" MIRROR | U - COLLIMATOR |
| G - SPATIAL FILTER | V - 4" MIRROR ON KINEMATIC BASE (Generate Mode) |
| H - TRANSLATION TABLE | W - 4" MIRROR |
| I - DIFFUSER (Generate Mode) | X - 4" MIRROR (Viewing Mode) |
| J - OBJECT (Generate Mode) | Y - 8" x 11" MIRROR (Viewing Mode) |
| K - 9" FL FRESNEL FIELD LENS | Z - 4' x 10' OPTICAL TABLE |
| L - TV PROJECTION LENS | |
| M - THERMOPLASTIC CAMERA | |
| N - HALF-WAVE PLATE | |



Figure 5-5. HGS Alternative: Phase Conjugate Optical Configuration

- C. Variable polarizing beam splitter — splits the beam into a pair of mutually perpendicular beams. One beam forms the object wavefront and travels in the same direction as the input beam. The other beam forms the reference wavefront and is redirected 90° to the left. The optics in the beam splitter ensures that the output beams have the same polarization. The relative intensities of the two beams is controlled by a knob on the beam splitter. By adjusting this knob, it is possible to vary the object/ reference beam intensity ratio.
- D. Rotational polarizer — changes the orientation of the object beam polarization to 45 degrees off-axis.
- E,F. 2" mirrors — directs the reference beam to a spatial filter and extends the path length of the reference beam so that the object and reference beam path lengths are approximately equal.
- G. Spatial filter — directs the reference beam through a $10\ \mu\text{m}$ pinhole and a 10X objective. The pinhole produces a point source illumination. The pinhole is aligned with the 10x objective so that a uniform cone of expanding light is output to fully illuminate the object (J).
- H. Micropositioning table — Varies the location of the object (J) over a six inch travel.
- I. Diffuser — Diffuses light reaching the transmissive object.
- J. Object — 2-D transmissive object that serves as the coherent light source for the hologram. The object can be a 35 mm slide, a glass plate, or an SLM.
- K. Fresnel field lens — images the projection lens to the viewing zone and forms the exit pupil of the system.
- L. Projection lens — 100 mm focal length lens to relay the object wavefront onto the thermoplastic camera (M).
- M. Thermoplastic camera — electro-mechanical device for erasing and developing the holographic image stored on a thermoplastic plate.
- N. Half-wave plate — changes the polarization of the reference beam to match the polarization of the object beam which is off-axis at a 45 degree angle.
- O,P. 2" mirrors — directs the reference beam to a polarizer and extends the path length of the reference beam so that the object and reference beam path lengths are approximately equal.
- Q. Polarizer — removes any light which is not polarized at a 45 degree angle with respect to vertical.

- R. Attenuator — used to adjust the intensity of the reference beam when the K ratio is set. There is considerable light loss in the object beam after passing through the Sharp SLM/s.
- S. 2" mirror — directs the reference beam to a spatial filter.
- T. Spatial filter — directs the reference beam through a 10 μm pinhole and a 20X objective. The pinhole produces a point source illumination. The pinhole is aligned with the 20x objective and a uniform cone of expanding light is output to illuminate the entrance pupil of the collimator (T).
- U. Collimator — produces a collimated reference beam with an approximate 2 inch spot size.
- V. 4" mirror on kinematic base — directs the reference beam to another 4" mirror on the path to the thermoplastic camera (L). The kinematic base allows the mirror to be removed during the viewing mode.
- W. 4" mirror — directs the reference beam to the thermoplastic plate.
- X. 4" mirror — directs the reference beam to the thermoplastic plate to reconstruct the object wavefront during viewing mode.
- Y. 8" x 11" mirror — mirror placed at the 45° angle to direct the object wavefront to the observer's eye during viewing mode. (Note: The object is removed.)
- Z. 4' x 10' optical table — provides vibration isolation during holographic generation mode.

Prior to experimentation, a variety of objectives were evaluated for use within the object beam spatial filter (G). This evaluation was conducted so that the optimum pinhole and objective combination could be found that would fully illuminate the active area of the SLM, no more and no less. Minimizing the spread of the cone of light emanating from the spatial filter would reduce the overall light loss following transmission through the SLMs. The optimum setting used a 10X objective with a 10 μm pinhole.

During the generation mode, 2-D object planes (J) are sequentially exposed onto the thermoplastic plate (M). The location of each 2-D object plane in the holographic space is a function of the position of the object (J). The object is mounted on a micropositioning table (H). Before exposing a 2-D plane, the position of the table, and thus the object, is varied along

the six-inch table travel. Similar to the baseline configuration, a 3-D holographic image is built by sequentially exposing 2-D depth planes.

During the viewing mode, the object (J) and diffuser (I) are removed. A mirror (Y), oriented at a 45° angle from the optical path, is mounted on the micropositioning table (H) in place of the object. To reconstruct the hologram, the object beam is blocked and the mirror on the kinematic mount is removed. This allows the reference beam to reflect from mirror (X) onto the rear of the thermoplastic plate. The light then passes through the TV projection lens, the Fresnel field lens, and is reflected from the mirror (X) in the direction of the observer.

A major advantage of the phase conjugate recording system is the capacity to increase the image magnification relative to what was achieved with the baseline. In the baseline configuration, a large-diameter, low-f/# projection lens (P in Figure 5-2) was required to generate a 3:1 magnification of the 1.5 square inch thermoplastic image. Thus, the holographic display created an image that was 4.5 inches wide. This was short of the desired 6 inch wide display goal. If even lower f/# optics were used in the baseline, considerable image distortion would result. With the phase conjugate recording system, however, it was possible that a lens with a smaller f/# could be used without causing serious image distortions. For example, a lens with a smaller focal length could decrease the f/# ($f/\# = \text{focal length} / \text{diameter}$) and thereby enlarge the resultant display image. During reconstruction, because the light is reversed through the optics, distortions are eliminated. Therefore, one experiment for the phase conjugate system was replacement of the TV projection lens (L) with a lens having a shorter focal.

An experiment using a fly's eye lens (FEL) would also be undertaken. The primary impetus for the use of the fly's eye lens was (1) to "effectively" increase the size of the thermoplastic recording aperture from a 1.5-inch square to a 2- or 3-inch square and (2) to increase the display field-of-view. As previously mentioned, a FEL is composed of a matrix of small lenslets. Each lenslet represents the view from a unique perspective. If a FEL were placed between the projection lens (P) and the thermoplastic plate (Q), a greater spectrum of angles would be incident on the thermoplastic. As the cone of light carrying the image information strikes the FEL, light at the edge of the cone is bent back inward toward the 1.5"

recording medium. This could theoretically increase the field-of-view of the image. Without the FEL, light at the edge of the cone would not strike the recording area and would be lost.

In the eventual HERSS system, in order to accomplish the 2- or 3-inch square "effective" recording aperture, a FEL would be employed with a multiplexing technique. For each exposure, only one quadrant of the thermoplastic would be available for recording. The other three quadrants would be covered by a travelling masking device or perhaps a binary SLM that only allowed light to be transmitted through one quadrant. For the first depth plane in a series, the image would be recorded on one quadrant. For the next depth plane, a different quadrant would be exposed. The quadrant rotation would proceed until all depth planes were exposed. The increase in the spectrum of angles that are recorded in each quadrant could possibly combine to increase the aperture of the thermoplastic. To achieve a six inch holographic viewing volume, a magnification factor of only 2 or 3 would be necessary. This would reduce the burden on the display optics to magnify the output image; low-f/# optics would not be required. An additional benefit of the FEL configuration would be the capability to reduce the exposure per unit area on the thermoplastic. In effect, using the quadrant multiplexing, each quadrant would only receive one-fourth of the exposure energy.

Three experimental goals were set for testing with the phase conjugate recording system:

- (1) Determine the baseline image quality of the phase conjugate recording system;
- (2) Investigative the ability of the various phase conjugate recording configurations to increase the field-of-view of the HGS display and the magnification of the thermoplastic image at the viewing aperture (e.g., use of a fly's eye lens as well as use of a Fresnel field lens instead of the TV projection lens); and
- (3) Perform preliminary experiments focusing on use of a Sharp SLM as the object source for the hologram. The Sharp SLM is monochromatic unit based on TFT technology. It is 2.4 inches wide by 1.8 inches high having a pixel resolution of 384 x 240 pixels and a 30:1 contrast ratio.

5.5.1 EXPERIMENTATION

Experimentation with the phase conjugate configuration included:

- Single exposure of the thermoplastic using a black glass plate with a bright square annulus as the holographic object source,

- Single exposure of the thermoplastic using the glass plate as the holographic object source along with replacement of the TV projection lens with a Fresnel lens of a shorter focal length.
- Single exposure of the thermoplastic using the glass plate as the holographic object source along with positioning of a fly's eye lens between the projection lens and the thermoplastic camera.
- Single exposure of the thermoplastic using the glass plate as the holographic object source along with use of a fly's eye lens and the Fresnel field lens (instead of the TV projection lens).
- A series of single exposures of a Sharp SLM onto the thermoplastic to determine the baseline performance associated with use of this alternative SLM as the holographic object source.

5.5.2 RESULTS

Testing with the phase conjugate optical configuration indicated that this alternative is superior over the baseline configuration for several reasons:

- Provides for distortion-free imaging.
- Eliminates the need for a relay lens.
- Generates a holographic image that is at least comparable to, if not better, than the image obtained with the baseline optical setup.

For the single exposure of the glass plate, using the optimum K ratio and power density as determined during baseline testing, a sharp and bright hologram was generated with little noise and no apparent image distortion. Plate 1 of Figure 5-6 presents a photograph of this hologram.

The image quality and image magnification possible with use of a shorter focal length lens in the optical system was determined. The 4.5-inch-focal-length TV projection lens (L) was replaced with a 3.0-inch-focal-length Fresnel lens that enabled a higher magnification in the output image. Again a hologram of the bright square annulus on the black glass plate was generated (see Plate 2 of Figure 5-6). While enlarged by approximately 30%, noise in the image was evident in the recording. This is attributable to the use of the Fresnel lens which produces scatter. While the phase conjugate system can eliminate distortions, scatter cannot be perfectly compensated for by the system.

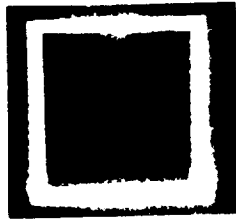


PLATE 1

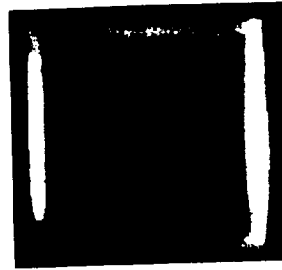


PLATE 2

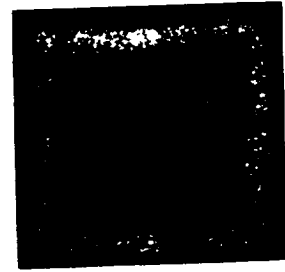


PLATE 3

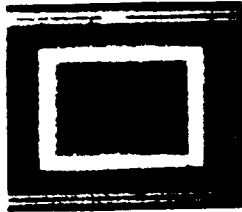


PLATE 4



PLATE 5



PLATE 6



PLATE 7



PLATE 8

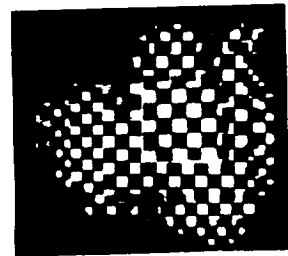


PLATE 9



PLATE 10

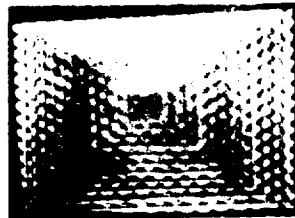


PLATE 11



PLATE 12

Figure 5-6. Photodocumentation: Holograms Recorded on Thermoplastic

To determine the image quality possible with placement of a fly's eye lens in the optical path, another hologram of the black glass plate with the open square annulus was generated. The FEL was placed between the TV projection lens (L) and the thermoplastic camera (M). The FEL was placed as close as possible to the thermoplastic camera without interfering with the reference beam wavefront that approached the camera at a 24° angle. A poor hologram recording resulted. Areas of the hologram containing image information were very noisy even though the background on the plate had little noise.

In an attempt to improve the image quality of the FEL recording, the TV projection lens (L) was replaced with the 3.0-inch-focal-length Fresnel lens and was used in conjunction with the FEL to generate another hologram of the glass plate. Image quality improved with use of the Fresnel lens (see Plate 3 of Figure 5-6). However there was still considerable noise in the image of the glass plate annulus compared to the cases without use of the FEL (i.e., Plates 1 and 2).

It is possible that performance using a FEL can be improved if a high quality FEL with very small lenslets is employed. It was concluded, however, that further investigation into the use of the FEL was not warranted. Image quality will probably not approach the quality achieved without a FEL. Furthermore, accurate collimation and alignment during the phase conjugate reconstruction is more critical when a FEL is used and will result in a more complicated system. Therefore, the FEL would only be considered at a later date if the multiple-SLM approach to circumvent the multiple exposure limitations of the thermoplastic was not successful.

Testing was also conducted using the Sharp SLM as the object source. An image of a rectangle approximately 1.5 inches wide and 1 inch high was transmitted to the SLM. Two holograms were generated: one of a bright rectangle on a dark background and another of a dark rectangle on a bright background (see Plates 4 and 5 of Figure 5-6). For the bright-on-dark case, the pixels forming the rectangle were open to let the laser light pass through while the other pixels were closed. For the dark-on-bright case, all pixels were open with the exception of the middle rectangular annulus. A single exposure was recorded for each case. Good contrast was found for both cases. The holograms were clear and sharp with a high

level of contrast and very little noise. The results were significantly better than those obtained using the nView SLM as the object source.

To test the resolution of the Sharp SLM and the thermoplastic, three other holograms were generated. A video image of a 35 mm slide of the Air Force resolution test chart was transmitted to the SLM (see Plate 6 of Figure 5-6). A clear, sharp hologram was generated with the detailed structure comparable to the video image source. Also, holograms of complex images were generated. Specifically, 2-D video images of a horse and of a cat containing 256 gray levels were made (see Plates 7 and 8 of Figure 5-6). Good results were obtained with each hologram exhibiting enough contrast so that fine details could be discerned in the image.

The results of initial testing with the Sharp SLMs indicate that the units have adequate resolution and contrast for acceptable imaging for the HERSS laboratory development effort. The active area of the Sharp SLMs, however, is too small for the ultimate system if a full six inch cubic holographic image is to be achieved. While a six inch holographic depth can be achieved, the image is limited to the width and height of the Sharp SLM (2.4" x 1.8").

Furthermore, in performing experimentation, a brightness variation in the holographic image on the thermoplastic was noted. That is, brightness varied as a function of viewing angle of the thermoplastic film, and was particularly pronounced in the horizontal viewing plane. It was concluded that there is a fundamental spatial frequency recording limitation of the thermoplastic film across the horizontal plane. Specifically, a diffraction efficiency rolloff produces excessive image brightness variation across the viewing field. Spatial frequencies that are recorded at the optimum resolution of 800 lpm will exhibit the optimum diffraction efficiency and will be bright and have high contrast. This optimum spatial frequency is found only for normally incident rays. The recording of the rays that are either to the left or right of normal will have a higher or lower spatial frequency, a corresponding lower diffraction efficiency, and a corresponding shift in brightness.

The result is that a maximum viewing field angle of only 15-20 degrees can be achieved with the thermoplastic camera. This fundamental problem can only be overcome with a complicated filtering technique to compensate for the non-linear brightness. Alternative

recording mediums with a much higher resolution would also be investigated, such as the Du Pont photopolymer.

5.6 MULTIPLANE-BY-MULTIPLANE EXPOSURE METHOD

With the originally proposed plane-by-plane exposure method, approximately 20 2-D depth planes of information could be recorded on the thermoplastic before the holographic image started to seriously degrade. Within a six-inch depth (152.4 mm), 20 depth planes represents a display resolution of 0.3 inch (7.6 mm). The HGS depth resolution goal was a resolution between 0.08 inch (2 mm) and 0.12 inch (3 mm). Thus, to meet this design goal, between 51 and 76 2-D depth planes had to be recorded. To circumvent the multiple exposure limitation of the thermoplastic and to record a greater number of 2-D depth planes in one hologram, a multiplane-by-multiplane exposure method was devised. In addition to increasing the information content within one holographic frame, another potential advantage of the method is that the entire volume could be recorded in less time. For example, moving the micropositioning table between every 2-D plane exposure would no longer be necessary. A single move of the micropositioning table would now cover a multiplane exposure.

The multiplane-by-multiplane exposure method employs a stack of three SLMs. The object (J) in Figure 5-5 is replaced with a stack of three SLMs that are sandwiched together. In addition, a polarizer precedes the stack and an analyzer follows the stack to allow the appropriately polarized light to enter and leave the stack. For the laboratory prototype, each SLM in the stack was separated by a distance of 0.12 inch (3 mm). The housing on each SLM unit prevented the units from being positioned closer than this distance. For each exposure, three 2-D depth planes of information are sent to the 3-SLM stack for simultaneous exposure. Between exposures, the 3-SLM stack, which is mounted on the micropositioning table, is moved through space so that the images of the next three 2-D depth planes are appropriately positioned within the holographic volume. If 20 multiple exposures could be achieved within in the six-inch volume, then 60 depth planes of information would be recorded and the HGS design goal could be met.

To elaborate upon this method, after the object beam passes through the diffuser (I in Figure 5-5), the laser light will strike the polarizer to allow only light polarized at a 45° angle to enter the SLM stack. Following the polarizer, light is transmitted through cells that have been opened in the SLM stack. Light is not transmitted through opaque or closed cells. After the light is transmitted through the SLM stack, it must also pass through an analyzer. Again, only light with a 45° polarization is allowed to pass, thereby eliminating any stray reflections or light that can add noise to the holographic image.

In the 3-SLM exposure method, the holographic recording film must have a clear view of each image pixel in the stack. This means that pixels ahead and in back of image pixels must be open. A simple way of accomplishing this is to set image-containing pixels to an opaque state and all other pixels to full open. This would create a dark image on a bright background. This approach was not pursued, however, because the non-image light could degrade the multiple exposure recording capability of the film. Instead, the actual implementation required that image-containing pixels be set to an open state. All other pixels were closed except those that provided a clear view of image pixels. Thus, a bright image on a dark background was created.

In summary, in the 3-SLM stack, open SLM cells serve three purposes:

- (1) Represent image information for a particular depth plane. In this case, the DIPS shape function is used to determine what image information is associated with a depth plane. Pixels where image information exists will be opened thereby creating a bright image on a dark background. These pixels, however, will not be set to full open. Instead, the DIPS brightness function is used to determine the grey scale value for each image-containing pixel. Depending on the specific grey scale value, image-containing pixels are set somewhere between full off and full on.
- (2) Allow light to be transmitted to corresponding image-containing pixels on the SLMs that follow in the stack. These pixels are set to full open.
- (3) Allow image information on preceding SLMs to be transmitted through the stack on the path to the thermoplastic plate. These pixels are also set to full open.

Three experimental goals were set for testing with the multiplane-by-multiplane exposure method:

- (1) Determine the baseline image quality of the technique with single exposure of a stack of glass plates;
- (2) Determine the baseline image quality with single exposure of one 3-SLM stack; and
- (3) Determine image quality with multiple exposure of the 3-SLM stack. Identify the maximum number of 2-D image planes that can be recorded in one holographic frame.

It is noteworthy that a single exposure of a glass plate stack was used to determine the optimum performance possible using the stacking method. First, each of the four glass plates in the stack was large (6 inches by 6 inches compared to the smaller 2.8 x 1.7 inch Sharp SLM format). Second, a high contrast checkerboard pattern, printed on an acetate sheet, was attached to each glass plate (allowing for an approximate 1000:1 contrast compared to the 30:1 contrast of the Sharp SLM). These conditions would mimic the best performance possible if an SLM with a large format and a high contrast ratio was available. It is expected that increased SLM size and higher contrast will become available as the technology rapidly develops.

5.6.1 EXPERIMENTATION

Experimentation with the multiplane-by-multiplane exposure method included:

- Single exposure of a glass plate stack,
- Single exposure of the 3-SLM stack,
- Multiple exposure of the 3-SLM stack.

For the baseline experiment with the glass stack, a hologram of a stack of four glass plates was generated. Each glass plate contained a square annulus of a checkerboard pattern. The square annulus on each plate was of increasing size.

To implement the 3-SLM multiplane exposure technique, the three Sharp SLMs were stacked adjacent to each other. An effort was made to align the individual pixel elements across the SLMs as closely as possible. That is, the relative horizontal and vertical positions of each SLM was visually adjusted so that the honeycomb grid of electronic components was aligned. Furthermore, in order to implement the HGS stacking method, it was necessary to invert the intensity of the image on one of the SLMs to account for variations in the polarization coatings among the SLM units. This adjustment allowed the light of the object beam to be transmitted through the entire SLM stack. The horizontal ordering of pixels was also inverted for one of the SLMs so that all three image planes were rectified. These changes were necessary because the SharpVision Video Projection system used an optical combining technique to superimpose the images of each SLM instead of a stacking technique. Due to the incompatibilities, the HGS multiplane stacking technique required the development of a custom SLM addressing method.

For experimental testing with the 3-SLM stack, holograms were generated using a 3-D pyramidal shape. Specifically, a numerical database of a pyramid with a checkerboard surface pattern was used. To generate a hologram, the numerical database was "sliced" into a set of annular depth planes of varying thickness and size. During a particular exposure, three concentric image slices were sent to each of the three SLMs in the stack. The stack was moved to a new position for the next exposure. In a 10-exposure hologram of a 3-SLM stack, 30 annular slices of the pyramid were recorded. For a 20-exposure hologram, 60 annular slices were recorded. Each annular slice was of increasing length and width so that a pyramidal shape was formed through depth.

Based upon previous optimal experimental results using the thermoplastic camera, a K ratio of one and an exposure energy of $70 \mu\text{J}/\text{cm}^2$ were utilized.

5.6.2 RESULTS

For the baseline experiment with the glass stack, a clear, sharp hologram of the checkerboard pattern was generated as illustrated in Plate 9 of Figure 5-6. The four annular slices were clearly visible through the holographic depth as the observer's head moved around the hologram. For some viewers, when the hologram was viewed from only one vantage point, it was difficult to perceive the depth in the image. That is, the checkerboard appeared to

be in only one plane. It is possible that this illusion was the result of the use of the checkerboard pattern. The continuous pattern across the retina could have been misinterpreted as being in one plane. When the viewer's head moved, the depth reappeared. This demonstrates the importance of motion parallax information to resolve ambiguous retinal images.

Multiplane exposure testing commenced with generation of a single-exposure hologram of the 3-SLM stack. For this initial case, the 3-D pyramid was sliced into three depth planes of equal width. Each depth plane image was transmitted to each of the three SLMs and a single exposure hologram was generated. A clear, low noise hologram resulted. However, perceiving depth within the 3-SLM stack was difficult. This was attributed to the use of checkerboard pattern itself.

Multiple exposure testing with the 3-SLM stack followed. Plates 10, 11 and 12 of Figure 5-6 illustrate the results for 7, 13, and 21 multiple exposures of the 3-SLM stack, respectively. These plates represent recordings of 21, 39, and 63 image planes. For the 7-exposure hologram (21 planes), the hologram was clear and sharp. For the 13-exposure hologram (39 planes), the image quality degraded — image brightness decreased and background noise increased. For the 21-exposure hologram (63 planes), the noise reached an unacceptable level. It appears that approximately 13 exposures can be made before the image seriously degrades. Compared to the baseline plane-by-plane exposure method, this represents an 95% increase in the number of image planes that can be recorded. In the baseline condition, 20 exposures (20 image planes) could be recorded before the noise started to seriously affect image quality. In the multiplane-by-multiplane exposure method, 13 exposures (39 image planes) could be recorded.

When viewing these multiplane exposure holograms, depth positions were clearly discernable from one 3-SLM stack position to the next. However, it was not possible to discern differences in depth within an SLM stack even when the observer's head was moved. It is unlikely that this effect could be attributed to the checkerboard pattern alone. Alternatively, this effect could be related to the field-of-view of a single pixel. As previously noted, in order for the light from an image-containing pixel to reach the thermoplastic, the corresponding pixel

on an adjacent SLM is set to full open. Because the light transmitted through the SLM is diffuse (i.e., projected through a diffuser), a cone angle exists through which the modulated light travels. Accordingly, the corresponding areas on adjacent SLMs should be somewhat larger in area than just a single pixel. Currently, because only one pixel is opened on an adjacent SLM, the cone of light emanating from the image-containing pixel would thus be partially occluded. Limitations in computer power precluded testing the effect of opening a greater number of pixels on adjacent SLMs to allow the image-containing cone of light to be fully transmitted through the stack.

Finally, in experimenting with the 3-SLM stack, a faint Moire pattern could be seen in the background of the holograms when the observer's head was moved through the field of view. The moire pattern moved across (and behind) the image field. This problem is a result of stacking the SLMs. A Moire' pattern results from the superposition of multiple (two or more) periodically opaque surfaces with coherent or incoherent light in the stack. Because the resultant spatial intensity distribution is a product, the individual periodic patterns will beat with one another, and a product pattern will result, usually with a much reduced periodicity. This causes some obscuration of the desired intensity pattern. The periodic structure of the SLMs which causes this problem is in the pixel structure itself, i.e., the thin opaque (5-10 micron) electronic structures which separate the pixels.

5.7 HGS ALTERNATIVE RECORDING MATERIAL: DU PONT PHOTOPOLYMER

As noted in the preceding sections, several limitations were encountered with use of the thermoplastic material. The most critical limitation concerns the multiple exposure capability of the material. Using the baseline HGS optical design, only 20 image planes could be exposed before serious image degradation occurred. Even with the multiplane-by-multiplane exposure method, the thermoplastic could only hold 13 exposures (representing 39 image planes) before image quality deteriorated. This information content is still short of the 51 image planes needed to meet a 3 mm depth resolution goal. A second critical limitation associated with the thermoplastic is the limited size of the plate — a 35 mm square. Considering that a magnification factor of three can be achieved, this means that the transverse size of the final holographic image can only be 105 mm or approximately four inches wide

instead of six inches wide. Finally, the holographic image exhibits an image brightness variation across the plate.

For these reasons, hologram generation using a newly released Du Pont photopolymer (HRF-700 Series) was undertaken. The photopolymer is a volume recording material having a resolution of over 4000 lpm. This means that the photopolymer can hold much more information than the thermoplastic. Furthermore, the material is available in sheet sizes 8.5 by 11 inches. This reduces the requirement for magnifying the final holographic image. Finally, the HRF-700 material has a lower film speed than the HRF-352 material previously used during the preliminary experimental studies in Colorado. This meant that the exposure time could be lowered using the new material.

Single and multiple exposure testing with the photopolymer was conducted to determine the image quality and exposure duration of the Du Pont material compared to the thermoplastic.

5.7.1 EXPERIMENTATION

Experimentation with the Du Pont photopolymer included:

- Single exposure of one glass plate square annulus as the holographic object source,
- Single exposure of a stack of four glass plates as the holographic object source,
- Single exposure of one SharpVision SLM as the holographic object source,
- Multiple exposure of one SharpVision SLM as the holographic object source,

For testing with the photopolymer, the thermoplastic camera (M in Figure 5-5) was removed and replaced with a lens mount for holding a photopolymer plate. To create a plate, the photopolymer was cut into approximately 3 inch squares and sandwiched between two pieces of glass that were also approximately 3 inch square. The glass plates were held together by masking tape and were kept in a light-tight box prior to exposure. Following exposure, the laboratory room lights were turned on so that the material was bathed with ultraviolet (UV) light. The UV bath served to instantly complete the polymerization process and "fix" the

display. It should be noted that the glass used in making the photopolymer plates were of low quality and caused some aberrations in the final imagery.

When generating multiple exposure holograms using the Sharp SLM, a video image of a magazine advertisement was used as the underlying data base. The message read: "If you thought Digital wasn't committed to UNIX-based RISC computing". A custom software program was used to cut the single 2-D image into a set of concentric square annuli. The number of annuli in the set varied depending on the desired number of multiple exposures. Thus, to generate a hologram, each annulus was sequentially transmitted to the SLM for exposure. Following exposure, the SLM was moved in longitudinal space, the next annulus in the set was transmitted to the SLM for exposure. This transmit-expose-move cycle was repeated until the final holographic image was built. In effect, the final image contained the text message warped over a 3-D pyramidal shape.

The experiments to evaluate the HRF-700 material followed the same pattern used in the evaluation of the thermoplastic. First, baseline performance, including optimum exposure energy and K ratio, was established using a single glass plate and a stack of four glass plates as the holographic object source. Next, a single-exposure hologram of one SLM was generated. Multiple exposures of one SLM followed in order to determine the maximum number of plane-by-plane exposures possible with the material. Due to contract resource constraints, the latter experimental test was not completed. This also meant that testing with the multiplane-by-multiplane exposure method was not conducted using the photopolymer as the recording material.

5.7.2 RESULTS

As a baseline experiment, a single exposure hologram of a glass plate with a square annulus was generated. As shown in Plate 1 of Figure 5-7, a clear, bright hologram was created using a K ratio of 1 and an exposure energy of 30 mJ/cm². This K ratio and exposure energy were then used to generate a hologram of a four-plate glass stack. One plate contained a 35 mm slide of the Air Force resolution test chart. Another plate contained a square checkerboard annulus. A third plate contained a dot at each corner; a fourth plate contained a

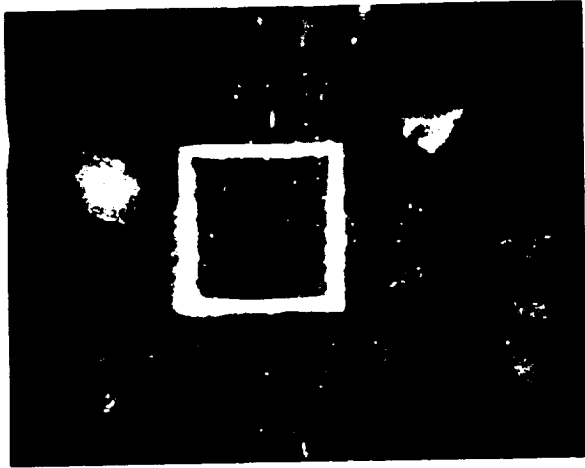


PLATE 1

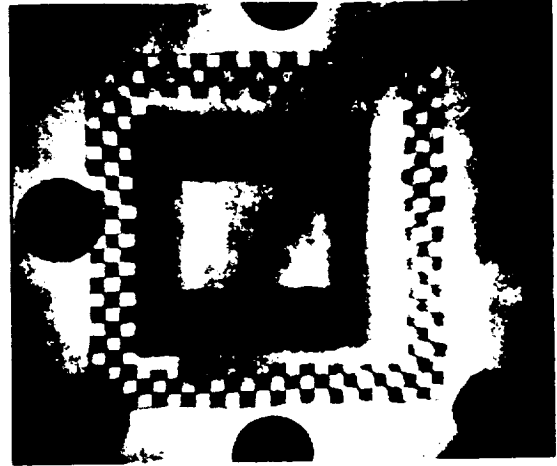


PLATE 2

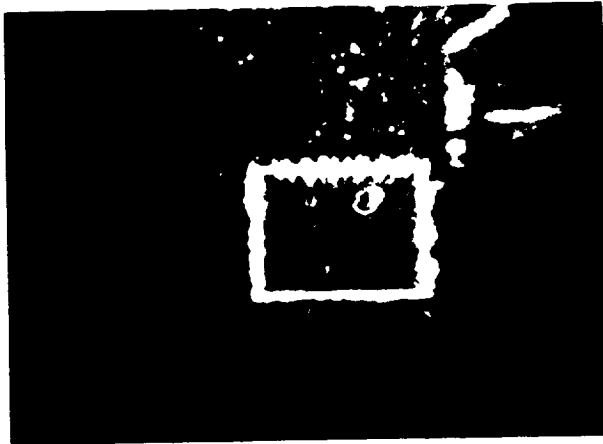


PLATE 3



PLATE 4

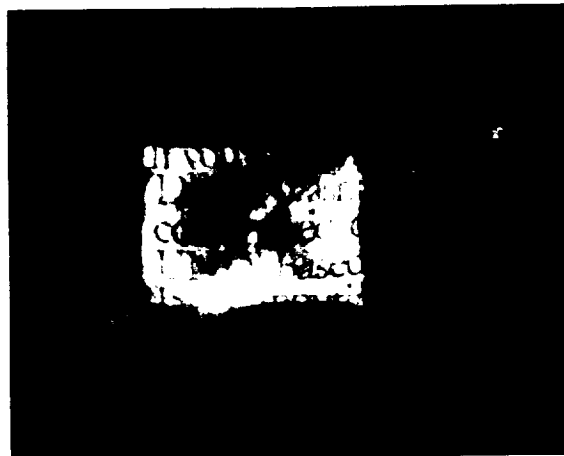


PLATE 5

Figure 5-7. Photodocumentation: Holograms Recorded on Dupont Photopolymer

dot at the top, sides, and bottom. The plate size was approximately 6 inches high and wide. A two inch separation distance was set between each plate such that all the plates straddled a six inch depth. As shown in Plate 2 of Figure 5-7, a clear, bright hologram was generated.

Experimental testing with the Sharp SLM as the object source was then conducted. The first step was generation of a single exposure hologram of one SLM. A single square annulus of the text message was transmitted to the SLM for this purpose. As shown in Plate 3 of Figure 5-7, a good clear hologram resulted. The image was not as bright as what was obtained with the glass plates, but this was expected considering that the contrast ratio is considerably less using the SLM as the test object compared to using the glass plates.

The next experimental step was exploration of the multiple exposure characteristics of the photopolymer. A K ratio of 1 and an exposure intensity of 30 mJ/cm² were used as the initial recording parameters. Using these parameters for a 10 exposure hologram, a sharp but dim hologram was generated. With further testing it was determined that the intensity for multiple exposures should be less than the intensity for a single exposure. For the 20-exposure case, the optimum energy was one-half the recommended power. As illustrated in Plate 4 of Figure 5-7, a bright clear hologram was generated for the 20-exposure case using a K ratio of 1 and an exposure intensity of 15 mJ/cm². Furthermore, unlike the thermoplastic, the image brightness was constant throughout the field-of-view.

The optimum recording parameters changed for the 40-exposure case. Here, the best results were obtained with only one-third the recommended intensity or 10 mJ/cm². Furthermore, the K ratio was optimally set to less than one. That is, the reference beam power was set to a lower energy level than the object beam. Plate 5 of Figure 5-7 presents the results of the 40-exposure hologram. A clear hologram is produced even though the image is somewhat dimmer than the 20-exposure case (Plate 4).

A 60-exposure hologram was attempted. Based on the results of the 10-and 20-exposure cases, the energy was set to less than half the recommended intensity for a single exposure. Several energy levels and K ratios were tested. In each case, no image was recorded. In subsequent communications with Du Pont engineers, it was determined that the

energy for a single exposure (1/60th of the total) did not meet a minimum energy level required to activate the polymerization process. For future testing, it was recommended that before multiple exposure begins, the plate should be bathed with the reference beam for a brief period until the threshold is surpassed (Steven Macara, personal communications, 1990).

5.8 PERFORMANCE OF THE HGS CONFIGURATIONS

To meet NASA performance goals, a myriad of factors were considered in the design of the HGS system. Many tradeoffs existed between the object size, the recording material's size and sensitivity threshold, the laser power and exposure time required to generate a holographic frame, the optical component parameters, and the size and quality of the holographic image. For a near-real-time display, it is most desirable that the recording material be small so that a greater intensity of light can be incident on any given point. With a larger recording material, the light energy is spread across a larger area resulting in less intense light at any given point. With less intensity at any given point, a longer exposure time is needed to reach the light sensitivity threshold of the recording medium. Thus, assuming a fixed laser energy, a hologram can be more quickly generated with a small recording material. However, a small recording material means that the image must be magnified to produce the desired display size.

Achieving a large magnification factor is a challenging problem for the optical designer. To achieve a reasonable magnification of the image, a low $f/\#$ projection lens with a relatively large aperture (i.e., diameter) must be used. The diameter of the projection lens affects the size of the exit pupil; the $f/\#$ (focal length / diameter) affects the magnification of the image within the exit pupil. The problem is that large magnification factors require large-diameter, low- $f/\#$ optics that, in turn, can contribute serious distortions to the image.

For the baseline optical configuration, with the knowledge that the thermoplastic recording material would be approximately 1.5 inches wide (the Newport material was 1.5 inches wide while the alternative Micraudel material was 2.0 inches wide) and with the knowledge that the desired holographic volume was a 6.0-inch cube, a magnification factor of three was provided by the optical configuration. A low- $f/\#$ projection lens ($f/1$) was a key

element in providing the 3x magnification needed to magnify a 2.0-inch holographic image to a 6.0-inch-diameter volume.

During preliminary proof-of-concept testing at the University of Colorado, a full-parallax, multiple-exposure, holographic image was generated using the baseline optical configuration. At that time, thermoplastic-recording-material tests indicated that the Newport camera exhibited superior performance over the Micraudel camera for the multiple exposure application. However, because the Newport material was only a 1.5-inch format, a magnification factor of three limited the holographic volume to 4.5 inches wide by 4.5 inches high. A 6.0-inch display depth could easily be created by moving the projection through an approximate 2.0-inch space to achieve the required 6.0-inch-depth magnification.

Furthermore, during the preliminary testing, several other limitations with the baseline system were noted. The most critical limitation was that less than 30 multiple exposures could be recorded on the thermoplastic. Because the method of exposure for the baseline system was plane-by-plane, this meant that less than 30 depth planes could be recorded in one frame. Thus, a resolution of 5 mm (at best) could be achieved over the 152.4 mm (6.0-inch) depth. If the depth resolution goal of 3 mm was to be met, 51 planes had to be recorded within the holographic volume.

Another limitation of the baseline configuration was that the image of the object varied in size as the micropositioning table moved the projection lens to the next depth plane position between exposures. This meant that the eventual display system would need to adjust the size of each image before displaying it on the SLM to rectify the image planes. This would increase the computational load and possibly the time to generate a holographic frame. In addition, there were some distortions found in the final imagery relating to the 3x magnification of the holographic image as can be expected with low-f/# optics. Finally, the field-of-view of the image was limited to approximately 20°.

A phase conjugate recording configuration was developed to address the limitations of the baseline system. The revised configuration was based on the principles of phase conjugation and reverse ray tracing. Using these principles, two limitations of the baseline

system could be overcome. First, image distortions induced by the optics during hologram generation were removed during the phase conjugate reconstruction. Second, the images within the hologram were presented at the same scale as the original object, thereby eliminating the need to use a computer algorithm to adjust the size of the image transmitted to each SLM prior to exposure. Because the scale of the object remained constant, however, this placed an additional constraint on the system design. To obtain a 6.0-inch holographic volume, the object also had to be 6.0 inches wide and high. For this reason, the nView SLM (approximately 10 inches wide and high) was selected as the hologram object source.

Testing with the phase conjugate optical configuration with a glass-plate test object confirmed that this alternative was superior to the baseline for several reasons:

- Distortion-free imaging was possible.
- The phase conjugate configuration required only one large-aperture, low-f/# lens. The baseline required two such lenses (one for hologram generation and one for hologram viewing). The phase conjugate system used the same large-aperture, low-f/# lens for hologram generation and viewing by reversing the path of light through the optics during image reconstruction.
- In terms of image brightness and clarity, holograms generated with the phase conjugate configuration were at least comparable to, if not better, than the images obtained with the baseline.

Another major advantage of the phase conjugate recording system was the opportunity to increase the image magnification relative to what was achieved with the baseline to meet the 6.0-inch display dimension goal. Specifically, use of a projection lens with a lower f/# could be attempted to increase the magnification of the 1.5-inch thermoplastic greater than threefold. The image distortion associated with a lower f/# lens could not be tolerated in the baseline configuration. The phase conjugate system could eliminate optical distortions. A 3.0-inch-focal-length Fresnel lens replaced a 4.5 inch-focal-length TV projection lens. The resultant hologram image was noisy. While the optical distortion was eliminated, the light scatter induced by the Fresnel lens could not be perfectly compensated. This configuration was not pursued further.

The phase conjugate system also provided the opportunity to test the use of a fly's eye lens to increase the field-of-view of the holographic image. This lens could be placed in the optical path between the object and the thermoplastic recorder. Light at the edge of the cone of light travelling to the thermoplastic that would normally be lost could be captured by this lens. That is, the fly's eye lens provided the opportunity to capture this light and redirect it to the thermoplastic, thereby increasing the field-of-view in the recorded image. Again, use of this lens would not have been possible with the baseline due to image distortion.

Another potential use of the fly's eye lens was in conjunction with a quadrant multiplexing scheme that could "effectively" increase the size of the thermoplastic recording aperture and reduce the exposure per unit area. Increasing the "effective" aperture from 1.5-inch to a 2- or 3-inch area would mean that a six-inch holographic volume could be achieved with just a magnification factor of three. Furthermore, by exposing only one quadrant at a time, the exposure bias buildup could be avoided. This could increase the multiple exposure capacity of the thermoplastic. Such a multiplexing technique could serve as a backup method to an alternate multiplane-by-multiplane exposure method.

Hologram generation with the fly's eye lens, however, resulted in a poor image recording. Areas of the hologram containing image information were very noisy even though the background on the plate had little noise. While it is possible that performance can be improved with a higher quality fly's eye lens, it was concluded that further investigation was not warranted. Regardless of any potential benefit this lens could offer, image quality was too severely affected to justify further development efforts.

Holograms generated with the nView SLM were extremely noisy. It was concluded that due to an inadequate contrast ratio, this device was unsuitable as the object source for a hologram. The noise in the holographic image was attributed to the considerable light leakage through the nView pixel cells that were set to an opaque state. As an alternative, a Sharp SLM with a higher contrast ratio was successfully used as the object source. A disadvantage of the Sharp SLM, however, was that it is only currently available in a small size format — 2.4 inches wide by 1.8 inches high — meaning that the final holographic image is limited to those same transverse dimensions. The six-inch deep display can still be attained by longitudinally

repositioning the SLM within a six-inch depth during hologram generation. The 6.0-inch transverse dimension can only be achieved when a 6 inch by 6 inch SLM becomes available with a suitable contrast ratio. Other larger format Sharp SLMs are currently under development (e.g., Adachi, Matsumoto, Nagashima, et al., 1990).

To circumvent the multiple exposure limitation of the thermoplastic, a multiplane-by-multiplane exposure technique was devised and implemented with the phase conjugate configuration. Using the Sharp SLMs, this technique employed the use of a 3-SLM stack. At each exposure of the thermoplastic, three depth planes were simultaneously recorded. Thus, with 13 exposures, 51 image planes could be recorded to meet the depth resolution goal of 3 mm. Another key benefit of this technique was the reduction of time to generate a single holographic frame. If 51 images were recorded in 13 exposures instead of 51 exposures, then 38 micropositioning-table moves could be eliminated. This would be a significant savings of time considering that table move and settle times were found to be 250 msec and 1000 msec, respectively. The settle time was necessary in order to allow vibrations induced by the table move to dissipate.

The viability of the 3-SLM exposure scheme was demonstrated through experimentation. Seven, 13, and 21 exposures were made of the stack. This meant that 21, 39, and 63 image planes of information were recorded on the thermoplastic. The 7-exposure hologram (21 planes) was clear and sharp. However, as the number of exposures increased, the image quality degraded; image brightness decreased and noise increased. Furthermore, when viewing a hologram, it was not possible to discern differences in depth within the images formed by one SLM stack (the 3 SLMs are separated by approximately 3 mm). However, depth positions were clearly discernable from one stack position to the next. Finally, there was also a faint Moire pattern in the background of the holograms when the observer's head was moved through the field of view.

The multiple-SLM exposure scheme provides a promising technique for meeting HGS resolution and near-real-time generation goals. However, further research is needed. A possible explanation for the inability to perceive depth within a 3-SLM stack is that the field-of-view of a single pixel is relatively small considering that only one pixel is set to full open on

adjacent SLMs to allow the cone of light to be transmitted through the stack. It may be necessary to open several pixels; this, however could lead to a noisy image. To determine exactly how many pixels should be opened requires further research to fully characterize what factors lead to noise with multiple exposures. Specifically, the contribution of the total light (or image information) within all the planes to the noise must be determined relative to the effect of the number of exposures alone. Furthermore, the faint moire pattern found in holograms generated with the 3-SLM stacking method could possibly be eliminated by aligning the electronic grids of each SLM. In the ultimate system, a custom device having micrometer adjustment controls could be used in the alignment. Finally, psychophysical experimentation is needed to determine what SLM spacing distance and pixel opening method will result in a perceivable separation of image planes.

Another major conclusion of the experimentation is that future HGS research should focus on the use of photopolymer as the recording medium of choice. There are three primary limitations of the thermoplastic. First, the thermoplastic cannot store enough image planes to meet the depth resolution goal. Second, its small 1.5-inch format limits the holographic volume to a 4.5-inch transverse dimension. Third, the viewing field angle is limited to only 15° to 20°. The first and third limitations are related to the 800 lpm resolution possible with the surface-relief thermoplastic material. Alternatively, the Du Pont photopolymer is a volume recording material with a significantly greater resolution of 3000-4000 lpm. Recent developments at Du Pont (HRF-700 series films) led to improvements in the material that allow a higher modulation index and an improved photospeed. Another key advantage of the photopolymer is that it is available in a large format (8.5 x 11 inch sheets). Thus, the photopolymer has the potential to store a greater number of multiple exposures, improve the field-of-view of the final hologram, as well as provide the 2.0-inch format necessary to achieve a 6.0-inch holographic output image.

Some testing with the Du Pont HRF-700 material was conducted. Single and multiple exposures of the photopolymer were made using 6.0-inch square glass plates as well as SLM test objects. Single exposure of one glass plate as well as single exposure of a stack of four glass plates resulted in clear and bright holograms. Multiple, plane-by-plane exposures of a single SLM resulted in clear holograms for 10, 20 and 40 exposure cases. While noise did

increase with the number of exposures, the noise was significantly lower noise than that exhibited with the thermoplastic. The initial results indicate that the photopolymer can store greater information content. Further research with the photopolymer is warranted using the 3-SLM stacking.

To project the ultimate ability of the photopolymer to be used as a near-real-time recording material, one must consider the exposure time required by the material as well as the time to record a minimum of 51 image planes. With the 3-SLM stacking technique, it is possible that one holographic frame with the required resolution can be generated with 13 table moves. Considering that the time for one move-settle cycle is 1250 msec, 13 cycles represent a fixed time of approximately 15 seconds in the generation of one holographic frame. (Image planes can be simultaneously transmitted to the SLMs during the move-settle time if a separate video driver is used to address each SLM in the ultimate system. In the laboratory prototype model, SLM-addressing times were much longer on the order one one minute). Considering use of the argon-ion laser with one-watt power on the 514 line, an exposure time of between 24 and 30 seconds is required for the photopolymer. The UV bath development time is instantaneous. With the current photopolymer and a slightly revised HERSS configuration that can more quickly address the SLMs, it is possible that a holographic frame can be generated in 39 to 45 seconds (15 seconds for the move-settle cycles and 24 to 30 seconds for the exposure).

Concerning the ultimate ability of the photopolymer to produce a holographic frame even faster, Du Pont researchers project that the sensitivity of the photopolymer can be improved by a factor of six. This could mean a reduction in exposure time to 4 or 5 seconds. With this projected improvement, it is possible that a full-parallax holographic frame could be generated in about 20 seconds (15 seconds for the move-settle cycles and 4 or 5 seconds for the exposure).

5.9 SUMMARY

Using the baseline configuration, a full-parallax, multiple-exposure, holographic image was generated on the thermoplastic material using the plane-by-plane exposure technique. Several limitations, however, were noted with this configuration. Three limitations

were associated with the thermoplastic recording material. Two limitations were associated with the optical configuration itself and another was associated with the plane-by-plane recording method. The primary disadvantage of the thermoplastic material was a storage limit of less than 30 multiple exposures. This meant that the resolution achievable with the baseline configuration was 5 mm at best, short of the 3 mm depth resolution goal. Second, the thermoplastic material was limited to a 1.5-inch format. This meant that the threefold image magnification provided by the baseline configuration generated a holographic image that was 4.5 inches wide by 4.5 inch high by 6 inches deep, short of the goal of a 6.0-inch cubic volume. Third, the field-of-view of the image was limited to approximately 20°. This limitation was ultimately attributed to the 800 lpm recording resolution of the thermoplastic. The recording resolution affects the quality of the diffraction grating stored on the material. In order to achieve an increased field-of-view, a material with a higher recording resolution would be necessary.

A major limitation with the baseline optical configuration was that some image distortions were evident in the final hologram. This could be expected with the use of large-aperture, low-f/# lenses. These lenses, however, were necessary to produce a reasonably large magnification of a small recording material. As previously noted, a small recording material is desirable to maximize the intensity of light at any point, thereby reducing the time to deliver sufficient energy to meet the exposure requirements of the recording material. Another limitation of the baseline optical system was that the image of the object varied in size between depth-plane exposures. This meant that the eventual display system would need to adjust the size of each image before displaying it on the SLM to rectify the image planes. This would increase the computational load and possibly the time to generate a holographic frame.

A disadvantage of the plane-by-plane recording method was the time required to generate a single holographic frame having a 3 mm depth resolution. To achieve a depth resolution of 3 mm within the 6.0-inch holographic volume required that 51 image planes be recorded. Before a single image plane could be recorded, however, the micropositioning table had to be moved to the next longitudinal axis location followed by a fixed time period for any vibrations associated with the table move to dissipate. The time to move the table was approximately 250 msec; the time to dissipate any vibrations was approximately 1000 msec.

Considering table move and settle times alone, exposing 51 images would expend 64 seconds. The required exposure time for the thermoplastic material, assuming use of an argon-ion laser with a one-watt energy on the 514 line, would be approximately 1 second. The development time for the Newport thermoplastic camera was approximately 30 seconds. Thus, generation of a holographic frame would be possible in about 95 seconds. These computations, of course, reasonably assume that the SLM can be addressed within the table move-settle time.

A phase conjugate recording configuration was developed to address the limitations of the baseline system. Using these system, the two optical component limitations of the baseline could be overcome. First, image distortions induced by the optics during hologram generation were eliminated. Second, the images within the hologram were presented at the same scale as the original object, thereby eliminating the need to use a computer algorithm to adjust the size of the image transmitted to each SLM prior to exposure. Experimentation with the phase conjugate optical configuration confirmed that distortion-free imaging was possible and that holograms of comparable, if not better, image quality could be produced.

Another major advantage of the phase conjugate recording system was the opportunity to test other optical configurations that could alleviate some of the disadvantages of the thermoplastic material. Specifically, a lower- $f/\#$ projection lens (a Fresnel lens) was employed to increase the image magnification of the thermoplastic image. A fly's eye lens configuration was also employed in an attempt to (1) increase the field-of-view of the holographic image, (2) increase the effective aperture of the thermoplastic material to a 2 or 3 inch diameter so that a magnification factor of three would generate a 6.0-inch holographic image volume; and (3) ultimately increase the multiple exposure capacity of the thermoplastic if used in conjunction with a quadrant multiplexing technique. Testing with these lenses in the baseline configuration would not have been possible due to the severe image distortions associated with their use. The phase conjugate system could eliminate these distortions. Holograms generated with the low- $f/\#$ Fresnel lens, however, produced considerable noise in the image due to uncorrectable light scatter induced by the Fresnel lens. This option was not further pursued. Also, experimentation with the fly's eye lens resulted in a hologram with very poor image quality. Regardless of any potential benefit these lens could offer, further development was not justified because image quality was too severely affected.

A disadvantage of the phase conjugate optical configuration was the strict requirement on the size of the SLM. Specifically, the SLM size had to be at least 6-inches wide and high to meet the transverse dimension goal of the holographic image. For this reason, the nView SLM (approximately 10 inches wide and high) was selected as the hologram object source. Unfortunately, the performance of this device was poor and was deemed unsuitable for the HGS subsystem. In its place, a Sharp SLM with a higher contrast ratio was successfully employed. However, because the Sharp SLM is only commercially available in a small format, the size of the image within the holographic volume is limited to the transverse dimensions of the Sharp unit, 2.4 inches by 1.8 inches. This limitation is expected to be alleviated when larger format SLMs, currently under development, are released. Adachi et al.(1990) report the development of a Sharp SLM with a 5.5 inch diagonal and a 100:1 contrast ratio.

A multiplane-by-multiplane recording method employing a 3-SLM stack was also developed to (1) circumvent the multiple exposure limitations of the thermoplastic and (2) to reduce the time required to expose 51 image planes. Considering table move-settle time alone, the holographic frame update could be reduced by as much as 48 seconds. The viability of this technique was demonstrated through experimentation with the phase conjugate configuration. Compared to the baseline plane-by-plane exposure method, there was a significant improvement in the information content recordable on the thermoplastic. However, it appears that approximately 13 exposures (39 image planes) can be recorded before the image is seriously degraded. Furthermore, while depth positions were clearly discernable from one stack position to the next, it was not possible to discern depth differences within a 3-SLM stack. Finally, there was also a faint Moire pattern in the background due to the misalignment of the electronic grids across SLMs. As previously stated, the multiple-SLM exposure scheme provides a promising technique for meeting HGS resolution and near-real-time generation goals. However, further research is needed to determine the impact of the pixel-opening method within a 3-SLM stack on depth perception. Related research to fully characterize the factors introducing noise in the final holographic image is also necessary.

Another major conclusion of the experimentation is that future HGS research should focus on the use of photopolymer as the recording medium of choice. Critical drawbacks of

the thermoplastic include: (1) Limited multiple-exposure capability; (2) Availability in only a 1.5-inch format that limits the holographic viewing volume to only 4.5 by 4.5 by 6.0 inches; and (3) a 800 lpm recording resolution that limits the viewing field angle to only 15° to 20°. Alternatively, a Du Pont photopolymer (HRF-700 Series) is the preferred recording medium. The photopolymer is a volume recording material with a resolution of 3000-4000 lpm that is available in large formats (8.5 by 11 inches). This material has the potential to store a greater number of multiple exposure with a higher image quality than was possible with the thermoplastic. An improved viewing field angle is also possible with this material.

Preliminary experimentation with the HRF-700 series photopolymer indicated that a high quality image could be recorded with 40 multiple exposures of a single SLM. A drawback of the photopolymer, however, is the slower photospeed compared to the thermoplastic. Exposure time for this material is approximately 24 - 30 seconds using the argon-ion laser; development time is instantaneous with the UV bath. Considering a system employing the 3-SLM exposure method, a holographic frame with 51 image planes can theoretically be recorded in approximately 40 seconds (16 seconds for the 13 move-settle cycles plus 24 seconds for exposure). Du Pont researchers also project that the material photospeed can be improved by a factor of six (Steven Macara, personal communications, 1990). Thus, it is possible that a holographic frame update of 20 seconds (16 seconds for the 13 move-settle cycles plus 4 seconds for exposure) can ultimately be achieved.

In summary, the recommended HGS design incorporates the following:

- Optical recording configuration : Phase conjugate system,
- SLM: Matrix-addressable unit at least 6.0 inches on each side with a contrast ratio at least 30:1.
- Exposure method: Multiplane-by-multiplane employing a 3-SLM stack, and
- Recording Material: Du Pont photopolymer in a 2.0-inch format.

A 6.0 by 6.0 inch SLM is not currently commercially available. It is expected, however, that such a device will be released in 1991.

6. CONCLUSIONS AND RECOMMENDATIONS

NASA design goals for the HERSS 3-D display system, as presented in Sections 1 and 2, include:

1. Build a full-parallax, holographic volume with the dimensions of a 6.0-inch cube,
2. Display the image in near-real time update (30 seconds),
3. Provide a depth resolution of two to three millimeters,
4. Minimize computational complexity,
5. Maximize operator safety and comfort, and
6. Provide a capability for overlaying graphics onto the holographic image.

As part of the HERSS effort, full-parallax holograms were generated by multiply exposing a thermoplastic recording material using a numerically-represented image data base. This indicates the viability of the overall HERSS concept. A major accomplishment was also achieved in the development of a flexibly-designed laboratory that allows evaluation of alternative configurations.

With respect to the NASA design goals for a near-real-time 3-D teleoperations display, the HGS subsystem has made significant progress. While the IAS and DIPS subsystems are unable to meet NASA goals, an alternative technology is now available to perform the IAS/DIPS functions. Specifically, a laser range scanner can be utilized to build the numerical data base needed by HGS of the objects at the remote work site. The strengths and shortcomings of each subsystem are summarized below followed by recommendations for future directions.

Using the HGS subsystem, it has been demonstrated that multiple SLM images can be recorded on a single holographic recording medium. This has been demonstrated with both thermoplastic and photopolymer recording materials. The HERSS recording method simply superimposes multiple 2-D images on the recording material. Complicated techniques for angular or spatial multiplexing are not required. Experiments were performed to characterize how many exposures can be recorded on the thermoplastic. When recording 2-D images in a plane-by-plane fashion, it was found that the thermoplastic could store about 20 images. While hologram were generated with up to 30 image planes, it was concluded that 20 images was a reasonable limit for a relatively good image quality. Twenty images represents a depth resolution of only 7.6 mm.

An innovative 3-SLM-stack recording technique was implemented to circumvent the multiple exposure limitation of the thermoplastic. Another key advantage of this multiplane-by-multiplane recording technique is that a 50% savings in time can be achieved compared to the baseline plane-by-plane recording method. Using the 3-SLM stack, it was found that the thermoplastic could hold about 39 images. For a 6.0-inch-deep (152.4 mm) holographic volume, 39 image recordings are equivalent to an approximate 4 mm resolution. Due to this depth resolution limit, as well as a limited 15° to 20° viewing-field, and an ultimate limit to a 4.5" by 4.5" by 6.0" holographic volume, research with alternative photopolymer recording material was pursued.

Initial testing of a Du Pont photopolymer material (HRF-700 Series) has been accomplished. The recording capabilities of this material are clearly superior to the thermoplastic. Preliminary experimentation indicated that information content can at least be doubled. Indeed, a clear, low-noise hologram was generated following 40 exposures of a single SLM. Future research is necessary to determine the multiple exposure capacity of this material using the 3-SLM stacking technique. Corollary research is also desirable into the refinement of the 3-SLM exposure technique as well as psychophysical research to measure depth perception within the holographic volume. In particular, experimentation with various pixel-opening paradigms should be pursued to determine how much the field-of-view of a single pixel in the final holographic image should be increased. The resultant effect of the alternative configurations on depth perception should simultaneously be pursued. Methods to

reduce the settle-time following a micropositioning table move should also be investigated to reduce the overall time to generate a single holographic frame.

It is projected that with specific technology developments, the NASA design goals for HERSS can be met. First, a larger size SLM is needed to meet the 6.0-inch-cubic display volume. Development of a high contrast ratio (100:1), 5.5-inch diagonal, Sharp SLM has already been reported in the literature (Adachi, et al., 1990). A second critical area for development is in the increase of the photospeed of the photopolymer material. Researchers at Du Pont predict that the photospeed can be improved by a factor of six (Steven Macara, personal communications, 1990).

With these SLM and photopolymer technology improvements, it is predicted that a revised HERSS configuration can easily meet Goal 1 to build a 6.0 inch display volume, Goal 2 to update the image in less than 30 seconds, and Goal 3 to provide a depth resolution of two to three millimeters. Goal 4 can be met with use of the phase conjugate recording configuration which records and reconstructs images at the same scale as the original object. Concerning the fifth goal, the photopolymer can easily be sealed with a mylar covering to avoid the release of any noxious chemicals. Goal 6 is an inherent HERSS capability because the source of the hologram is a numerical representation of the objects at the remote work. Graphic symbology can easily be incorporated into the numerical data base.

Neither the AT nor the non-AT IAS configurations developed during this effort can meet NASA design goals. Although the AT system can meet the long-term 0.1 mm depth resolution goal, it cannot meet the requirement to image a 6.0-inch cubic object space unless a complicated and time-consuming patchworking technique is used. Such a patchworking technique for image acquisition is not feasible for a near-real-time display system. Alternatively, although the non-AT system can meet the six-inch cube imaging requirement, it cannot meet the depth resolution requirements.

In addition, the DIPS algorithms which process the raw image data bases collected by the IAS will require substantial development. The z-contrast algorithm exhibited the best performance, but the output was still very noisy and unacceptable for subsequent 3-D display.

The performance of the z-contrast algorithm could be improved, i.e., by creating a cooperative object with texture or spray paint patterns at the resolution limit of the imaging system. However, substantial research efforts would still be required to determine if the ultimate performance of these algorithms would be suitable for NASA display resolution goals.

In combination, the performance limitations of the IAS and DIPS prototypes lead to the conclusion that this passive method for determining the surface points of a remote object is not feasible for the NASA application. Alternatively, developments in laser scanning technology indicate these devices can acquire an image data base with a depth resolution significantly less than 1 mm over a six-inch cubic area. Laser scanning devices are not generally dependent on specific object characteristics as are the DIPS image restoration algorithms. Thus, the most practical recommendation for collection of a numerical data base for the teleoperation applications is with the use of a laser scanning device. To demonstrate the ease of integrating a laser scanner data base with the HERSS HGS subsystem, a data base was acquired from Servo-Robot of Canada. A holographic image of the data base was recorded using the 3-SLM multiplane exposure technique.

Therefore, in order to meet the six NASA goals, it is recommended that the any future HERSS configuration be modified to include:

- Use of a laser range scanner to collect a 3-D image data base of the remote objects in place of the originally proposed IAS/DIPS subsystems.
- Use of the phase conjugate recording configuration in the HGS.
- Use of a photopolymer as the holographic recording material.
- Use of a 3-SLM stacking technique to multiply expose the recording material.
- Acquisition of a 6.0 inch by 6.0 inch SLM with a high contrast ratio (preferably 100:1) when one becomes commercially available.

Additionally, it is important to experimentally access the ability of teleoperators to optimally utilize holographic displays. It is apparent that the development of any holographic system involves numerous tradeoffs; this experimentation would allow the determination of

which features provide the greatest usability for teleoperators. The types of experimentation which should be pursued include the following:

- Determine the optimum pixel-opening paradigm for the 3-SLM stack.
- Develop and execute psychophysical experiments to determine depth perception within the holographic volume.
- Evaluate the tradeoffs between features such as resolution and image brightness.
- Develop a test condition simulating an actual teleoperations task in order to demonstrate enhanced performance with the use of HERSS.

Our research conducted under this Phase II effort indicates that holographic display systems hold great promise for providing acceptable 3-D display for NASA teleoperation requirements. Alternative 3-D technologies, such as stereography, have not yet been perfected and are unable to meet all of the goals stated above. To date, holographic display techniques have not exhibited characteristics that preclude their usage by a significant portion of the population. For example, stereographic displays exclude at least 5 percent of the population.

A limitation of the ultimate HERSS system, utilizing current and projected technologies, is an update rate of between 20 and 30 seconds. However, it is believed that this relatively slow update rate would still provide a viable 3-D display system for use in teleoperations, particularly when compared with the time required to perform current precise positioning of the remote manipulator arm. Utilizing the multiple views presented on the current set of 2-D television displays for the final precise positioning (i.e., within six inches of the target object) can require two to 20 minutes. Therefore, it is recommended that the above research be pursued in order to transition holography further along the continuum of meeting NASA teleoperation goals.

REFERENCES

- Adachi, M. Matsumoto, T. Nagashima, N., Hishida, T., Morimoto, H., Yasuda, S., Ishii, M., and Awane, K. (1990). A high-resolution TFT-LCD for a high-definition projection TV. Proceedings of the Society for Information Display (May 1990). Playa del Rey, CA: Society for Information Display.
- Caulfield, H.J. (1983). Holographic Display. Report No. NSF/IST-83003, Billerica, MA: Aerodyne Research, Inc.
- Cole, R.E., Pepper, R.L., and Pinz, B.E. (1981). The Influence of Head-Movement Parallax on Perceptual Performance under Direct and TV-Displayed Conditions. Technical Report 678, San Diego, CA: Naval Ocean Systems Center, (AD-A101192).
- Gaynor, E.S., Rhodes, W.T., and Caulfield, H.J. (1987). Exposure compensation for sequential superposition holographic display. Submitted to Applied Optics??
- Hodges, L. and McAllister, D. (1987). True three-dimensional CRT-based displays. Information Display, Vol. 3, No. 5, 18-22.
- Iavecchia, H.P., Arnold, W.K., Gaynor, E.S., Rhodes, W.T., and Rothenheber, E.H., (1987). Holographic Enhanced Remote Sensing System (HERSS); Feasibility Study. Final Technical Report 2083. Willow Grove, PA: Analytics, Inc.
- Iavecchia, H.P., Rhodes, W.T., Rothenheber, E.H., and Janiszewski, J. (1990a). Development of a Prototype Holographic Enhanced Remote Sensing System (HERSS): Quarterly Report for November 1989 to January 1990 (2168-06). Willow Grove, PA: Analytics, Inc.
- Iavecchia, H.P., Rhodes, W.T., Rothenheber, E.H., and Janiszewski, T.J. (1990b). Development of a Prototype Holographic Enhanced Remote Sensing System (HERSS): Auxiliary Document to Quarterly Progress Report 2168-06. Willow Grove, PA: Analytics, Inc.
- Janiszewski, T.J., Iavecchia, H.P., and Mathur, S. (1990). Holographic Enhanced Remote Sensing System: Software Guide. Technical Report 2168-02. Willow Grove, PA: Analytics, Inc.
- Lohmann, A. W. (1987). Parallel interfacing of integrated optics with free-space optics. Optik, Vol. 75.

- Rhodes, W. T., Sitter, D. N., and Rothenheber, E. H. (1989a). Conditions for space-invariant 3-D imaging. Presented at the 1989 Annual Meeting of the Optical Society of America, Orlando, FL, 10 October 1989, paper No. FC1.
- Rhodes, W. T., Sitter, D. N., and Rothenheber, E. H. (1989b). Conditions for space-invariant 3-D imaging. In preparation.
- Sher, L.D. (1990). The SpaceGraphTM Display: Principles of Operation and Application, in W.E. Robbins, *Advances in 3-D Display Technologies*, Seminar Notes M-9 (May 14, 1990), Playa del Rey, CA: Society for Information Display.
- Spangenberg, J. (1989). Phase II Statement of Work (Revised). Analytics Letter dated 25 July 1989.
- Traub, A.C. (1967). Stereoscopic display using rapid varifocal mirror oscillations. *Applied Optics*, Vol. 6, No. 6 (June 1967), 1085-1087.
- Weingartner, I. (1983). Holography — techniques and applications. *J. Phys. E. Sci Instrum.*, Vol 16, 16-23.
- Williams, R.D. and Garcia, F. (1988). A real-time autostereoscopic multiplanar 3-D display system. *Proceedings for the Society of Information Display*, Anaheim, CA, May, 1988.
- Williams, R.D. and Garcia, F. (1989). Volume visualization displays. *Information Display*, Vol 5, No. 4.

APPENDIX A.
HERSS CENTRAL COMPUTER SYSTEM

A. HERSS CENTRAL COMPUTER SYSTEM

This appendix provides a general description of components and functions of the HERSS software and hardware system. More detailed information concerning the use and operation of these components, including user interface documentation, is reported in Janiszewski, Iavecchia, and Mathur (1990).

A.1 COMPUTER SYSTEM FUNCTIONS

Four primary computer control system functions were required for the HERSS project including:

- Control the IAS image acquisition process and data file management,
- Control execution of the DIPS algorithms,
- Provide a graphics capability to assess DIPS algorithm performance, and
- Control the HGS holographic generation process.

Tables A-1 through A-4 list general steps involved in executing each of these primary functions.

Table A-1. Primary Steps for IAS Control

STEP	ACTION
1.	Accept input from the user including the name of the image data base to be generated, the number of 2-D video images to be collected, as well as the stepping distance that the micropositioning table will move between video frame grabs.
2.	Establish files for the current IAS data base acquisition.
3.	Move the micropositioning table to the first (next) position.
4.	Capture the video image (perform A/D video conversion).
5.	Store the image.
6.	Return to Step 3 until the entire data base is collected.

Table A-2. Primary Steps for DIPS Execution

STEP	ACTION
1.	Accept input from the user including the name of the image data base to be processed and the specific DIPS algorithm to be applied to the raw data base (e.g., Frieden, Z-contrast, Hausler).
2.	Establish files for the current IDIPS processing task.
3.	Process raw video images using the algorithm specified by the user.
4.	Store the shape and brightness output files.

Table A-3. Primary Steps for DIPS Image Analysis

STEP	ACTION
1.	Accept input from the user including the name of the data base to be processed, the type of image to be viewed (e.g., DIPS shape or brightness function output, raw IAS video slice, DIPS processed video images).
2.	Load correct video into Facelt display buffer.
3.	Call Facelt to display buffer.

Table A-4. Primary Steps for HGS Control

STEP	ACTION
1.	Accept input from the user including the number of 2-D images to be recorded; the step distance to move the micropositioning table between exposure intervals; and the number of SLMs that should be used in the display process (none, 1, or 3).
2.	Move the micropositioning table to the first (next) position.
3.	Wait for the table to stop and settle.
4.	Send a processed image slice to the SLM(s) for cases when a numerical data base is to be holographically displayed.
5.	Trigger laser shutter to exposure the thermoplastic.
6.	Wait for shutter to close.
7.	Return to Step 2 until all the desired images are recorded.

NOTE: In the holographic recording process, the user must manually set the laser shutter time as well as the thermoplastic controller system to erase and develop the thermoplastic plates.

A.2 HARDWARE

A Macintosh IIx (Mac-IIx) was selected as the HERSS central computer control system. The Mac-IIx utilizes a 68030 processor and a 68882 math co-processor, both running at 16 MHz. The unit is equipped with advanced graphics and special purpose communication boards. A complete list of components in the central computer system follows:

- Mac-IIx, 8 MEG RAM,
- 1.44 MB 3.5 inch floppy drive,
- 19 " Multisync monitor,
- Truevision NuVista 32-bit color image capture and display board,
- Truevision VIDI/O RGB to composite video converter,
- 300 MEG SCSI hard drive,
- 60 MEG tape backup unit,
- National Instruments Parallel I/O card, and an
- Apple 8-bit color display card

Figure A-1 illustrates the HERSS hardware architecture. The Mac-IIx system is the main controlling microcomputer.

The NuVista 32-bit color image capture board is used to acquire images from the CIDTECH camera. This image capture operation is done within the main HERSS software at a resolution of 512 x 486, with 256 gray level intensity variations. The 300 MEG hard drive is used to save the large databases created by acquiring video images. The NuVista card also has the capability of displaying 32-bit color graphics. This card is utilized for data analysis functions and for displaying information on the three Sharp SLMs during hologram generation.

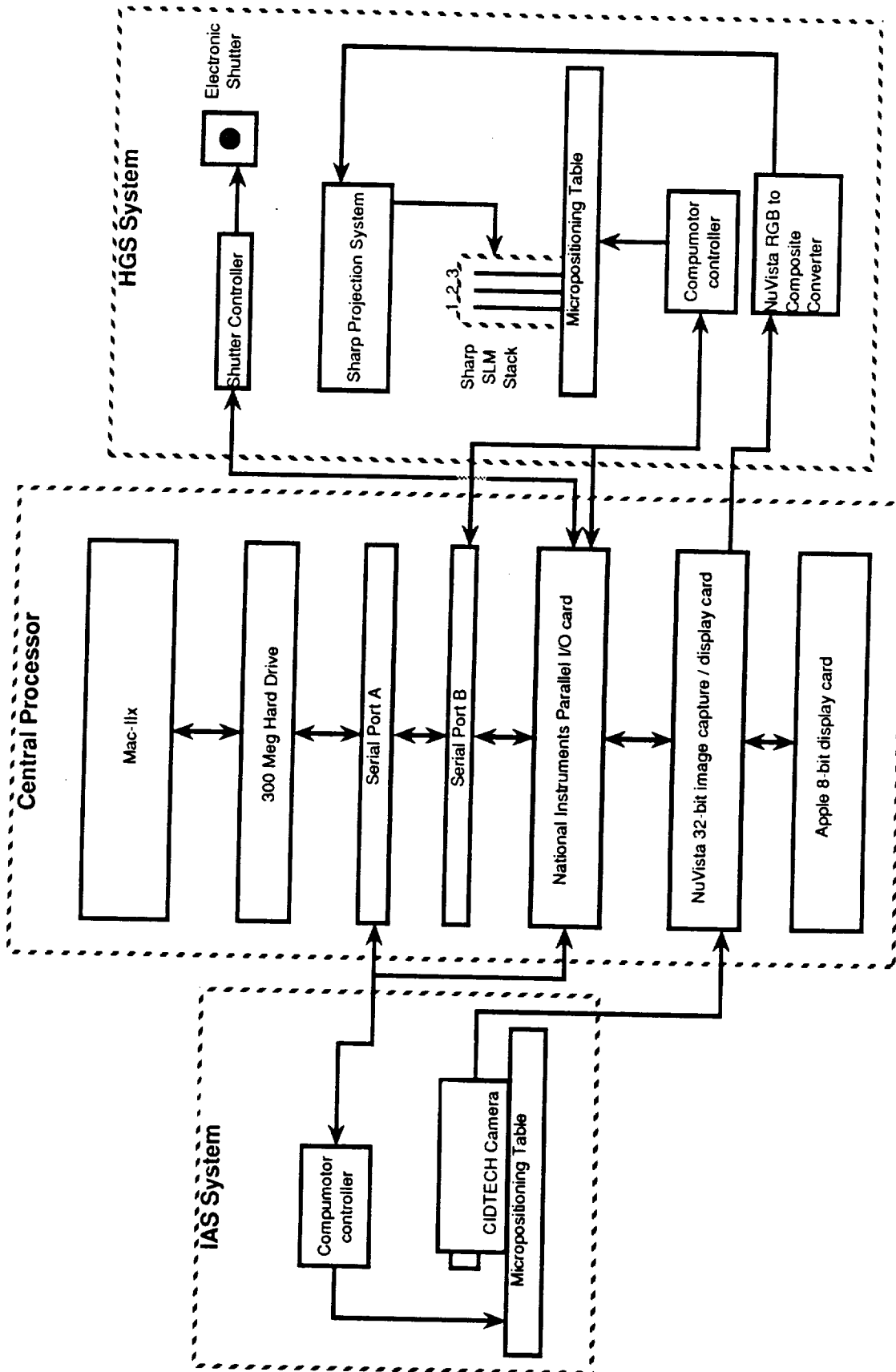


Figure A-1. HERSS Hardware Architecture

The National Instruments Parallel I/O card is used to handle various handshaking operations that take place between the Mac-IIx hardware and the laser shutter as well as the Compumotor drivers that control the position of the Daedal micropositioning tables. The Compumotor controllers are monitored by the I/O card in order to determine when table movement is complete. The laser shutter controller is also strobed and sensed by the I/O card to determine when the exposure cycle is complete.

The Compumotor controllers are addressed by the Mac-IIx via the two RS-232C serial ports. The HERSS software uses these serial ports to send commands to the controllers to move the tables to various positions. As mentioned above, the I/O card is then used to verify the table has settled before continuing with operations.

The additional Apple 8-bit color display card is used for two purposes. First, it acts as a display card to address the nView SLM. Secondly, it is used as the main display card when the Truevision NuVista card is being used to generate SLM images.

Backups of system software and image databases are accomplished using the 60 MEG tape backup unit. The backups are stored to insure no data loss if any problems are experienced with the 300 MEG hard drive.

A.3 SOFTWARE

A number of software packages were also purchased for use in the development of the HERSS system software including:

- Think C Version 4.0,
- FaceIt User Interface software,
- Macintosh Programmers Workshop (MPW), and
- MPW C.

All the HERSS system controller software was written in C using Think C, except for the NuVista image acquisition routines which were written in MPW C to keep compatibility with existing NuVista acquisition libraries. The primary controller software uses FaceIt to

handle the implementation of the high level user interface. FaceIt is the main link between the user and the lower level IAS, HGS, DIPS, and analysis controllers. The interactions between the various software elements of the HERSS software are shown in Figure A-2.

The IAS controller is responsible for acquiring and saving an image database. Lower level routines are invoked to move the positioning table to user-defined locations during image acquisition. Routines are also invoked to acquire and store image databases. The information controlling the acquisition of a database is entered via FaceIt user input routines. Following data acquisition, the databases are then stored on disk under a user-provided name.

The routines that acquire the video camera images for the IAS controller are written in MPW C and utilize lower level routines that were provided by True Vision. The images are acquired and stored in an image array allocated by the higher IAS controller and are written to disk by low level file I/O routines located in the HERSS system software. Figure A-2 illustrates the relationship between the low level file I/O and the higher level system functions.

The micropositioning tables are controlled by a module containing I/O routines necessary to move either of the two positioning tables to any supported position. The tables are controlled by sending serial commands from either of the RS-232 ports available on the Mac-IIx. By sending command strings to the Compumotor table controllers, the tables will move to the appropriate position. Feedback is provided from the Compumotor controller in the form of digital signals to indicate when a table move has been completed.

An image database file stores the information needed to post-process a database collected by the IAS. Upon acquisition, each image plane is saved in a unique depth plane file with a resolution of 512 x 486 by 8-bit intensity. The filenames for each image plane acquired are determined by appending the user-defined name of the database with a period and the particular plane number (e.g., DBNAME.1). The following information is also contained within the database file :

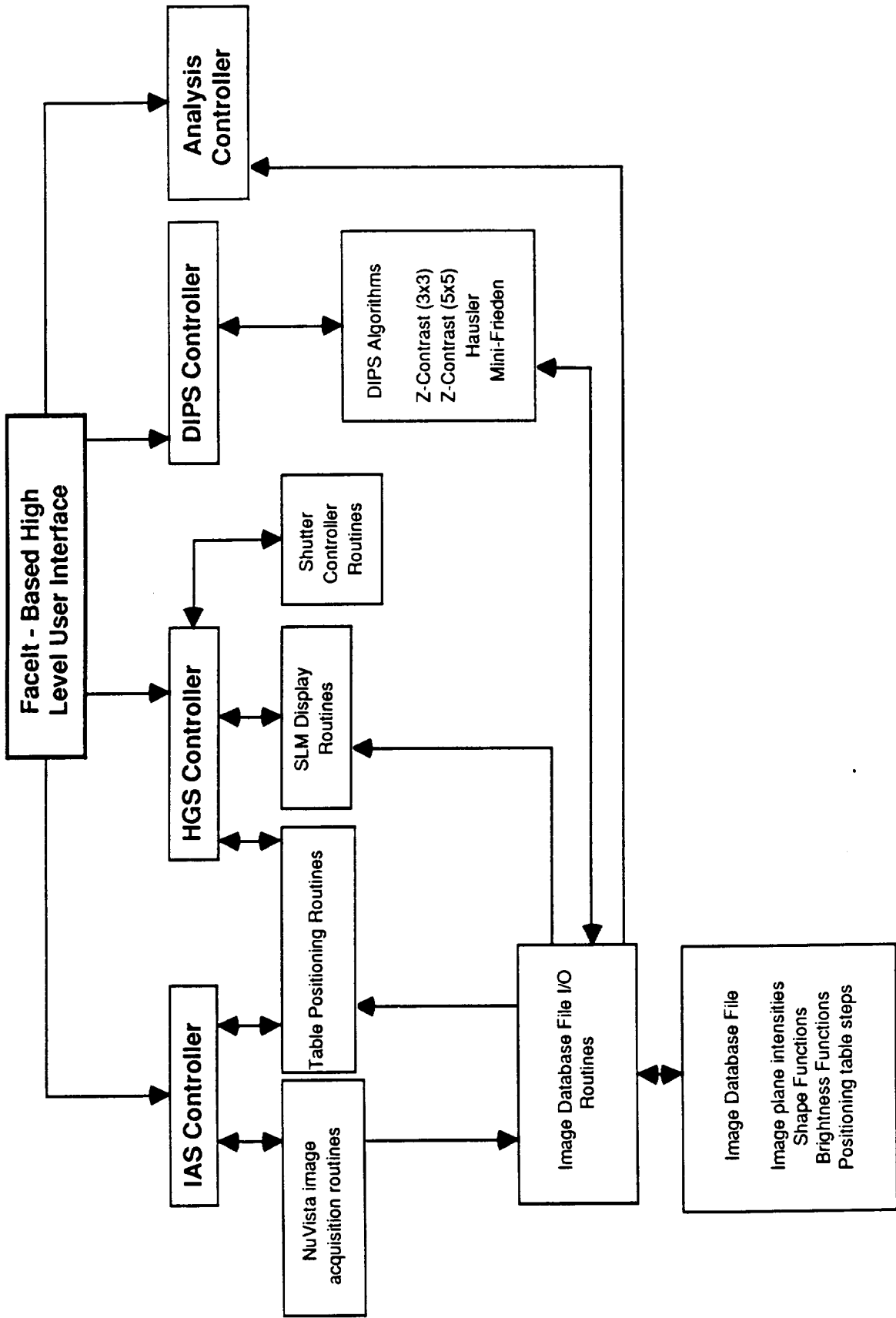


Figure A-2. HERSS Software Architecture

- Name of the database,
- Date and time of database creation,
- Number of image planes present in database,
- Any shape and brightness functions created by DIPS,
- Notes regarding the database, and
- IAS/HGS table step positions.

The DIPS controller is the simplest controller present in the HERSS system software since its primary function is to call whatever algorithm the user has selected to process a raw video image database. The DIPS algorithms access the low level file I/O functions to perform the necessary database file manipulations. Once the algorithm processing is completed, results are stored in two files in the database. One file contains the brightness function in a 512 x 486 array with an 8-bit intensity. The second file contains the shape function, in a 512 x 486 array with a 16-bit integer designating the plane of best focus. Specifically, each array element specifies the z depth location where a surface point is located.

The HGS controller uses the information processed by the DIPS algorithms to reconstruct the DIPS-processed image holographically. Specific instructions regarding the generation of a hologram is entered by the user via a FaceIt dialog box. This dialog box allows selection of either the single nView SLM or the three SharpVision SLMs, beginning and ending depth planes, and other information related to the generation of the hologram. The HGS controller uses custom SLM display routines to present the appropriate depth planes on the correct SLMs. The HGS positioning table is addressed in the same way as in the IAS system using the table step positions stored in the database. The HGS system also accesses low level shutter control routines to trigger the laser shutter at exposure time.

To address the SharpVision SLMs, custom routines were developed so that given a set of depth planes, the software displays the appropriate image planes on the appropriate SLMs. Image data is transmitted from the NuVista video board to the SharpVision system via an RGB-to-composite-video converter. The data for three image slices were simultaneously transmitted – one image was transmitted on the RED signal, one on the BLUE, and another on

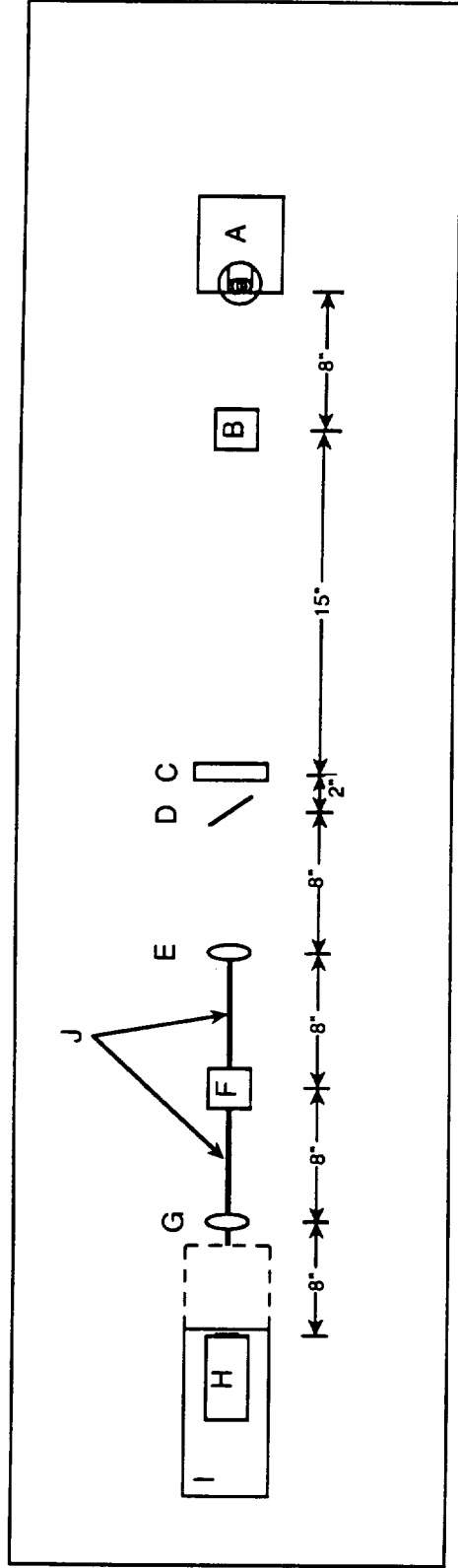
the GREEN. The SharpVision system electronics were used to transmit the appropriate image data to each monochromatic SLM in the stack.

On each SLM, pixels were closed if no information was present. Pixels containing image information were opened to the appropriate grey level; the corresponding pixels in the other SLMs were set to full open to allow light to strike an image-containing pixel or to allow the image information to be transmitted through to the thermoplastic camera. It was necessary to invert the intensity of the image on one SLM to account for variations in the polarization coatings among the SLM units. This adjustment allowed the light of the object beam to be transmitted through the entire SLM stack. The horizontal ordering of pixels was also inverted for one SLM so that the three image planes were rectified.

The analysis control module is at a higher level than the other controllers present in the system. This module is directly controlled by interaction of the FaceIt software with the user. Specific information is retrieved from the user-specified database files. This controller allows the user to view raw video image planes, the DIPS output of the shape and brightness functions. It also has the ability to reconstruct image information by using the DIPS shape and brightness functions. That is, the analysis module can present the image points contained in a specific depth plane as specified DIPS algorithm results. Additionally, the analysis controller can perform image plane subtraction operations for use in the simple comparison of images.

APPENDIX B

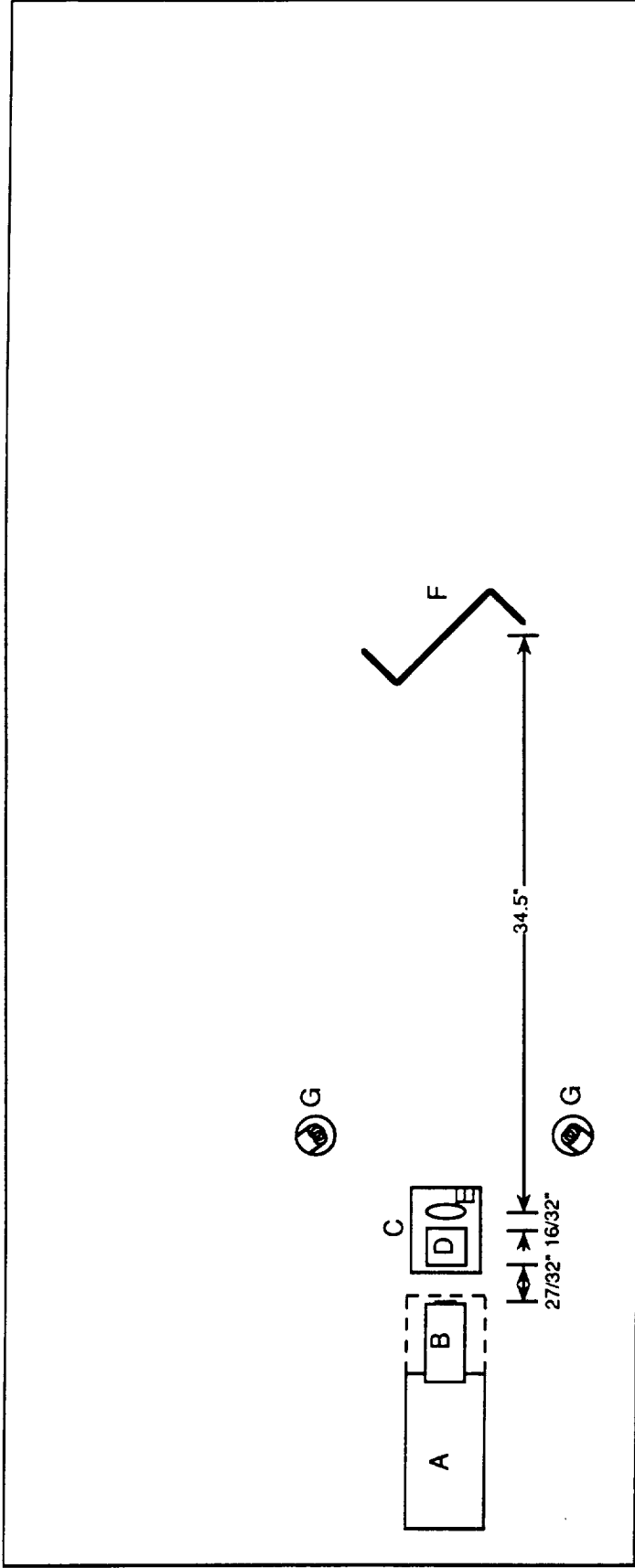
**IAS AND HGS OPTICAL CONFIGURATIONS
AND COMPONENT LISTS**



- A - ILLUMINATION SOURCE:
KODAK PROJECTION SYSTEM
- B - ORIEL IRIS 1/16" PINHOLE*
- C - DIFFUSER
- D - AIR FORCE TEST CHART
- E,G - ORIEL ACHROMATIC DOUBLET*
(200mm FL, 50mm DIA)
- F - ORIEL IRIS 1" APERTURE*
- H - CIDTECH VIDEO CAMERA
- I - DAEDAL TRANSLATION TABLE/COMPUMOTOR DRIVER
- J - OPTICAL RAIL*

* Equipment on loan from Georgia Institute of Technology

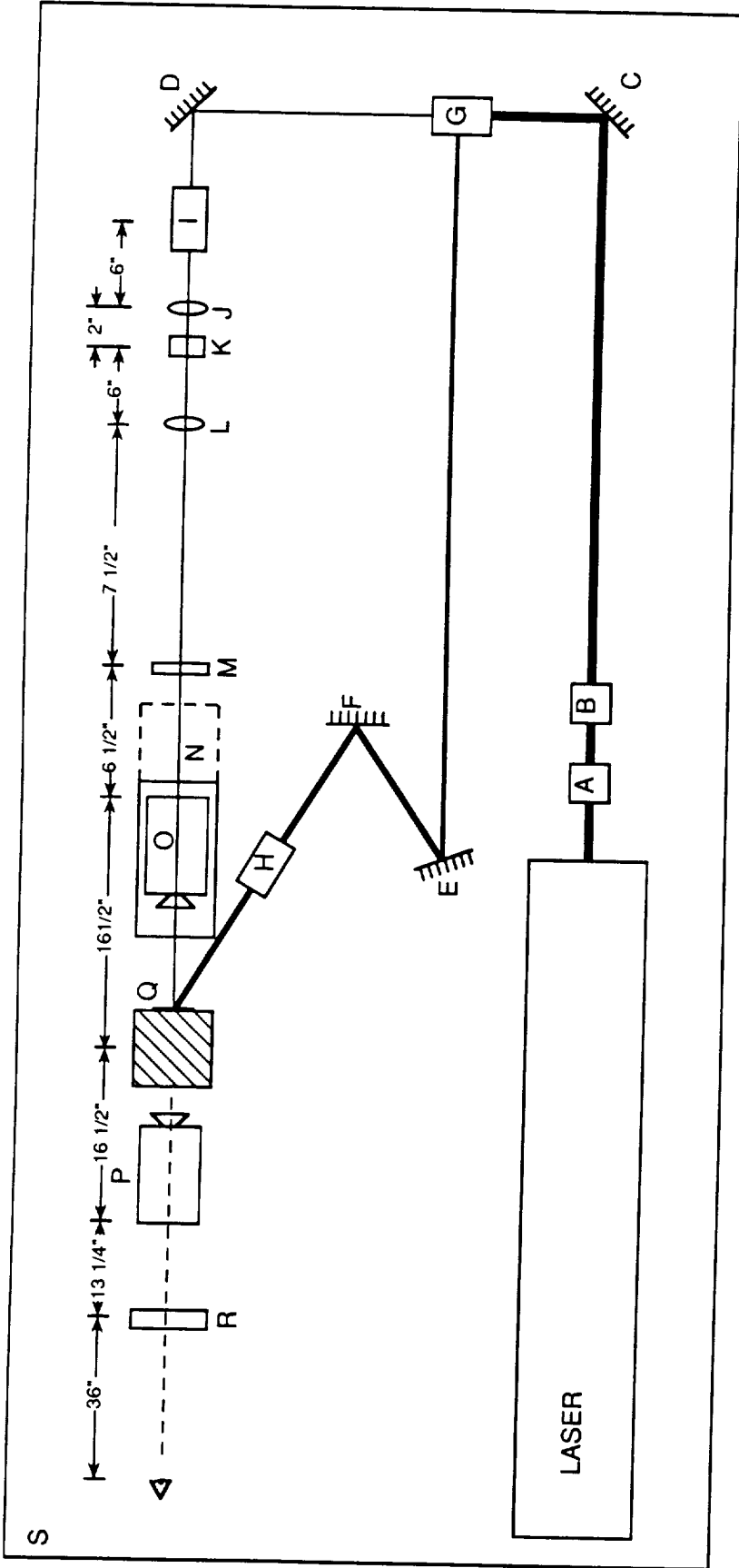
Figure B-1. IAS Baseline: AT Optical Configuration and Component List



- A - DAEDAL TRANSLATION TABLE/COMPUMOTOR DRIVER
- B - CIDTECH VIDEO CAMERA
- C - NIKON LENS ASSEMBLY*
- D - NIKON SHUTTER*
- E - MICRO-NICOR 55mm LENS*
- F - OBJECT
- G - ILLUMINATION SOURCES (INCANDESCENT BULBS)

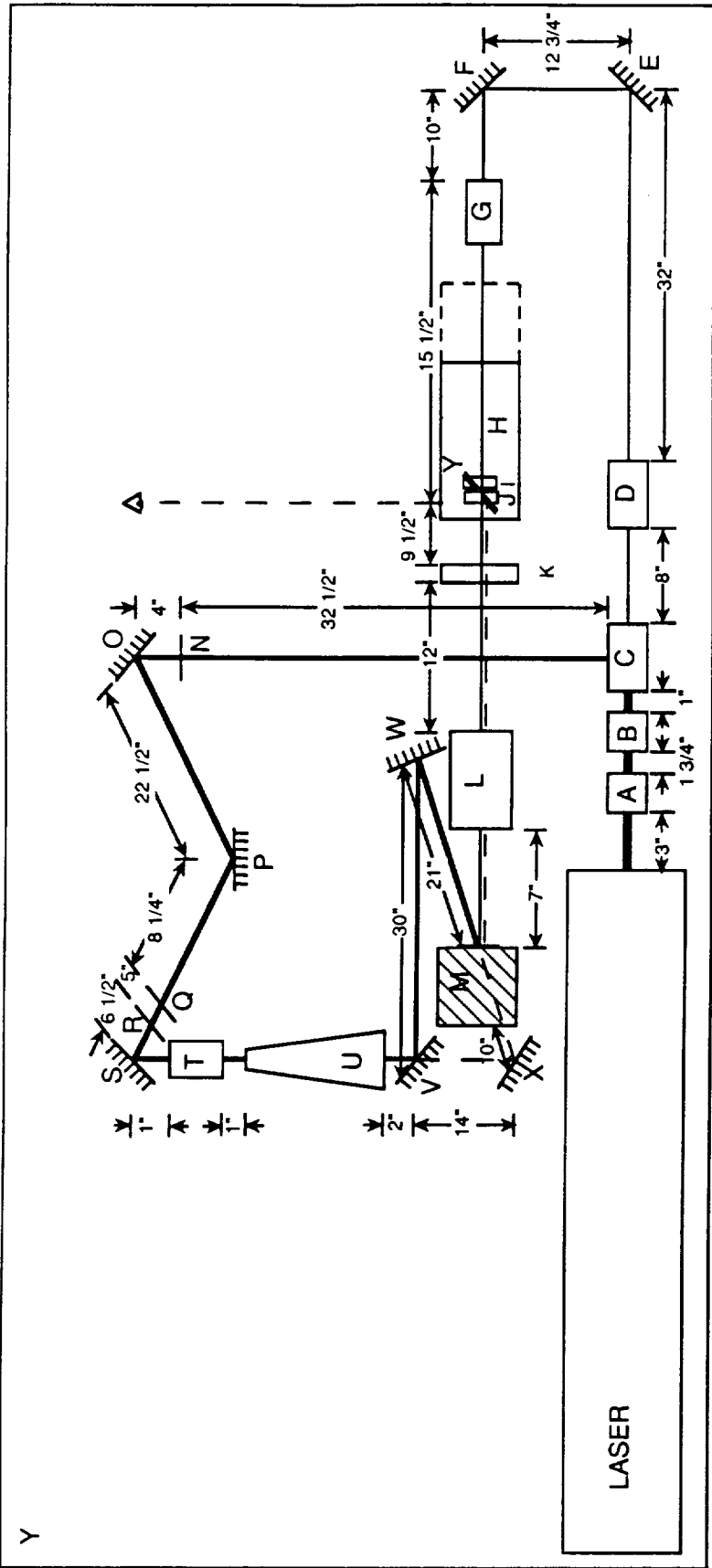
* Equipment on loan from Georgia Institute of Technology

Figure B-2. IAS Alternative: Non-AT Optical Configuration and Component List



- A - ORIEL BEAM RISER
 - B - UNIBLITZ SHUTTER
 - C,D,E,F - ORIEL 2" MIRRORS
 - G - NEWPORT BEAM SPLITTER
 - H,I - ORIEL SPATIAL FILTER (10 MICRON PINHOLE 20x OBJECTIVE)
 - J - ORIEL CONDENSING LENS (120mm FL, 95mm DIA)
 - K - OBJECT
 - L - ORIEL PROJECTION LENS (100mm FL, 25mm DIA)
 - M - DIFFUSER
 - N - DAEDAL COMPUTOR DRIVER TRANSLATION TABLE
 - O,P - H & R TV PROJECTION LENS
 - Q - NEWPORT THERMOPLASTIC CAMERA
 - R - FRESNEL TECHNOLOGIES 9"FL FRESNEL LENS
 - S - ORIEL 4' x 10' OPTICAL TABLE
- LASER SOURCE
 - REFERENCE BEAM
 - OBJECT BEAM
 - - - - - VIEWING GEOMETRY

B-3. HGS Baseline: Optical Configuration And Component List



— LASER SOURCE
— REFERENCE BEAM
— OBJECT BEAM
- - - VIEWING GEOMETRY

- O.P.S. - ORIEL 2" MIRROR
- Q - WAGAN OPTICS POLARIZER
- R - EALING ELECTRO OPTICS ATTENUATOR
- T - ORIEL SPATIAL FILTER
- U - OPTICS PLUS COLLIMATOR
- V - ORIEL 4" MIRROR ON KINEMATIC BASE (Generate Mode)
- W - ORIEL 4" MIRROR
- X - ORIEL 4" MIRROR (Viewing Mode)
- Y - EDMUND SCIENTIFIC 8" x 11" MIRROR (Viewing Mode)
- Z - ORIEL 4' x 10' OPTICAL TABLE

- A - ORIEL BEAM RISER
- B - UNIBLITZ SHUTTER
- C - NEWPORT BEAM SPLITTER
- D - NEWPORT ROTATIONAL POLARIZER
- E, F - ORIEL 2" MIRROR
- G - ORIEL SPATIAL FILTER (10 MICRON PINHOLE, 20x OBJECTIVE)
- H - DAEDAL TRANSLATION TABLE/COMPUMOTOR DRIVER
- I - DIFFUSER (Generate Mode)
- J - OBJECT (Generate Mode)
- K - FRESNEL TECHNOLOGIES 9" FL FRESNEL FIELD LENS
- L - H&R TV PROJECTION LENS
- M - NEWPORT THERMOPLASTIC CAMERA
- N - ORIEL HALF-WAVE PLATE

Figure B-4. HGS Alternative: Phase Conjugate Configuration and Component List

**NAS7-1036
Auxiliary Report 2168**

**HOLOGRAPHIC ENHANCED REMOTE SENSING SYSTEM:
SOFTWARE GUIDE**

August 1990

SCIENTIFIC AND TECHNICAL REPORT

Submitted to:

**National Aeronautics and Space Administration
Dr. Neville I. Marzwell
Contracting Officer's Technical Representative
Jet Propulsion Laboratory
Pasadena, CA**

Prepared by:

**Thomas J. Janiszewski
Helene P. Iavecchia
Sarvesh Mathur**

TABLE OF CONTENTS

1. INTRODUCTION

2. HERSS USER'S GUIDE

2.1	Accessing the HERSS Software System	2-2
2.2	Image Acquisition System.....	2-3
2.2.1	Equipment Set Up Procedures.....	2-3
2.2.2	Display Monitor Set Up Procedures	2-5
2.2.3	Software Set Up Procedures.....	2-7
2.2.4	Software Execution Procedures	2-9
2.3	Digital Image Processing System.....	2-9
2.3.1	Software Execution Procedures	2-10
2.4	Holographic Generation System.....	2-11
2.4.1	Equipment Set Up Procedures.....	2-11
2.4.2	Display Monitor Set Up Procedures	2-13
2.4.3	Software Set Up Procedures.....	2-14
2.4.4	Software Execution Procedures	2-14
2.5	Analysis.....	2-16
2.5.1	Display Monitor Set Up Procedures	2-16
2.5.2	Execution Procedures.....	2-16
2.6	Options	2-17
2.6.1	Execution Procedures	

REFERENCES

APPENDIX A. SOFTWARE FLOW DIAGRAMS

APPENDIX B. SOFTWARE CODE LISTING

LIST OF FIGURES

Figure 1	HERSS Main Menu and Associated Pull Down Menus.....	2-1
Figure 2	Macintosh-IIx System I/O Configuration.....	2-3
Figure 3	Hardware Interface between the CIDTECH Camera and the NuVista Board Installed on the Macintosh-IIx.....	2-4
Figure 4	Hardware Interface of the IAS Compumotor Driver to Serial Port A and the National Instruments Board Installed on the Macintosh-IIx.....	2-4
Figure 5	NuVista Set Up Dialog Box.....	2-5
Figure 6	Control Panel Dialog Box.....	2-6
Figure 7	New Project Dialog Box.....	2-7
Figure 8	Autostep Dialog Box.....	2-8
Figure 9	Hardware Interface of the HGS Compumotor Driver to Serial Port B and the National Instruments Board Installed on the Macintosh-IIx.....	2-11
Figure 10	Hardware Interface of the Uniblitz Shutter Controller and the National Instruments Board Installed on the Macintosh-IIx.....	2-12
Figure 11	Hardware Interface between the Macintosh-IIx, the VIDI/O Converter, the Sharp Projection System, and the Display Monitors.....	2-12
Figure 12	Auto Generate Hologram Dialog Box.....	2-15
Figure 13	Analysis Dialog Box.....	2-17
Figure 14	Talk to Table Dialog Box.....	2-18

1. INTRODUCTION

This document supplements a final report written as part of a research effort to build a prototype Holographic Enhanced Remote Sensing System (HERSS). This effort was sponsored by the National Aeronautics and Space Administration (NASA) under a Phase II Small Business Innovation Research (SBIR) Program, contract NAS7-1036. The system consisted of three primary subsystems: (1) an Image Acquisition System (IAS) to acquire video image data of a remote object; (2) a Digital Image Processing System (DIPS) to process the raw image data and build a numerical data base of the object; and (3) a Holographic Generation System to generate a hologram of the source object (Iavecchia, Gaynor, Huff, et al., 1990).

This document describes how to use the system software, hosted by a Macintosh-IIx, to control these three HERSS subsystems as well as various supporting functions. Section 2 presents instructions for system set up and use and contains detailed documentation for using the high-level software interface. The software-user interface was built using a Macintosh-like menu structure. Five primary menu options, as described in Section 2, include:

- **Project** - Assists the user in data file creation and management. Project data files contain user-specified commands and parameters that will subsequently be accessed by the IAS, DIPS, and HGS subsystems.
- **IAS** - Acquires a three dimensional image database according to parameter set in the current project file in memory.
- **DIPS** - Controls executions of image restoration algorithms to generate shape and brightness functions using the raw video data collected by the IAS.
- **HGS** - Generates a hologram using the shape and brightness functions produced by DIPS.
- **Analysis** - Assists the user in analyzing images collected by the IAS and/or processed by the DIPS.

Appendix A contains HERSS software flow diagrams. These flow diagrams are organized according to the user-interface design and illustrate branching to various submenus following user main menu selections. Any box in a particular flow diagram that

has a shaded box denotes that further flow diagram documentation is available within the Appendix. Eventually the submenus terminate with the names of software routines. The code associated with each software routine is listed in Appendix B.

Appendix B contains a listing of the software source code for the entire HERSS system. An index is provided at the end of Appendix B for the convenience of the reader containing the name of a routine, the module associated with the routine, and the page number where the routine can be found within the appendix. Each routine is further documented in the source code with a statement of functions as well as data input and output .

2. HERSS USER'S GUIDE

This section describes how to use the HERSS central computer to acquire an image data base with the Image Acquisition System (IAS), how to process the image data to determine the shape and brightness of the object with the Digital Image Processing System (DIPS), and how to display the object information holographically with the Holographic Generation System (HGS). Instructions for configuring equipment components and software parameter defaults are provided. The documentation is written assuming the reader is familiar with Macintosh computer system operations.

2.1 ACCESSING THE HERSS SOFTWARE SYSTEM

To access the HERSS software controller on the Macintosh-IIx, perform the following steps:

1. Locate the folder named HERSS.
2. Enter the Folder and execute the application program named HERSS.

Execution of the program will result in display of the main menu options illustrated in Figure 1. Pull down menus associated with each of these options are shaded.

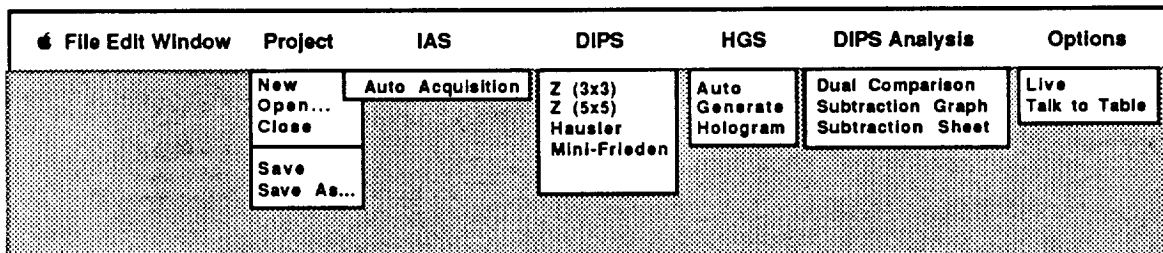


Figure 1. HERSS Main Menu and Associated Pull Down Menus

The primary functions of each main menu option are:

- PROJECT** — Establishes a project name for a particular data base that will be tracked throughout the image acquisition, image processing, and holographic display cycle. The user-generated project name will first be associated with the raw image data base collected by the IAS. This name will then be associated with the output files generated by the DIPS that define the object's shape and brightness following processing of the raw image data. The HGS will access the object shape and brightness functions while generating a hologram.
- IAS** — Automatically controls the image acquisition process. Accepts data from the user concerning the number of image slices to be collected in a six inch depth and the step distance between image slices. Controls the frame grabbing operation of the CIDTECH camera as well as the micropositioning table to which the camera is mounted. Automatically invokes a table-move - frame-grab cycle until an entire image data base is collected.
- DIPS** — Accepts data from the user specifying what raw image data base is to be processed (i.e., the PROJECT name) as well as the specific algorithm to be utilized. There are four alternative restoration algorithms that can be selected — Z-contrast (3 x 3), Z-contrast (5 x 5), Hausler, and Mini-Frieden.
- HGS** — Accepts data from the user specifying what processed image data base is to be holographically displayed, the total number of planes (or set of planes) to be exposed, what SLM(s) is (are) to be used (nView or Sharp), as well as what information within a data base will be displayed (ability to skip images or compress several image planes onto one SLM).
- DIPS ANALYSIS** — Provides the ability to visually or numerically compare the differences between two image data files named by the user. The Dual Comparison option presents a grey scale picture of pixel intensity values associated with each file. The Subtraction Graph option subtracts the pixel intensity values of one image file from the corresponding pixels in the other image file and displays the results using a grey scale. The Subtraction Sheet option performs the same operation as Subtraction Graph but presents a spreadsheet of the difference numerically.
- OPTION** — Allows the user to display the current CIDTECH camera image on the display monitor (Live) option or to send commands to the micropositioning tables (Talk to Table option).

Use of each of these functions will be elaborated in the following sections. The *File*, *Edit*, and *Window* main menu options are not needed to control the HERSS subsystems and will not be discussed further. These options, associated with the FaceIt software package, were used to build the HERSS user interface. The interested reader is directed to the FaceIt software documentation.

2.2 IMAGE ACQUISITION SYSTEM

This section explains the necessary procedures for set up of the IAS equipment components, set-up of the display monitor, specification of software parameters to control the acquisition process, as well as the final software execution procedures. Each of these procedures are discussed below.

2.2.1 Equipment Set Up Procedures

Figure 2 illustrates the Macintosh-IIx I/O configuration. Refer to this figure, as necessary, when establishing cable hook-ups between the Macintosh and peripheral equipment. IAS equipment set up procedures include:

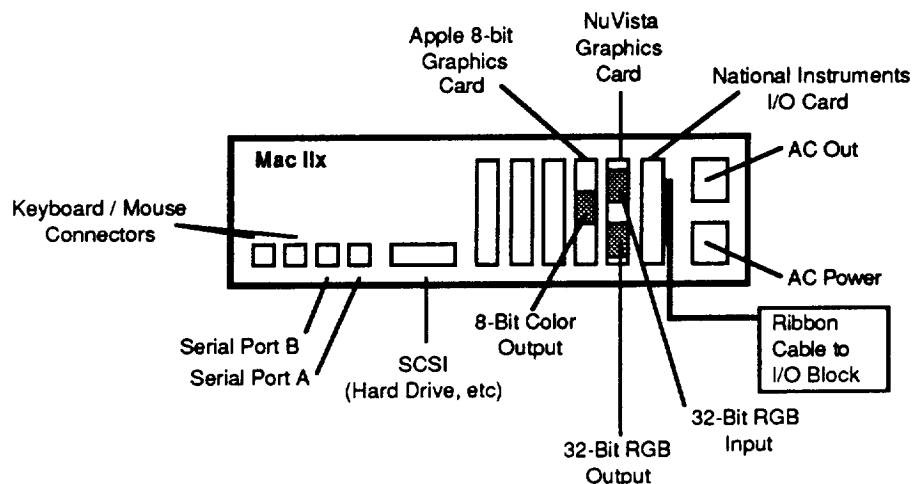


Figure 2. Macintosh-IIx System I/O Configuration

1. Set up the IAS lens configuration (e.g., afocal-telecentric or non-afocal-telecentric) including placement of the test object to be imaged.

- As shown in Figure 3, connect the CIDTECH BNC video output connector to the NuVista green BNC video input port installed in the Macintosh-IIx.

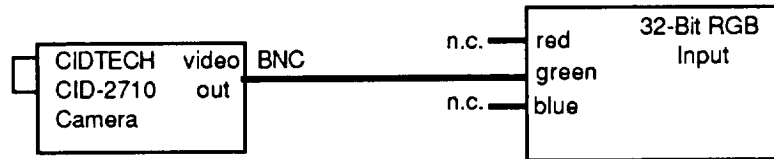


Figure 3. Hardware Interface between the CIDTECH Camera and the NuVista Board Installed on the Macintosh-IIx.

- As shown in Figure 4, connect the cables between the Compumotor driver and the Macintosh RS-232 port A as well as the National Instrument I/O board port which is installed in the Macintosh-IIx. Also connect cables between the Compumotor and the motor of the Daedal micropositioning table as illustrated in Figure 4.

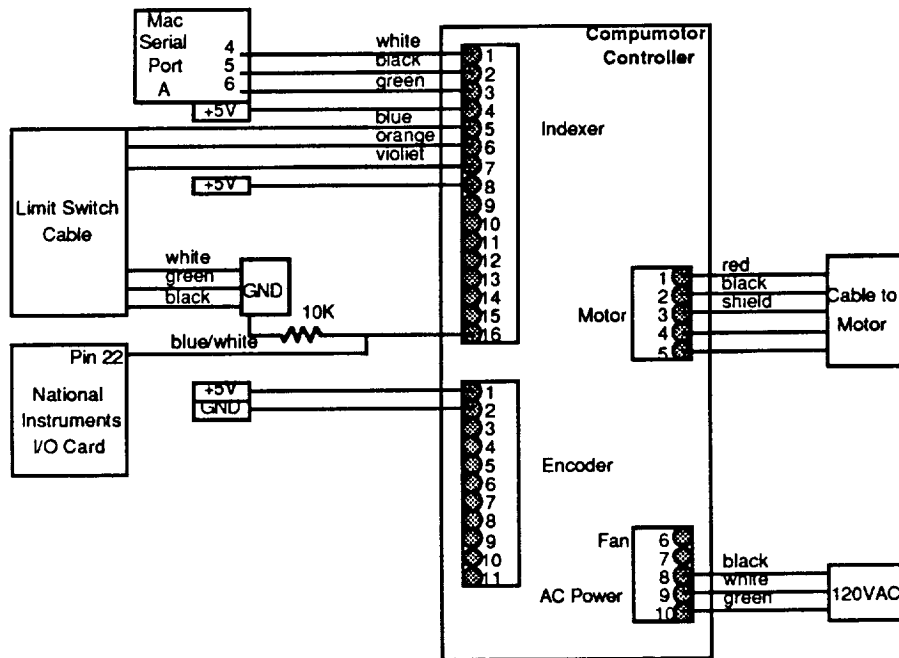


Figure 4. Hardware Interface of the IAS Compumotor Driver to Serial Port A and the National Instruments Board Installed on the Macintosh-IIx.

4. Turn the power on to the CIDTECH camera and the IAS micropositioning table.

2.2.2 Display Monitor Set Up Procedures

The set up procedures for the display monitor are:

1. Initiate execution of the NuVista SetUp program by double clicking on the file named **SetUp** in the HERSS folder. The SetUp dialog box will be displayed as illustrated in Figure 5.

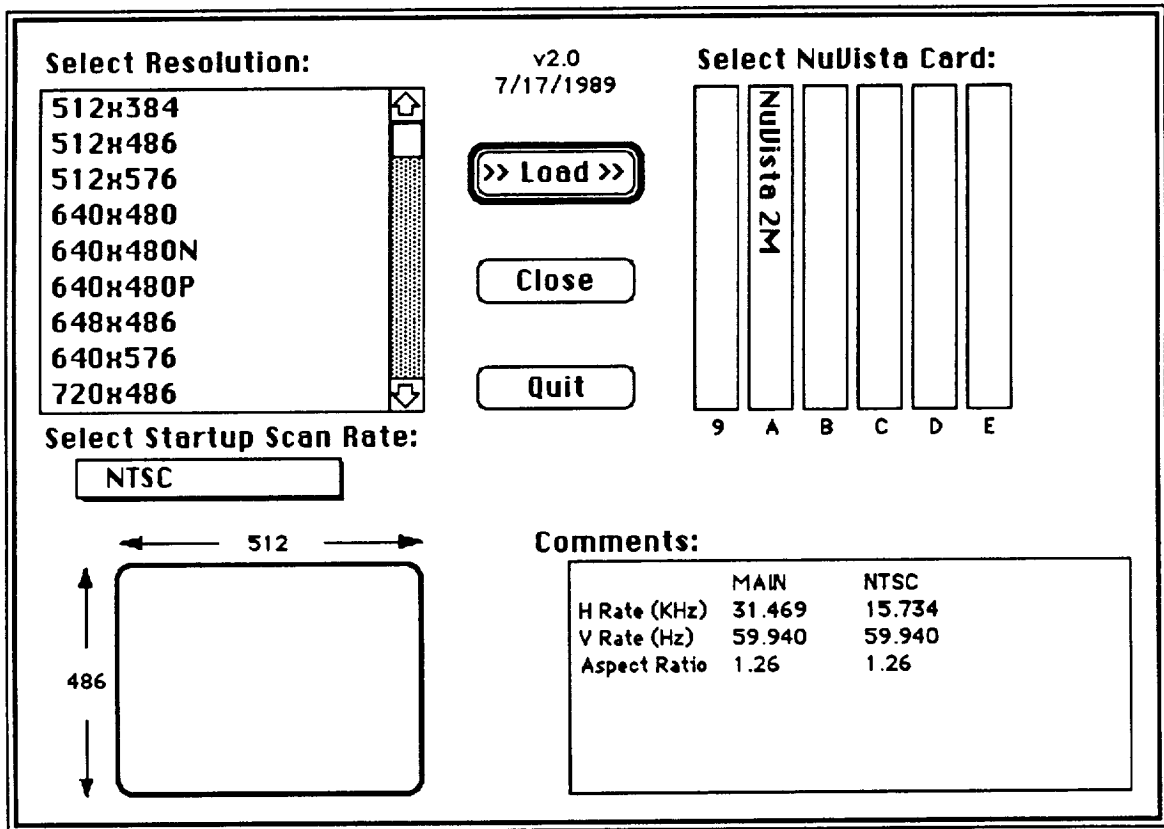


Figure 5. NuVista SetUp Dialog Box

2. Use the **Select Resolution** scroll bar to choose **512x486**. Position and click the mouse on this option.
3. Use the **Select Startup Scan Rate** pull down menu to select **NTSC**.
4. Select the Load option by clicking on the **Load** button.
5. Select the Quit option by clicking on the **Quit** button.

6. From the Macintosh Finder, use the **Special** pull down menu to select **Restart**.
7. In the Macintosh Finder, select **Monitors**. On the Control Panel dialog box illustrated in Figure 6, under the heading "Characteristics of selected monitor", select the **Colors** button and select **Millions** using the scroll bar.

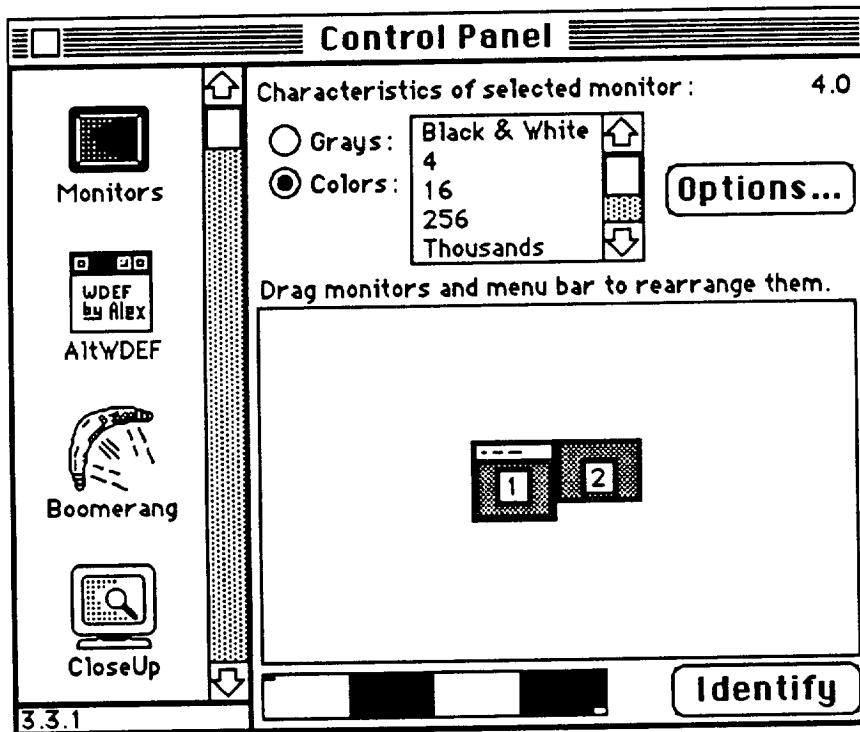


Figure 6. Control Panel Dialog Box

8. On the Control Panel dialog box, under the heading "Drag monitors and menu bar to rearrange them", note which monitor box has the top menu bar. This is the designated startup monitor. If the menu bar is not on Box 1, click onto the menu bar and drag it to Box 1. This will establish the NuVista as the startup monitor. Also, depress the **OPTION** key on the keyboard. Verify that a face appears on Box 1. If not on Box 1, click onto the face and drag it to Box 1. This is necessary for correct initialization of the NuVista board.
9. Close the Control Panel dialog box.
10. From the Macintosh Finder, use the **Special** pull down menu to select **Restart**.

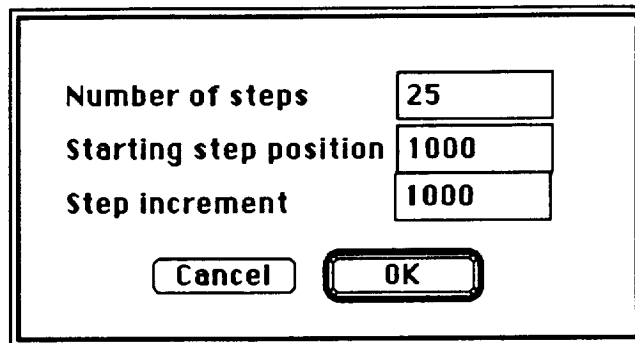


Figure 8. Autostep Dialog Box

5. Enter the total number of steps (or image acquisition cycles). Enter the starting position of the micropositioning table (usually equals 0, the home position). Enter the step increment (a positive number to move forward from the starting position and a negative number if moving backwards from the starting position). One inch of travel is represented as 64000 steps. In the example provided, 25 steps are selected; the starting position is 1000; the step distance increment is 1000, or approximately 0.016 inch.
6. After entering the parameters in the Autostep dialog box, select the OK option by clicking on the **OK** button. The Autostep dialog box will disappear. The New Project Dialog box will be displayed. The system will respond by entering all the table position commands in the adjacent vertical window named **Editor**. This window can be edited manually if desired.
7. Use the **Project** pull down menu to select the **Save as** option. The standard Macintosh dialog box window will be displayed. Enter the desired file (Project) name. Click the **Save** button. Transparent to the user, the HERSS system will create three files:
 - (a) *Projectname* — Key file that stores the name, date, time, and the number of steps (or image slices to be collected) as defined by the user in the **New Project** and **Autostep** dialog boxes.
 - (b) *Projectname.step* — Stores the step position commands to be sent to the micropositioning table during the image acquisition process as defined by the user within the **Autostep** dialog box.
 - (c) *Projectname.note* — Stores any notes associated with this project as defined by the user within the **New Project** dialog box.

2.2.4 Software Execution Procedures

To invoke the image acquisition process immediately following the set up procedures described above, only one step is required:

1. Use the **IAS** pull down menu to select **Auto Acquisition**.

The system will assume the acquisition parameters of the new project just created. The system will send a command to the micropositioning table, via the Compumotor driver, to move to the first step position and perform a frame grab. The move-grab-store cycle will automatically be repeated until the entire data base is collected. Each video image is stored in a separate file named *projectname.slice* (e.g., TESTCHART.1). *Projectname* is the user-defined data base name as specified in the New Project dialog box and *slice* is the image slice number, or depth plane number, of the current frame grab.

The user also has the option to invoke data acquisition parameters associated with a previously existing project. For this case, perform the following steps:

1. Perform the equipment set up procedures described in Section 2.2.1.
2. Perform the display monitor set up procedures described in Section 2.2.2.
3. Use the **Project** pull down menu to select **Open**. Use the standard Macintosh interface procedures to open the desired file name.
4. Edit the parameters in the Project dialog box, renaming the project if desired to avoid overwriting of an existing data base.
5. Use the **Project** pull down menu to select the **Save** (same file name) or **Save as** (new file name desired) option.
6. Use the **IAS** pull down menu to select **Auto Acquisition**.

2.3 DIGITAL IMAGE PROCESSING SYSTEM

This section explains the necessary procedures for execution of the DIPS image processing algorithms. No special equipment or display monitor set up procedures are necessary.

2.3.1 Software Execution Procedures

To invoke an image processing algorithm immediately following image acquisition, only one step is necessary:

1. Use the **DIPS** pull down menu to select one of the following algorithm options — **Z (3 x 3)**, **Z (5 x 5)**, **Hausler**, **Mini-Frieden**.

The first two options invoke the z-contrast algorithm with either a 3x3 or 5x5 kernel size. The third option invokes the Hausler algorithm. The last option invokes the revised Frieden shape algorithm that scans the brightness values along the z-axis and designates the surface point location as the depth plane with the maximum brightness. Each of these algorithms are described in the HERSS final report (Iavecchia, et al., 1990). Flow diagrams for these algorithms and the software code are provided in Appendix A and B, respectively.

If the desired raw image data base file is not open, perform the following steps:

1. Use the **Project** pull down menu to select **Open**. Use the standard Macintosh interface procedures to open the desired file name.
2. Use the **DIPS** pull down menu to select one of the following algorithm options — **Z (3 x 3)**, **Z (5 x 5)**, **Hausler**, **Mini-Frieden**.

Following selection of the desired image processing algorithm, the system will invoke the named algorithm. Processing time, on the average, is approximately one minute per image slice except for the Frieden algorithm which is approximately 10 seconds per image slice. When the processing is complete, the system automatically generates two output files — a shape function file and a brightness function file. The output file naming convention is *projectname.function.type*. *Projectname* is the user-defined data base name as specified in the New Project dialog box. *Function* is either "shape" or "bright". *Type* is the number 0, 1, 2, or 3 corresponding to either the Z (3 x 3), Z (5 x 5), Hausler, or Frieden algorithms, respectively. For example, the shape function output file following execution of the Z (3 x 3) algorithm on a project data base named "TESTCHART" is named Testchart.shape.0.

2.4 HOLOGRAPHIC GENERATION SYSTEM

This section explains the necessary procedures for set up of the HGS equipment components, set up of the display monitor, specification of software parameters to control the hologram generation process, as well as the final software execution procedures. Each of these procedures are discussed below.

2.4.1 Equipment Set Up Procedures

HGS equipment set up procedures include:

1. Set up the HGS optical configuration (e.g., phase conjugate recording system using a 3-SLM stack). Inject air into the legs of the optical table to induce vibration isolation.
2. As shown in Figure 9, connect the cables between the Compumotor driver and the Macintosh RS-232 port B as well as the National Instrument I/O board port which is installed in the Macintosh-IIx. Also connect cables between the Compumotor and the motor of the Daedal micropositioning table as illustrated in Figure 9.

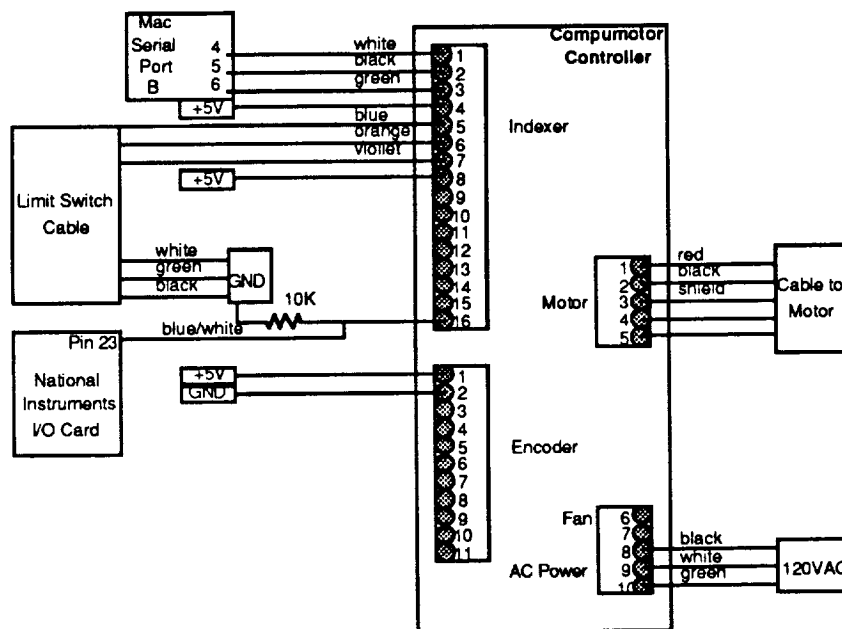


Figure 9. Hardware Interface of the HGS Compumotor Driver to Serial Port B and the National Instruments Board Installed on the Macintosh-IIx.

- As shown in Figure 10, connect the Uniblitz shutter controller to the National Instrument I/O board port which is installed in the Macintosh-IIx.

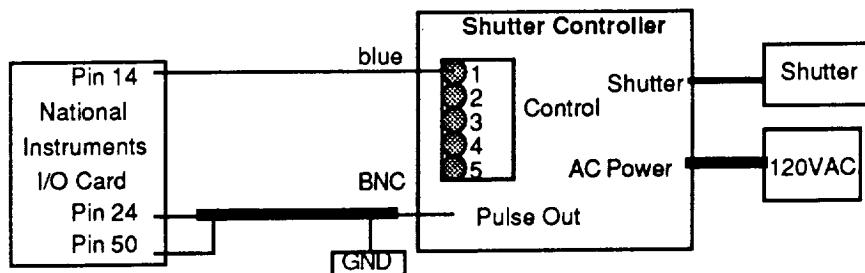


Figure 10. Hardware Interface of the Uniblitz Shutter Controller and the National Instruments Board Installed on the Macintosh-IIx.

- As shown in Figure 11, connect the RGB output of the Macintosh-IIx to the Truevision VIDI/O converter. Connect the VIDI/O composite video output to the video input port of the Sharp Projection System. Connect the video output port of the Sharp to an optional monitor, if desired to view the images being transmitted to the SLMs. Attach a cable between the Macintosh II-x 8-bit color output port and the input port of the MicroVitec monitor.

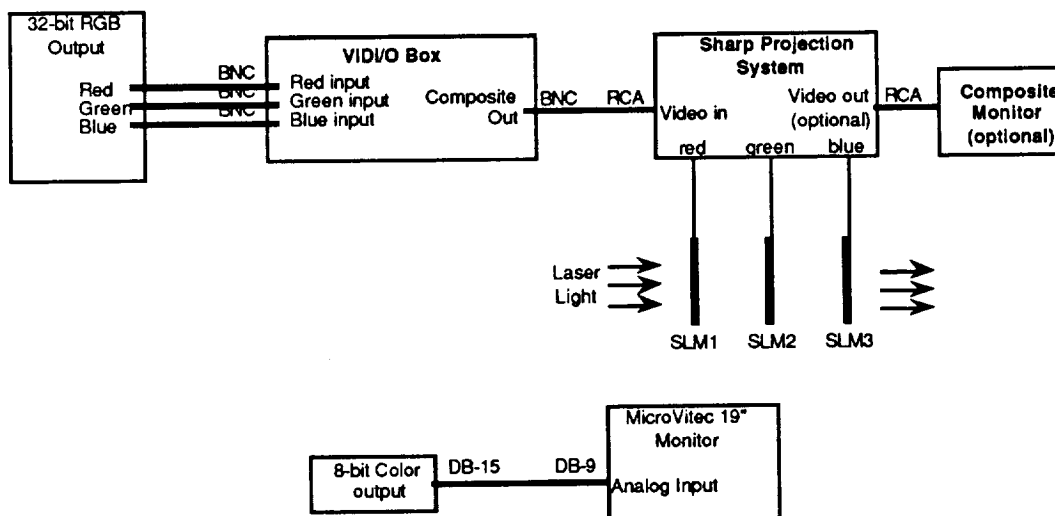


Figure 11. Hardware Interface between the Macintosh-IIx, the VIDI/O Converter, the Sharp Projection System, and the Display Monitors.

- When using the Sharp SLMs as the object source, adjust the color level and brightness settings on the Sharp projection system to achieve the highest contrast possible.

6. Turn on the power to the Uniblitz shutter, Sharp Projection System, HGS micropositioning table, laser, and associated monitors.
7. Set the exposure time on the Uniblitz shutter. (The HERSS software only controls the shutter open trigger.)
8. Perform the necessary set up procedures associated with the selected holographic recording medium. For example, when the thermoplastic camera is used, refer to the Newport Holographic Camera operations guide to determine the manual procedures associated with erasing the plate prior to exposure (or developing the plate following exposure). If the Du Pont photopolymer is used as the recording medium, turn the overhead lights off and mount the film plate. Similarly, set the laser power levels and K ratio to appropriate levels for the selected recording medium.

2.4.2 Display Monitor Set Up Procedures

The set up procedures for the display monitor are:

1. Initiate execution of the NuVista SetUp program by double clicking on the file named **SetUp** in the HERSS folder. The SetUp dialog box will be displayed as illustrated in Figure 5.
2. Use the **Select Resolution** scroll bar to choose **512x486**. Position and click the mouse on this option.
3. Use the **Select Startup Scan Rate** pull down menu to select **NTSC**.
4. Select the Load option by clicking on the **Load** button.
5. Select the Quit option by clicking on the **Quit** button.
6. From the Macintosh Finder, use the **Special** pull down menu to select **Restart**.
7. In the Macintosh Finder, from the **Apple** menu, select **Control Panel**. Then select **Monitors**. On the Monitors dialog box illustrated in Figure 6, under the heading "Characteristics of selected monitor", select the **Colors** button and select **Millions** using the scroll bar.
8. On the Control Panel dialog box, under the heading "Drag monitors and menu bar to rearrange them", note which monitor box has the top menu bar. If the menu bar is not on Box 2, click onto the menu bar and drag it to Box 2. Also, depress the OPTION key on the keyboard. Verify that a face appears on Box 2. If not on Box 2, click onto the face and drag it to Box 2.
9. Close the Control Panel dialog box.

10. From the Macintosh Finder, use the **Special** pull down menu to select **Restart**.

2.4.3 Software Set Up Procedures

Before actual a hologram can be generated, certain parameters must be specified. Procedures for specifying these parameters are:

1. Use the **Project** pull down menu to select **Open**. Use the standard Macintosh interface procedures to open the desired restored data base.
2. Edit the step positions in the vertical window named **Editor** as necessary, referring to section 2.2.4 as necessary.
3. If step positions were edited in step 2, use the **Project** pull down menu to select the **Save** option. The new step positions will then be saved to the project file.

2.4.4 Software Execution Procedures

To invoke the holographic generation process immediately following the set up procedures described above, only one step is required:

1. Use the **HGS** pull down menu to select the **Auto Generate Hologram** option. The **HGS** dialog box will be displayed as illustrated in Figure 12.
2. **Beginning slice** is set to the number of the first slice in the data base that should be displayed by the HGS system.
3. **Ending slice** is set to the last slice to be displayed by the system.
4. **Slice increment** affects the number of slices within the data base that will actually be displayed by the HGS subsystem. If all image planes within the database are to be displayed, then set **slice increment** to 1. To skip every other image plane, set **slice increment** to 2. Ignore this setting if using the 3-SLM stacking technique.
5. **Number of slices to sum** affects how the image data is compressed prior to being sent to the SLM. If the data base has 100 image planes and if it is desired that all 100 planes be compressed onto 10 planes, then set **number of slices to sum** to 10.
6. **Use Database** informs the system if the user wants to display a restored image file onto the SLM. The default setting is on, indicated by an x within the check box. If the user want to generate a hologram using a real object instead of using the SLM, then click onto the check box to disable the setting.

Beginning Slice	<input type="text" value="0"/>
Ending Slice	<input type="text" value="119"/>
Slice Increment	<input type="text" value="1"/>
Number of Slices to Sum	<input type="text" value="1"/>
<input checked="" type="checkbox"/> Use Database	<input type="button" value="Begin"/>
<input type="radio"/> Z-Contrast	
<input type="radio"/> Hausler	
<input checked="" type="radio"/> Projection	<input type="button" value="Cancel"/>
<input type="radio"/> Freiden	
<input type="radio"/> Servo Robot	
<input type="checkbox"/> Use Multiple SLMs	
SLM 2 Slice Offset	<input type="text" value="0"/>
SLM 3 Slice Offset	<input type="text" value="0"/>
SLM Set Slice Increment	<input type="text" value="0"/>
Slices Per Set Step	<input type="text" value="0"/>
<input type="checkbox"/> Use N-View SLM	

Figure 12. Auto Generate Hologram Dialog Box

7. If displaying a restored image file, select the radio button associated with the image restoration algorithm output file to be used by the HGS subsystem.
8. **Use Multiple SLMs** informs the system if the 3-SLM stacking technique will be used during hologram generation. The default setting is off, indicated by a blank check box. If the user wants to use the 3-SLM technique, then click onto the check box to enable the setting.
9. **SLM 2** and **SLM 3 slice offset** settings are used only when using the 3-SLM stack technique. The offset refers to the number of slices to skip in the image data base. If the user wants to skip over two images in the data base, then **SLM 2 slice offset** should contain 3 and **SLM 3 slice offset** should contain 6.
10. **SLM Set Slice Increment** refers to the total number of image planes that are contained within the 3-SLM stack. For example, for the case when there are three images contained on each SLM, and no image slices are skipped, then set **SLM Set Slice Increment** to 9.


11. **Use N-View SLM** informs the system if the user wants to display images on the N-View instead of the Sharp SLM. The default setting is off indicated by a blank check box. If the user want to generate a hologram using the N-View SLM, then click onto the check box to enable the setting.
12. To initiate the holographic generation process, click **Begin**.

2.5 ANALYSIS

This section explains the necessary procedures for set up of the display monitor as well as execution procedures. Each of these procedures are discussed below.

2.5.1 Display Monitor Set Up Procedures

Only one display monitor set up step is necessary:

1. In the Macintosh Finder, from the  Apple menu, select **Control Panel**. Then select **Monitors**. On the Monitors dialog box illustrated in Figure 6, under the heading "Characteristics of selected monitor", select the **Colors** button and select **256** using the scroll bar.

2.5.2 Execution Procedures

Execution procedures for the analysis module include:

1. From the DIPS analysis menu, select one of the following three options:
 - **Dual Comparison** - Displays two user selectable images.
 - **Subtraction Graph** - Subtracts two user selected images.
 - **Subtraction Sheet** - Subtracts two images and places result in spreadsheet form.

A dialog box, as illustrated in Figure 13, will be displayed.

2. For the left and right images, select the appropriate file by selecting **Choose Project**. The default is the currently opened project.
3. If the image the user selects is either a **brightness function**, a **shape function**, or a **processed slice**, the image restoration algorithm file output to be used should also be selected. This is accomplished by clicking on the appropriate radio button corresponding to each algorithm.

2.6.1 Execution Procedures

Execution procedures for the analysis module include:

1. From the Options menu, select one of the following two options:
 - **Live** - Set the NuVista graphics board to a live screen mode, displaying the input to the board from the CIDTECH camera in realtime. This option is helpful when setting up the **IAS** subsystem.
 - **Talk to table** - This option allows control of the IAS or HGS micropositioning tables via the dialog box illustrated in figure 14. The appropriate table to be controlled can be selected by clicking on the corresponding radio button. The **go home** button sends the current table to home position. The **absolute** button is used to move the table to a particular table location which can be set by editing the **absolute table position** box. The table can be moved in steps relative to its current position by using the **relative** button. The step distance to be moved from the current position should be set by editing the **relative step increment** box. The **done** button is used once all table movement is complete.

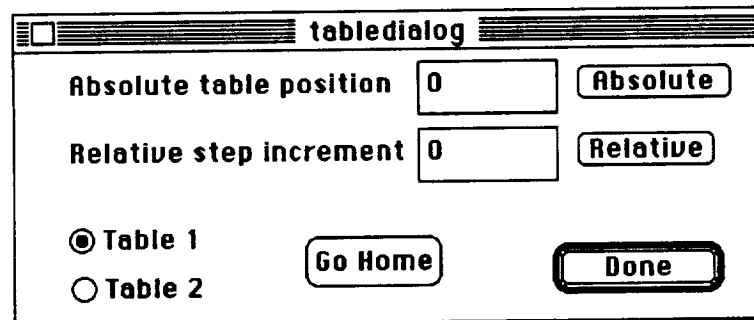
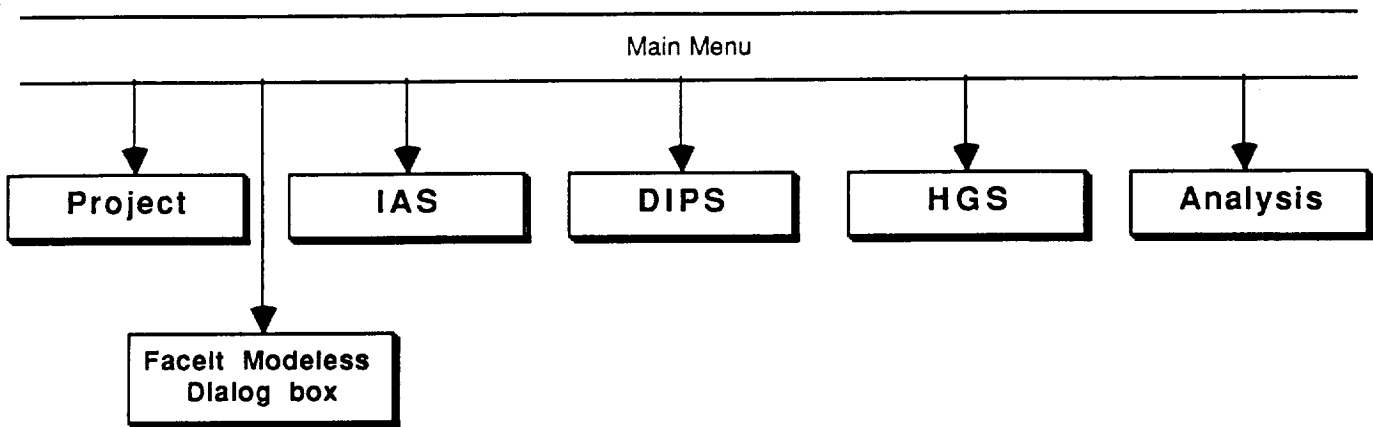


Figure 14. Talk to Table Dialog Box

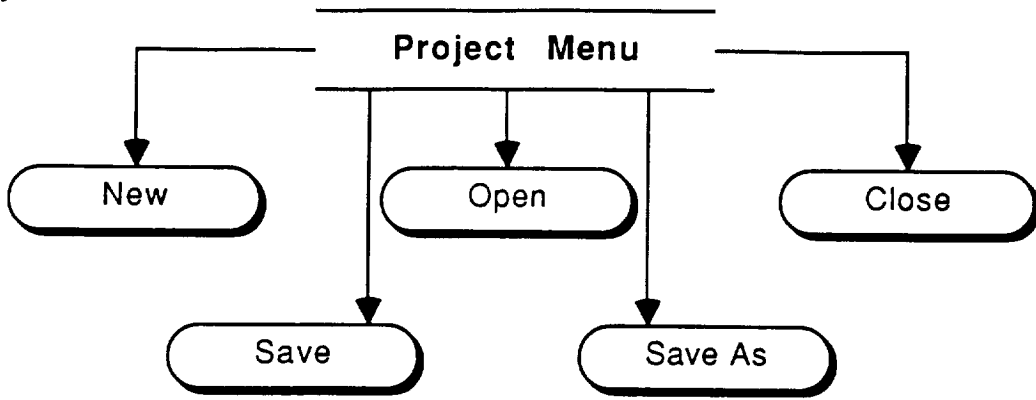
A. SOFTWARE FLOW DIAGRAMS

The HERSS user-software interface was built using the Macintosh menu bar and pull-down menu structure. The first top level drawing in this section corresponds to the Macintosh menu bar. Subsequent diagrams decompose into the related pull-down menus. These pull-down menu options are further decomposed to illustrate the connection between the user-interface and low-level software routines. Software flow diagrams are additionally provided for complicated software routines. Low-level software routines are listed in Appendix B.

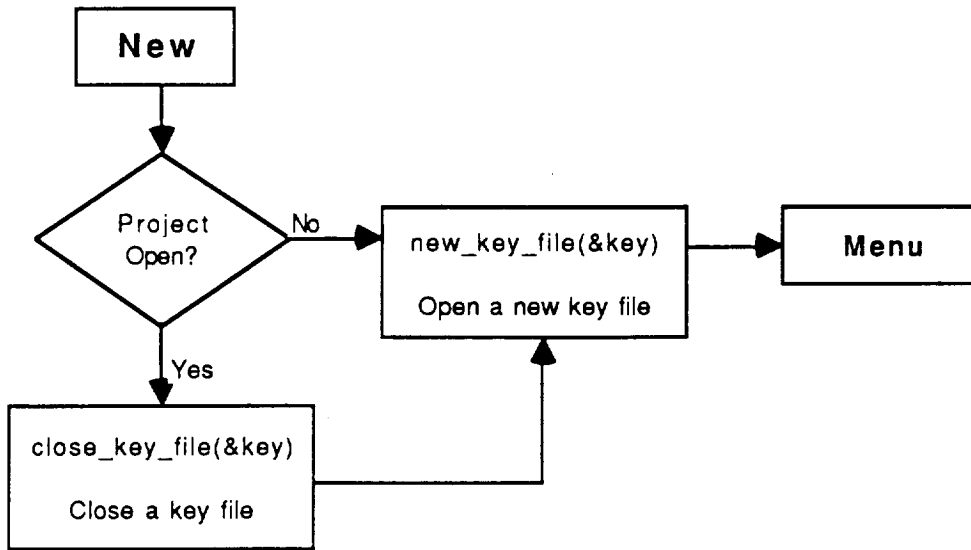
Top Level Drawing



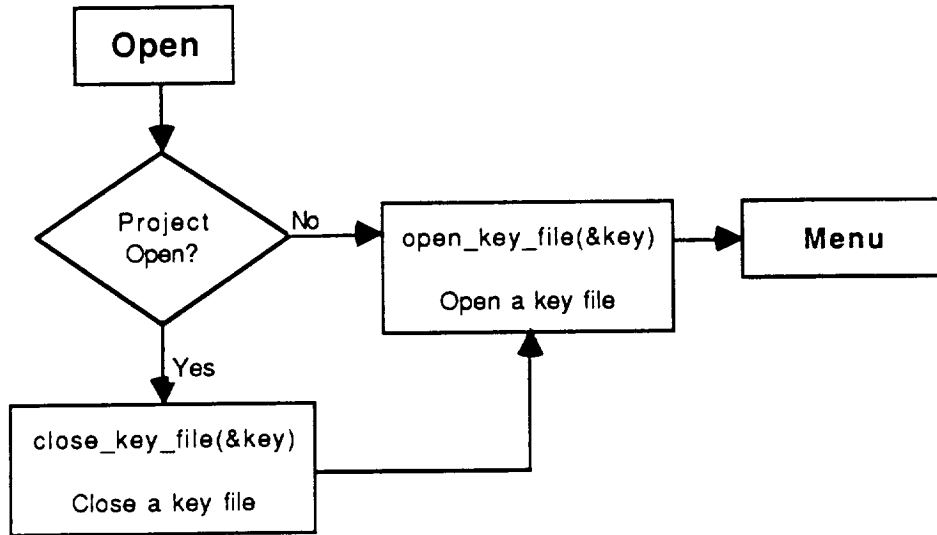
Project



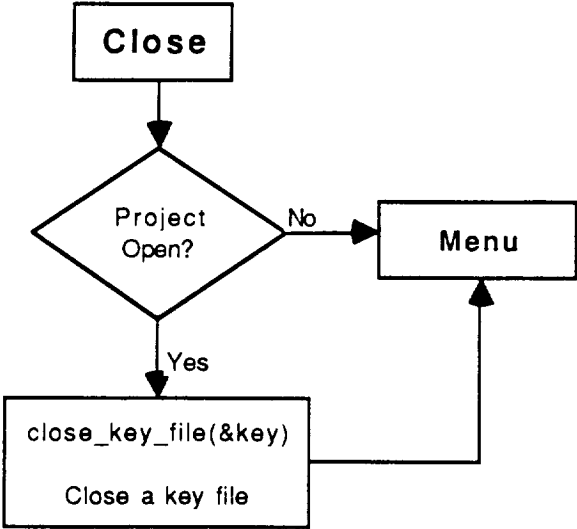
Project.New



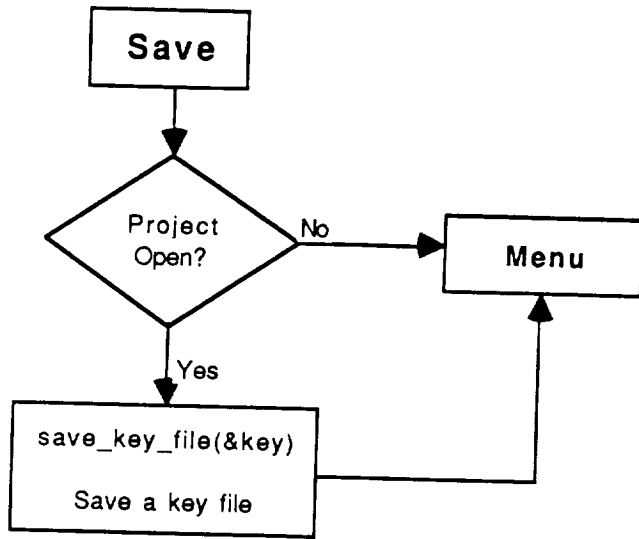
Project.Open



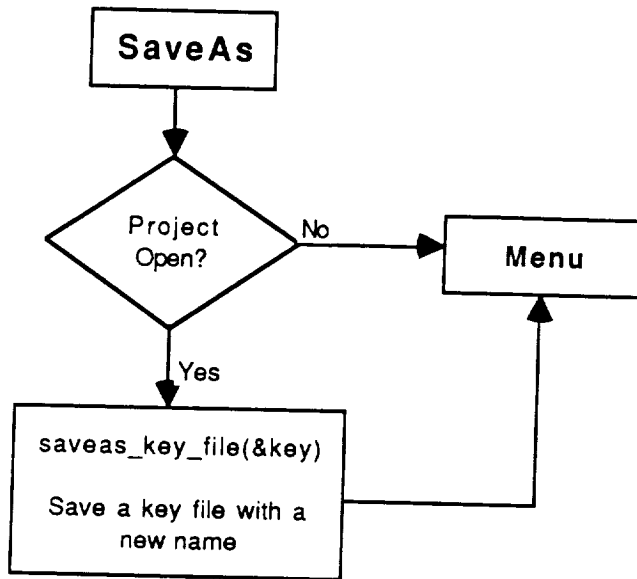
Project.Close



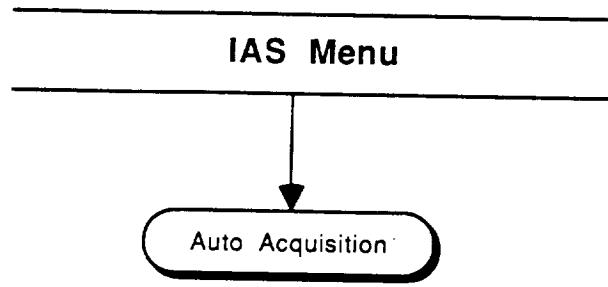
Project.Save

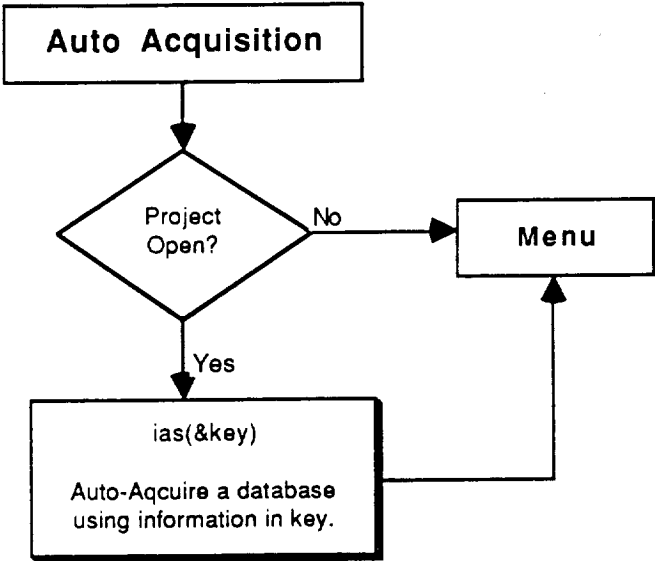


Project.Saveas

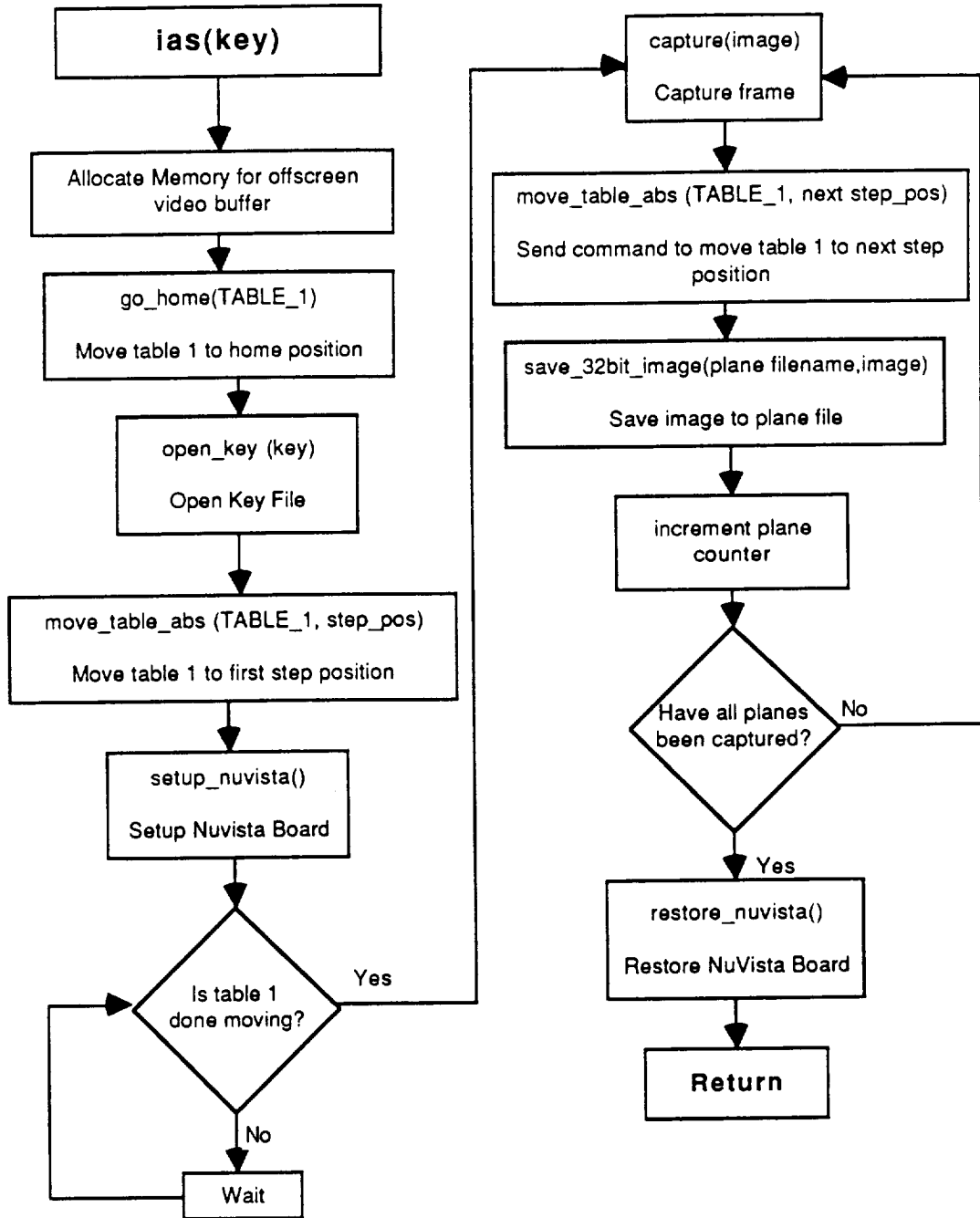


IAS

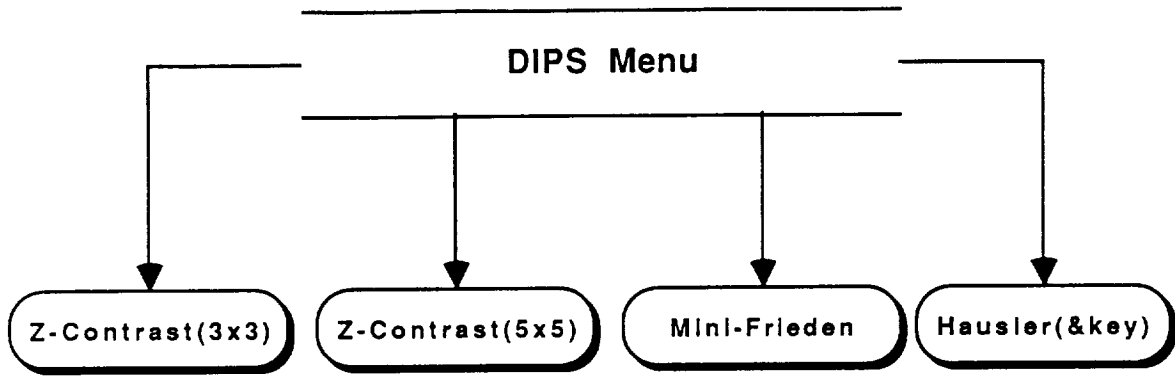




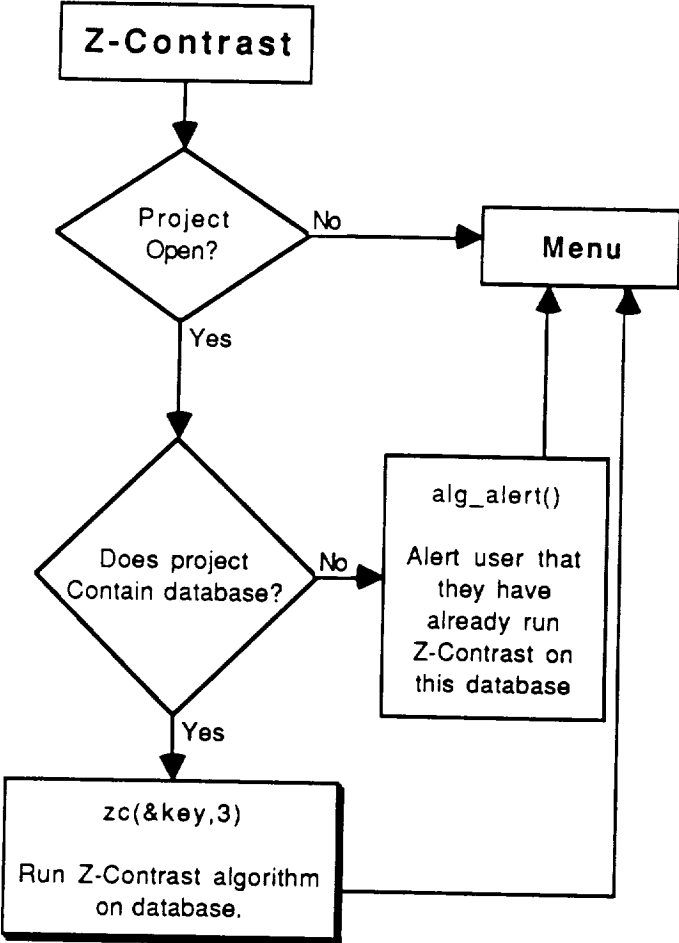
routine.ias

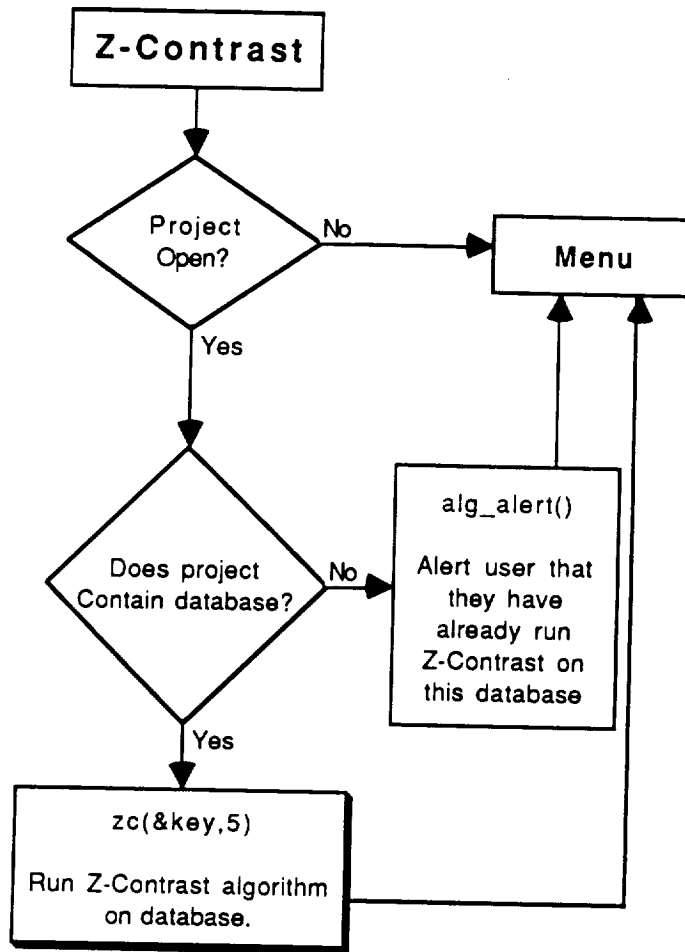


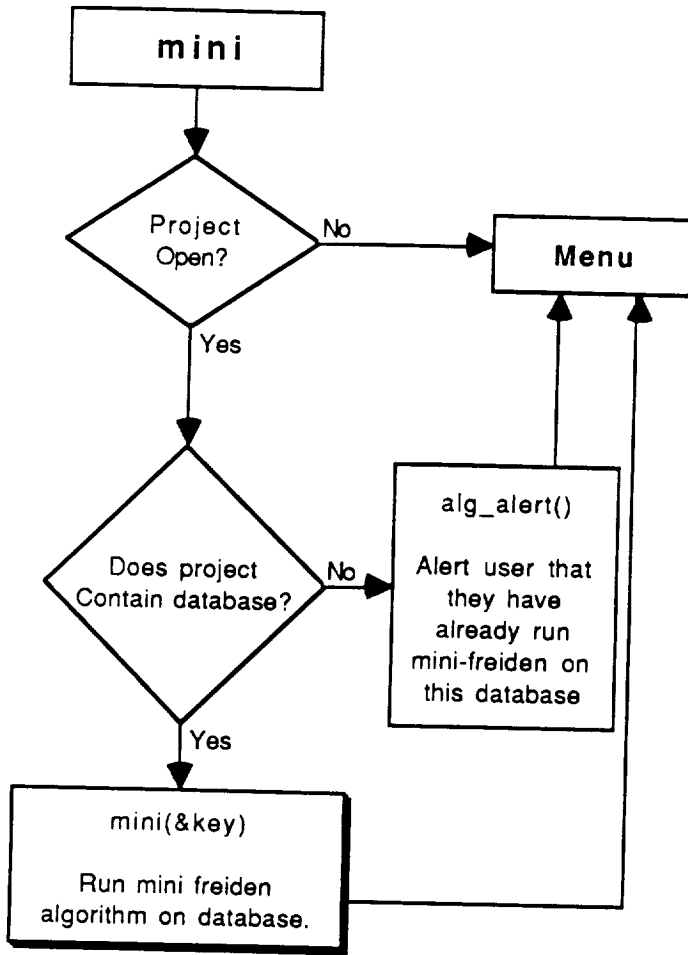
DIPS

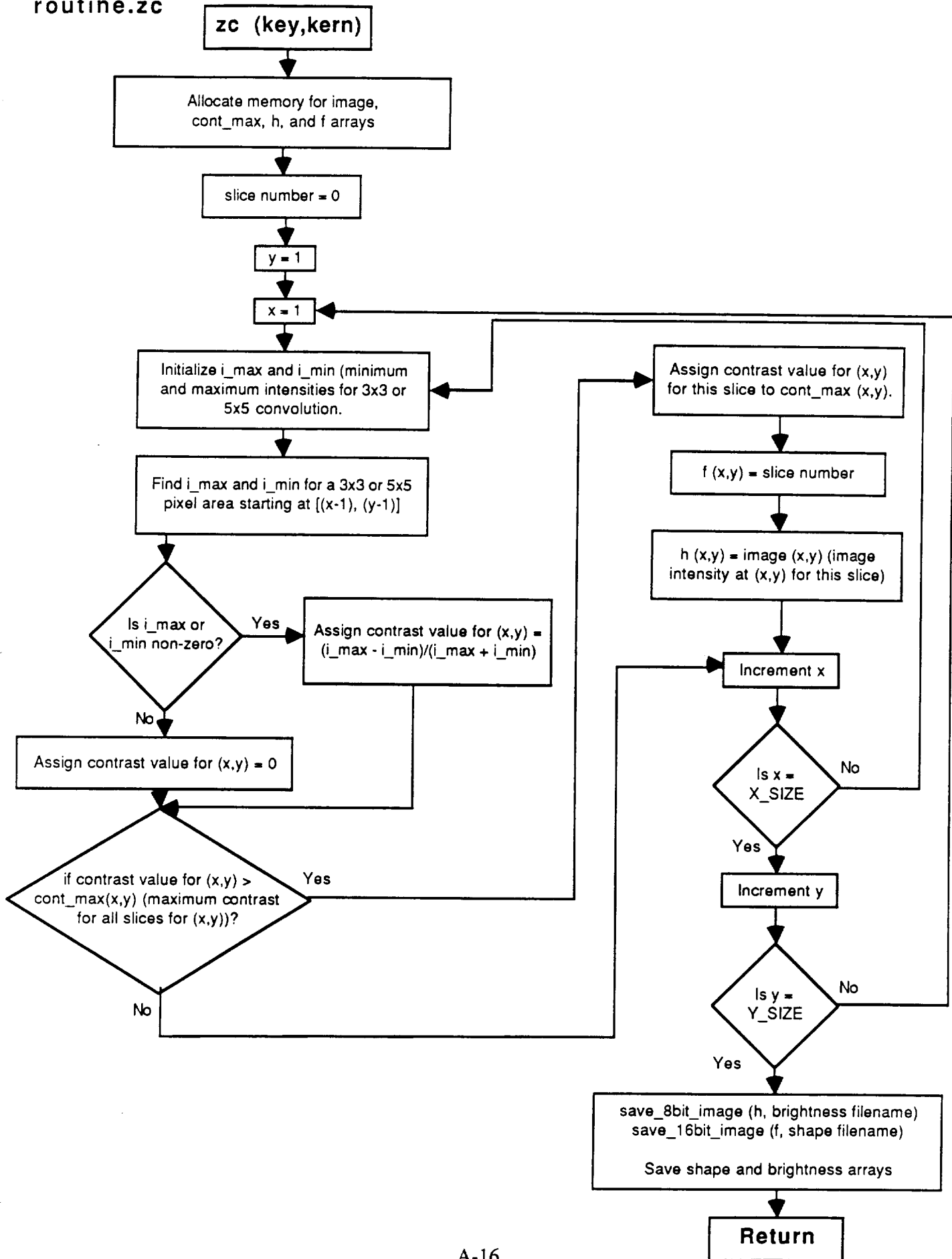


dips.zcontrast(3x3)

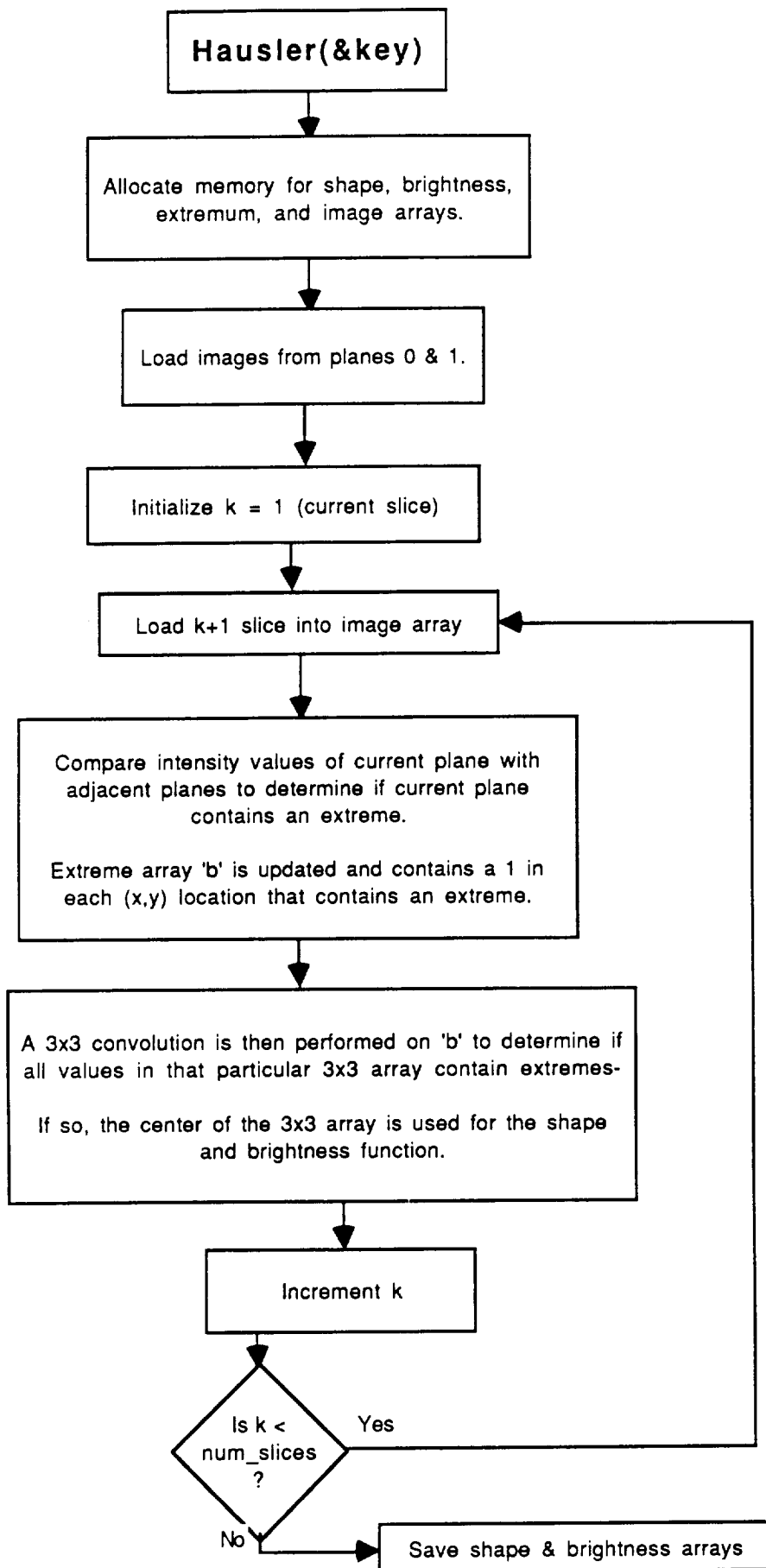




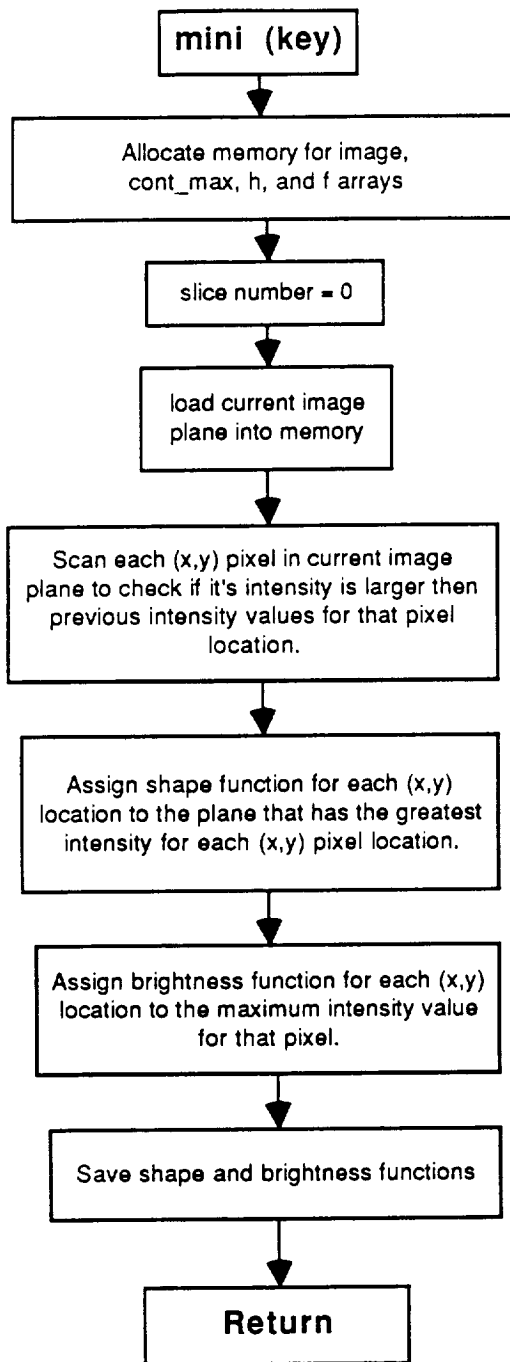




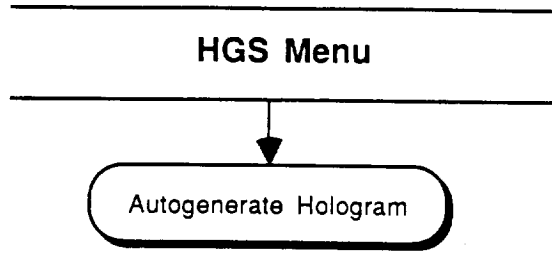
routine.hausler

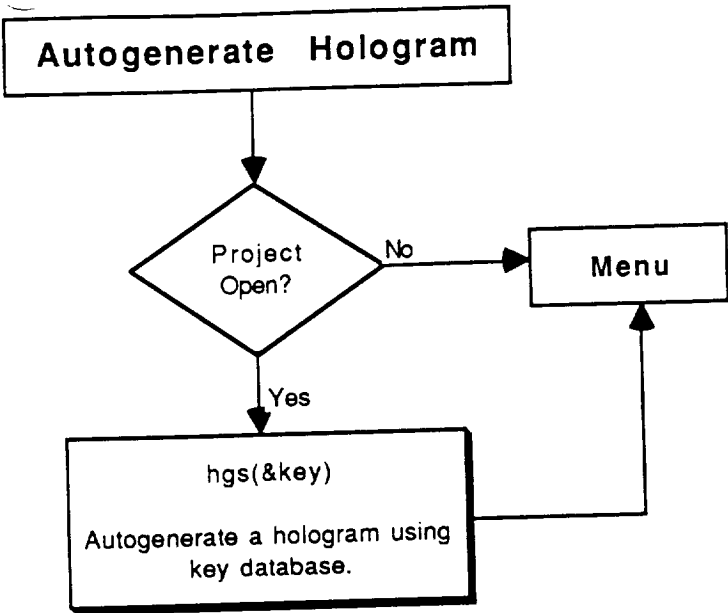


routine.mini

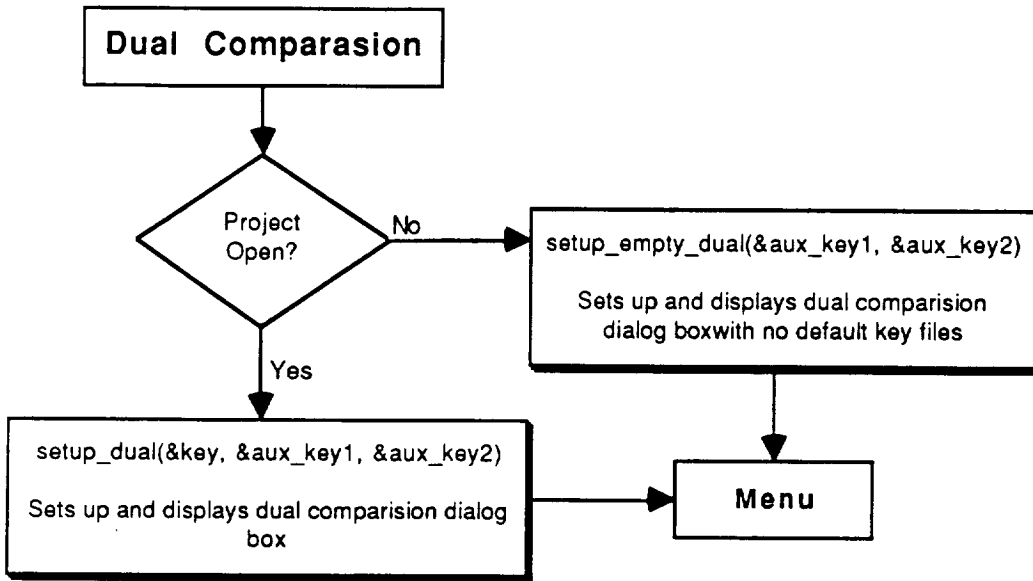


HGS





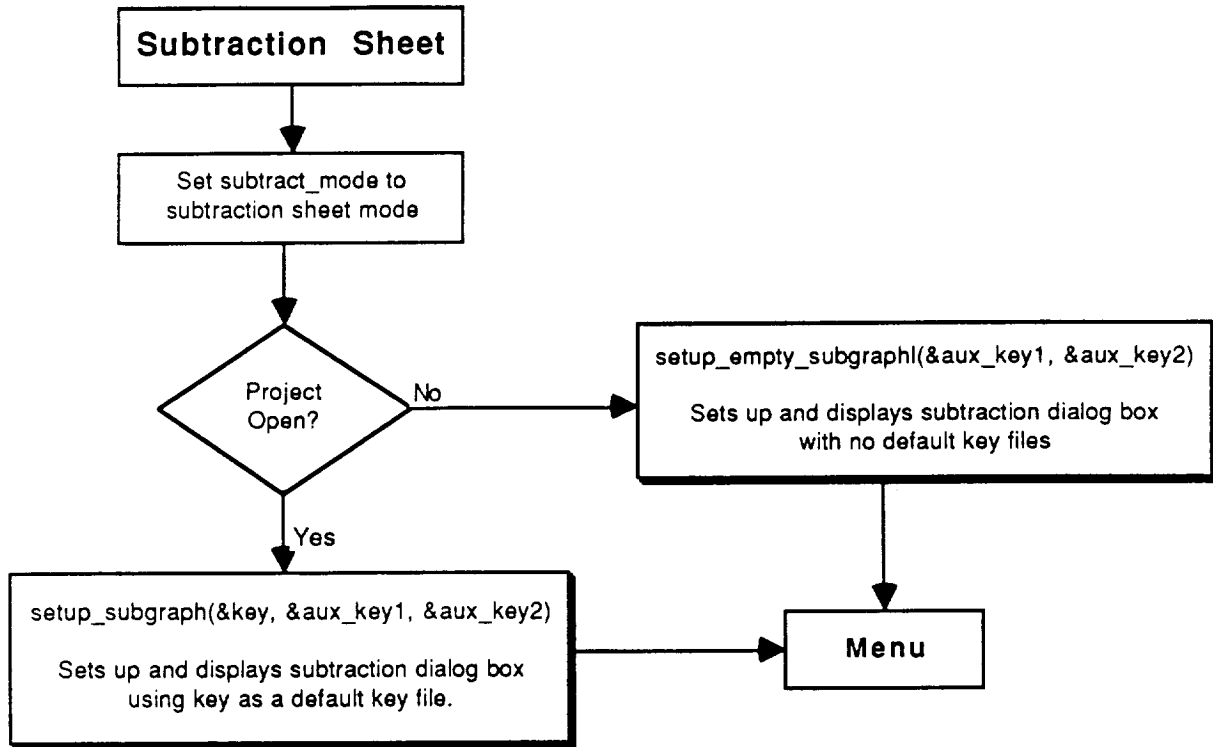
analysis.dualcomp



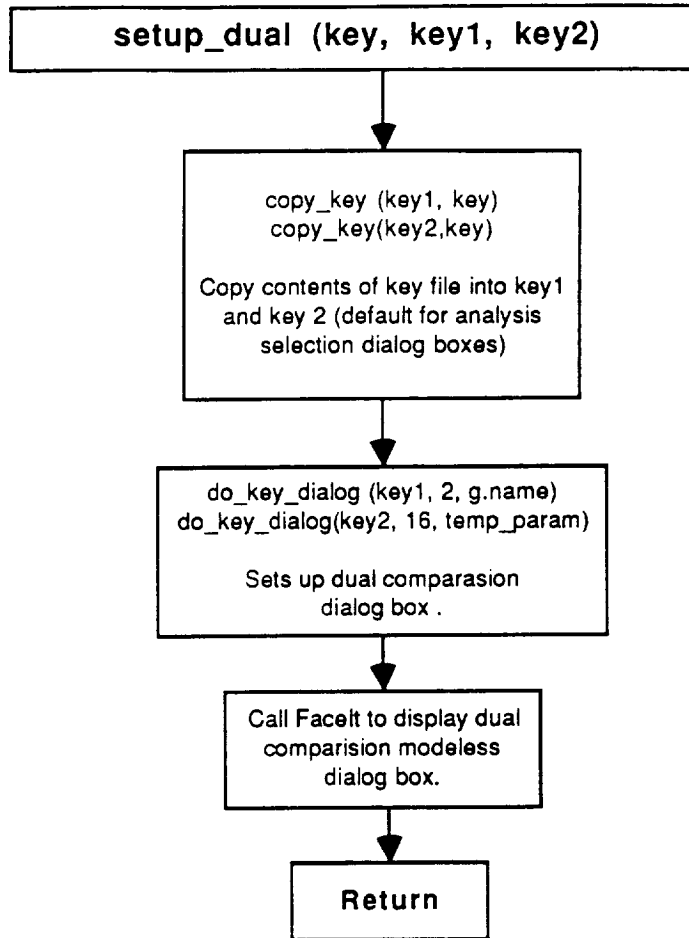
PRECEDING PAGE BLANK NOT FILMED

PRECEDING PAGE BLANK NOT FILMED

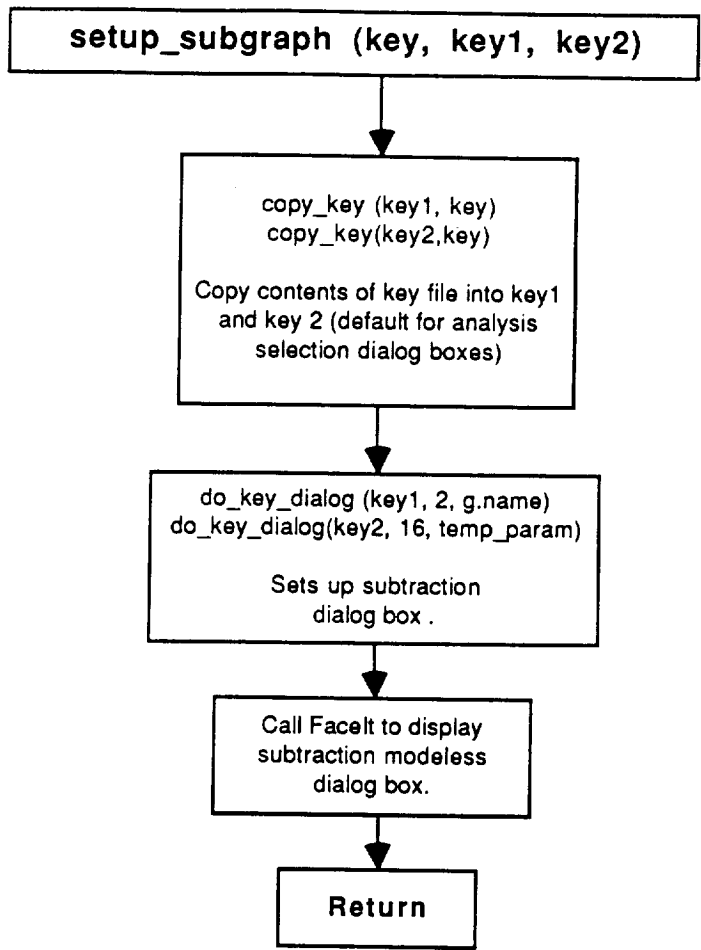
analysis.subsheet



routine.setup_dual

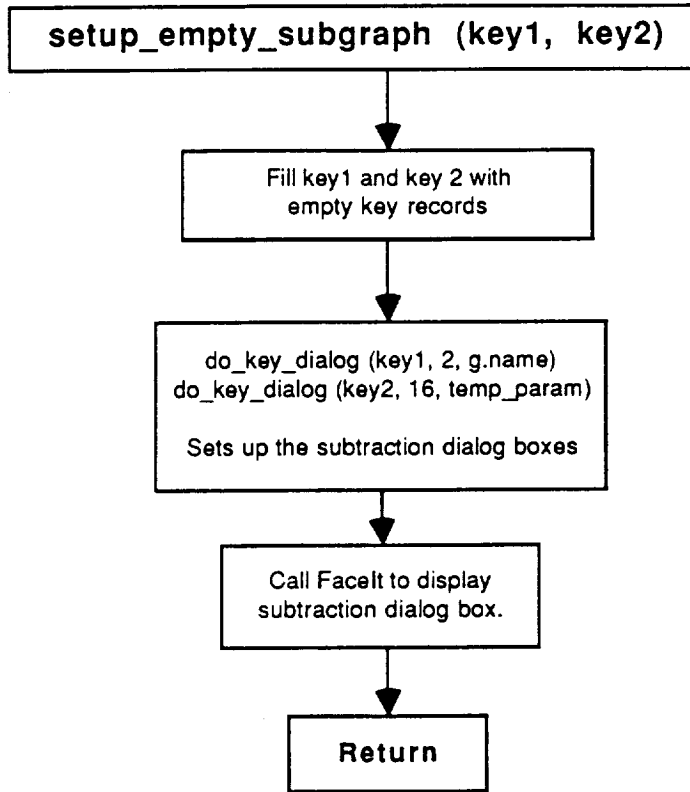


routine.setup_subgraph

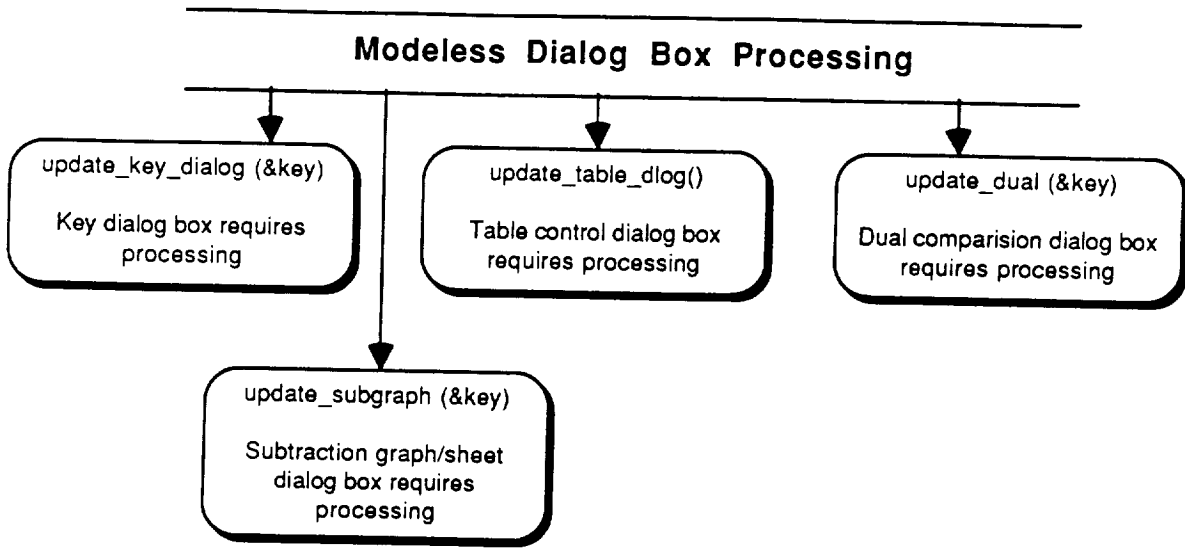


PRECEDING PAGE BLANK NOT FILMED

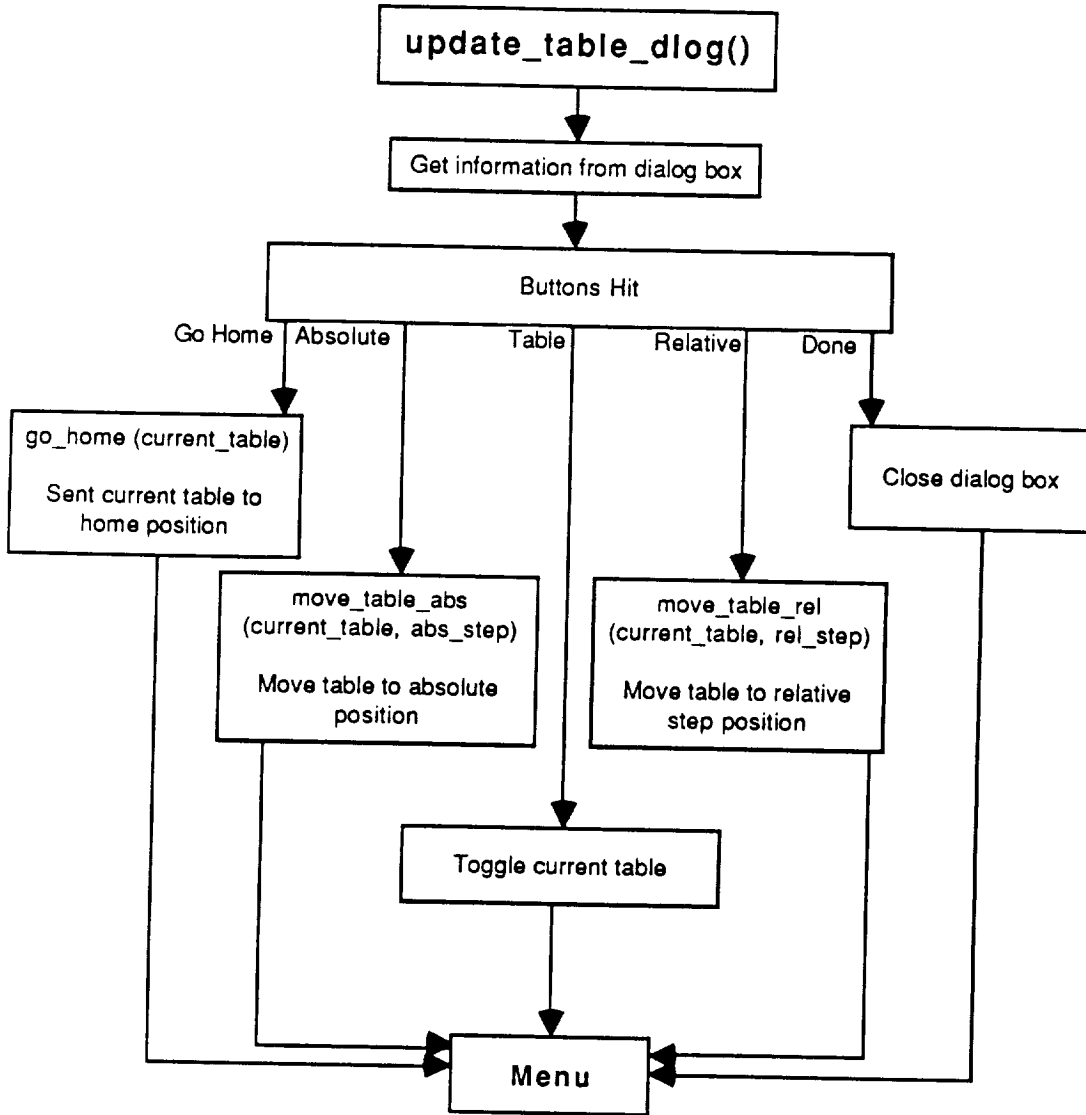
routine.setup_empty_subgraph



modless

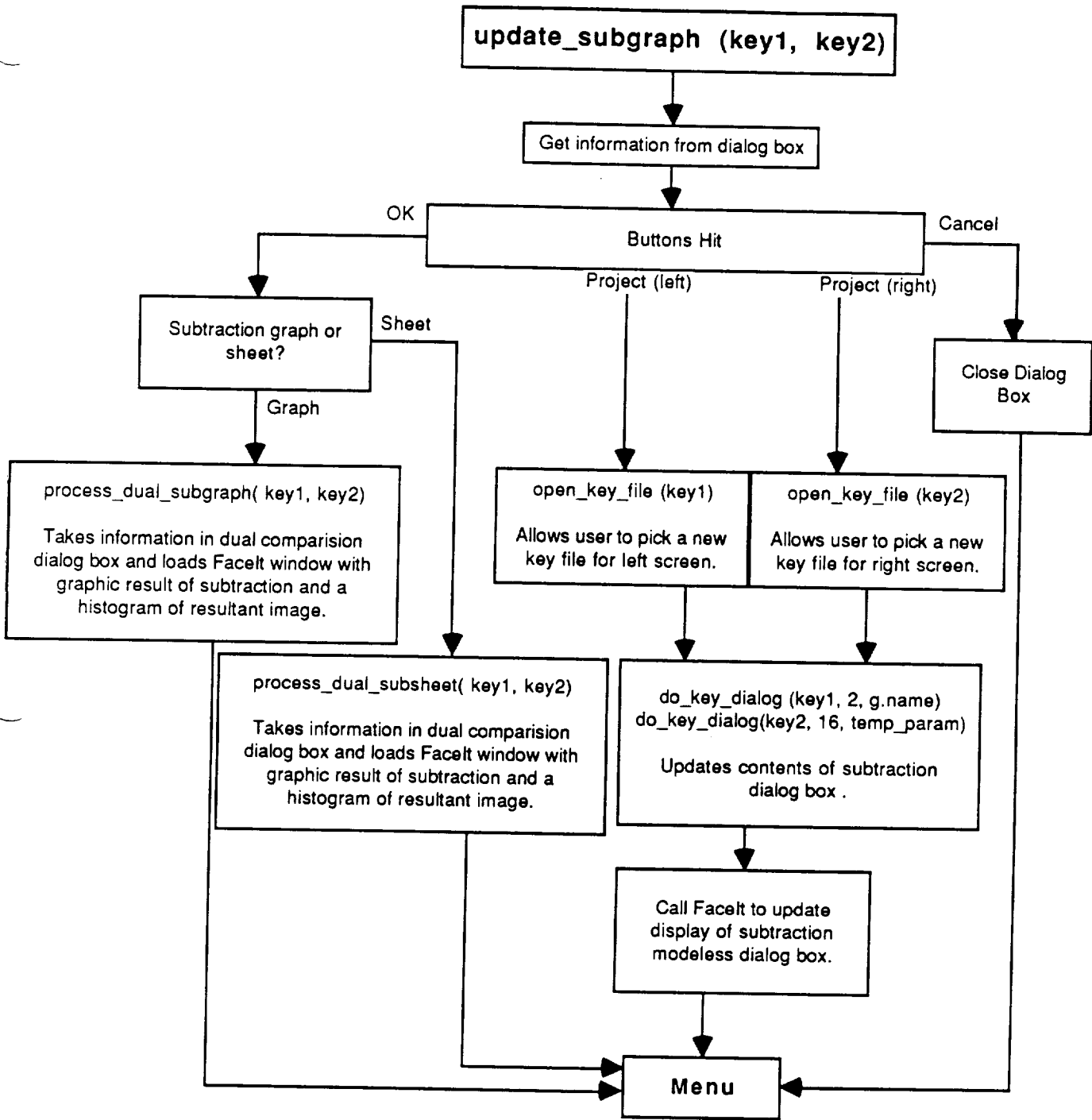


routine.update_table_dlog

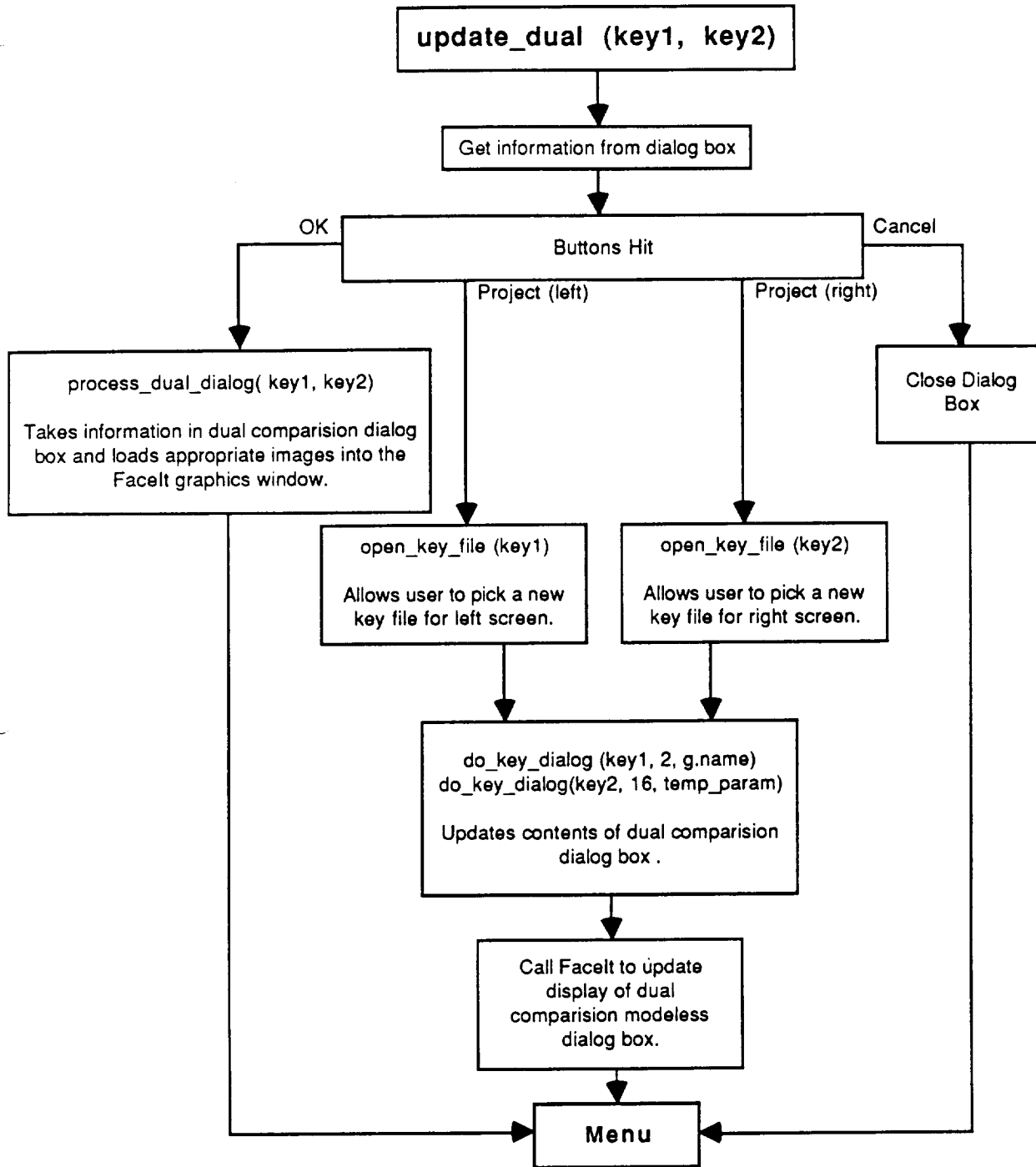


PRECEDING PAGE BLANKS NOT FILLED

routine.update_subgraph



routine.update_dual



Appendix B:
Software Source Code

alg_dialog.c

```
/*
alg_dialog contains misc. dialog utilities used for the DIPS processing.
*/

#include "herrs.h"
#include "FaceLCH.h"

/*
alg_alert lets the user know that they are about to re-run an algorithm on a
database that already had that algorithm run on it.

Input   :   User input if a algorithm should be re-run on a database
Function
Output  :   1 if algorithm should be re-run, 0 otherwise.
*/
alg_alert()

{
g.dialog[0] = 0;
g.dialog[1] = 0;
g.dialog[2] = 0;
FaceIt(0,OpnDlg,1010,0,0,0);
if (g.dialog[0] == 1)
{
FaceIt (0, EndDlg, 0, 0, 0, 0);
return(0);
}
if (g.dialog[1] == 1)
{
FaceIt (0, EndDlg, 0, 0, 0, 0);
return(1);
}
}
```

analysis.c

```
#include "herrs.h"
#include "FaceLCH.h"

extern int subtract_mode;

/*
  histogram takes an image which is of size X_SIZE * Y_SIZE * sizeof (char)
  and
  creates a histogram in the second quadrant of the FaceIt graphics window.

  Input   :   image - pointer to image array of size
X_SIZE*Y_SIZE*sizeof(char)
  Output  :   Histogram of image on 2nd quadrant of FaceIt graphics window.
*/

histogram (image)
  unsigned char *image;

  {
    long *count;
    long i;
    int j;

    if ((count = (long *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(long))) == NULL)
      {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
      }

    for (i=0; i < (X_LSIZE*Y_LSIZE); i++)
      count[(int)image[i]]++;

    PenNormal();
    for (j=0; j < 256; j++)
      {
        MoveTo (j+530, 486);
        LineTo (j+530, 486 - (int)(count[j]/512));
      }
    free (count);
    FaceIt(0,RetCtl,0,0,0,0);
  }

/*
  setup_empty_dual sets up a dual comparision dialog box for two empty key
  files.

  Input   :   key1, key2 - pointers to key_file structures
  Output  :   FaceIt dialog box containing information from key_file
  structures
*/

setup_empty_dual (key1, key2)
  key_file *key1, *key2;

  {
```

analysis.c

```
    char temp_param[40];

    key1->filename[0] = 0;
    key2->filename[0] = 0;
    g.dialog[0] = 0;
    g.dialog[1] = 0;
    g.dialog[2] = 0;

    do_key_dialog( key1, 2, g.name);
    do_key_dialog( key2, 16, temp_param);
    strcat (g.name, temp_param);

    FaceIt(0,SetDlg,1006,0,0,0);
    FaceIt(0,ShoDg4,0,0,0,0);
}

/*
  setup_dual sets up a dual comparision dialog box using key as a template for
  both key files.

  Input   :   key - pointer to key file structure to use as template
  Input   :   key1, key2 - pointers to key_file structures
  Output  :   FaceIt dialog box containing information from key_file
structures
*/

setup_dual (key, key1, key2)
    key_file *key, *key1, *key2;

{
    char temp_param[40];

    copy_key ( key1, key);
    copy_key ( key2, key);
    g.dialog[0] = 0;
    g.dialog[1] = 0;
    g.dialog[2] = 0;

    do_key_dialog( key, 2, g.name);
    do_key_dialog( key, 16, temp_param);
    strcat (g.name, temp_param);

    FaceIt(0,SetDlg,1006,0,0,0);
    FaceIt(0,ShoDg4,0,0,0,0);
}

/*
  update_dual is called to process any events inside of a dual comparision
  dialog
  box.

  Input   :   key1, key2 - pointers to key_file structures
  Input   :   User response to dialog box options - Done, Open Project, etc.
  Output  :   updated dual comparision dialog box
  Output  :   FaceIt graphics window w/ selected images
*/
```

analysis.c

```
update_dual (key1, key2)
    key_file *key1, *key2;
    {
    int dialog_item, proc_slice1, raw_slice1, proc_slice2, raw_slice2, s[21],
        temp_param[40];

    dialog_item = g.dialog[99];
    FaceIt (0, GetDlg, 1006, 0, 0, 0);

    switch (dialog_item)
        {
        case 1 :    process_dual_dialog(key1, key2);
                   break;
        case 2 :
                   FaceIt (0, HidDg4, 0,0,0,0);
                   break;
        case 3 :    open_key_file (key1);
                   do_key_dialog( key1, 2, g.name);
                   do_key_dialog( key2, 16, temp_param);
                   strcat (g.name, temp_param);
                   FaceIt(0,SetDlg,1006,0,0,0);
                   break;
        case 17 :   open_key_file (key2);
                   do_key_dialog( key1, 2, g.name);
                   do_key_dialog( key2, 16, temp_param);
                   strcat (g.name, temp_param);
                   FaceIt(0,SetDlg,1006,0,0,0);
                   break;
        }
    }
}
```

```
/*
process_dual_dialog takes the information in a dual comparison dialog box
and displays the given frames in the FaceIt graphics window.

Input   :   key1, key2 - pointers to key_file structures
Output  :   FaceIt graphics window filled with the two selected image arrays
*/
```

```
process_dual_dialog (key1, key2)
    key_file *key1, *key2;

    {
    char s[21], *image_addr1, *image_addr2;
    int proc_slice1, raw_slice1, proc_slice2, raw_slice2;

    if ((image_addr1 = (char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(char))) ==
        NULL)
        {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
        }
    if ((image_addr2 = (char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(char))) ==
```


analysis.c

```
NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }
    sscanf (g.name, "%5d%5d%20s", &proc_slicel, &raw_slicel, s);
    sscanf (&g.name[30], "%5d%5d%20s",&proc_slice2, &raw_slice2, s);

    get_comp_image (key1, image_addr1, 2, proc_slicel, raw_slicel);
    get_comp_image (key2, image_addr2, 16, proc_slice2, raw_slice2);
    view_2frame (image_addr1, image_addr2);
    free (image_addr1);
    free (image_addr2);
}

/*
  setup_empty_subgraph sets up a subtraction dialog box when there is no key
  file
  in memory.

  Input   :   key1, key2 - pointers to key_file structures
  Output  :   FaceIt dialog box containing information from key_file
  structures
  */
setup_empty_subgraph (key1, key2)
    key_file *key1, *key2;

    {
        char temp_param[40];

        key1->filename[0] = 0;
        key2->filename[0] = 0;
        g.dialog[0] = 0;
        g.dialog[1] = 0;
        g.dialog[2] = 0;

        do_key_dialog( key1, 2, g.name);
        do_key_dialog( key2, 16, temp_param);
        strcat (g.name, temp_param);

        FaceIt(0,SetDlg,1011,0,0,0);
        FaceIt(0,ShoDg5,0,0,0,0);
    }

/*
  setup_subgraph sets up a subtraction dialog box using key as a template for
  both
  key files.

  Input   :   key - pointer to key file structure to use as template
  Input   :   key1, key2 - pointers to key_file structures
  Output  :   FaceIt dialog box containing information from key_file
  structures
  */
setup_subgraph (key, key1, key2)
```

analysis.c

```
key_file *key, *key1, *key2;

{
char temp_param[40];

copy_key ( key1, key);
copy_key ( key2, key);
g.dialog[0] = 0;
g.dialog[1] = 0;
g.dialog[2] = 0;

do_key_dialog( key, 2, g.name);
do_key_dialog( key, 16, temp_param);
strcat (g.name, temp_param);

FaceIt(0,SetDlg,1011,0,0,0);
FaceIt(0,ShoDg5,0,0,0,0);
}
/*
update_subgraph1 is called to process any events inside of a subtraction
dialog
box.
*/

update_subgraph (key1, key2)
key_file *key1, *key2;
{
int dialog_item, proc_slicel, raw_slicel, proc_slice2, raw_slice2, s[21],
temp_param[40];

dialog_item = g.dialog[99];
FaceIt (0, GetDlg, 1011, 0, 0, 0);

switch (dialog_item)
{
case 1 :    if (subtract_mode)
              process_subsheet(key1, key2);
            else
              process_subgraph(key1, key2);
            break;

case 2 :    FaceIt (0, HidDg5, 0,0,0,0);
            break;

case 3 :    open_key_file (key1);
            do_key_dialog( key1, 2, g.name);
            do_key_dialog( key2, 16, temp_param);
            strcat (g.name, temp_param);
            FaceIt(0,SetDlg,1011,0,0,0);
            break;

case 17 :   open_key_file (key2);
            do_key_dialog( key1, 2, g.name);
            do_key_dialog( key2, 16, temp_param);
            strcat (g.name, temp_param);
            FaceIt(0,SetDlg,1011,0,0,0);
            break;
}
}
```

analysis.c

```
/*
  process_subgraph takes the information in a subtraction dialog box and loads
the
  subtraction result into the FaceIt graphics window along with a histogram of
the
  result.
```

```
Input   :   key1, key2 - pointers to key_file structures
Input   :   User response to dialog box options - Done, Open Project, etc.
Output  :   updated subtraction dialog box
Output  :   FaceIt graphics window w/ resultant image & histogram
```

```
*/
```

```
process_subgraph (key1, key2)
  key_file *key1, *key2;
```

```
{
  char s[21];
  unsigned char *image_addr1, *image_addr2;
  int proc_slice1, raw_slice1, proc_slice2, raw_slice2;
  int i;

  if ((image_addr1 = (unsigned char *)calloc((X_LSIZE*Y_LSIZE),
      (long)sizeof(char))) == NULL)
    {
      printf("\nUnable to allocate memory for offscreen buffer\n");
      exit(-1);
    }
  if ((image_addr2 = (unsigned char *)calloc((X_LSIZE*Y_LSIZE),
      (long)sizeof(char))) == NULL)
    {
      printf("\nUnable to allocate memory for offscreen buffer\n");
      exit(-1);
    }
  sscanf (g.name, "%5d%5d%20s", &proc_slice1, &raw_slice1, s);
  sscanf (&g.name[30], "%5d%5d%20s", &proc_slice2, &raw_slice2, s);

  get_comp_image (key1, image_addr1, 2, proc_slice1, raw_slice1);
  get_comp_image (key2, image_addr2, 16, proc_slice2, raw_slice2);

  subtract_ns_image (image_addr1, image_addr2);

  view_frame (image_addr1);
  histogram (image_addr1);
  free (image_addr1);
  free (image_addr2);
}
```

```
/*
```

```
process_subsheets takes the information in a subtraction dialog box and loads
the
  subtraction result into the FaceIt spreadsheet.
```

```
Input   :   key1, key2 - pointers to key_file structures
Input   :   User response to dialog box options - Done, Open Project, etc.
Output  :   updated subtraction dialog box
Output  :   FaceIt spreadsheet window filled w/ subtraction resultants.
```

analysis.c

```
*/

process_subsheet (key1, key2)
    key_file *key1, *key2;

{
    char s[21], *image_addr1, *image_addr2;
    int proc_slicel, raw_slicel, proc_slice2, raw_slice2, first_image,
    second_image;
    int i, *result;

    if ((image_addr1 = (unsigned char *)calloc((X_LSIZE*Y_LSIZE),
        (long)sizeof(char))) == NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }
    if ((image_addr2 = (unsigned char *)calloc((X_LSIZE*Y_LSIZE),
        (long)sizeof(char))) == NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }
    if ((result = (int *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(int))) == NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }
    sscanf (g.name, "%5d%5d%20s", &proc_slicel, &raw_slicel, s);
    sscanf (&g.name[30], "%5d%5d%20s", &proc_slice2, &raw_slice2, s);

    first_image = get_comp_image (key1, image_addr1, 2, proc_slicel,
    raw_slicel);
    second_image = get_comp_image (key2, image_addr2, 16, proc_slice2,
    raw_slice2);

    if (first_image && second_image)
        subtract_16_image (result, image_addr1, image_addr2);
    else
        if (first_image)
            subtract_16_8 (result, image_addr1, image_addr2);
        else
            if (second_image)
                subtract_8_16 (result, image_addr1, image_addr2);
            else
                subtract_8 (result, image_addr1, image_addr2);
    g.arrayptr[0] = result;
    FaceIt(1, SetSh1, Y_SIZE, X_SIZE, 0, 2);
    free (image_addr1);
    free (image_addr2);
}

/*
subtract_ns_image subtracts two unsigned X_SIZE*Y_SIZE character arrays and
scales the result to a unsigned charater format which is then placed in
image_addr1. Note that 0 is offset to equal 128 because of the 8 bit
```

analysis.c

character

limitation. (0 = -127, 128 = 0, 255=128)

Input : image_addr1, image_addr2 - pointers to image arrays of
size X_SIZE*Y_SIZE*sizeof(char)

Output : image_addr1 filled with non-signed result of subtraction

*/

```
subtract_ns_image (image_addr1, image_addr2)
    unsigned char *image_addr1, *image_addr2;
```

```
{
    long i;
    int result;
```

```
    for (i=0; i < (X_LSIZE*Y_LSIZE); i++)
    {
        result = image_addr1[i] - image_addr2[i];
        image_addr1[i] = (unsigned char)((result/2) + 128);
    }
}
```

/*

subtract_8 subtracts two X_SIZE*Y_SIZE character arrays and places the
result

in result, which is a signed integer

Input : image_addr1, image_addr2 - pointers to image arrays of
size X_SIZE*Y_SIZE*sizeof(char)

Input : result - empty image array of size X_SIZE*Y_SIZE*sizeof(int) to
hold result

Output : result filled with signed result of subtraction

*/

```
subtract_8 (result, image_addr1, image_addr2)
    char *image_addr1, *image_addr2;
    int *result;
```

```
{
    long i;
```

```
    for (i=0; i < (X_LSIZE*Y_LSIZE); i++)
        result[i] = (int)image_addr1[i] - (int)image_addr2[i];
}
```

/*

subtract_16_image subtracts two X_SIZE*Y_SIZE integer arrays and places the
result in result, which is a signed integer

Input : image_addr1, image_addr2 - pointers to image arrays of
size X_SIZE*Y_SIZE*sizeof(int)

Input : result - empty image array of size X_SIZE*Y_SIZE*sizeof(int) to
hold result

Output : result filled with signed result of subtraction

*/

```
subtract_16_image (result, image_addr1, image_addr2)
```

analysis.c

```
int *image_addr1, *image_addr2;
int *result;
{
long i;

for (i=0; i < (X_LSIZE*Y_LSIZE); i++)
    result[i] = image_addr1[i] - image_addr2[i];
}

/*
subtract_16_8 subtracts an X_SIZE*Y_SIZE character array from a
X_SIZE*Y_SIZE
integer array and places the result into the result signed integer array.

Input   :   image_addr1 - pointers to image arrays of size
X_SIZE*Y_SIZE*sizeof(int)
           image_addr2 - pointers to image arrays of size
X_SIZE*Y_SIZE*sizeof(char)
Input   :   result - empty image array of size X_SIZE*Y_SIZE*sizeof(int)
           hold result
Output  :   result filled with signed result of subtraction
*/

subtract_16_8 (result, image_addr1, image_addr2)
int *image_addr1;
char *image_addr2;
int *result;

{
long i;

for (i=0; i < (X_LSIZE*Y_LSIZE); i++)
    result[i] = image_addr1[i] - (int)image_addr2[i];
}

/*
subtract_8_16 subtracts an X_SIZE*Y_SIZE integer array from a X_SIZE*Y_SIZE
character array and places the result into the result signed integer array.

Input   :   image_addr1 - pointers to image arrays of size
X_SIZE*Y_SIZE*sizeof(char)
           image_addr2 - pointers to image arrays of size
X_SIZE*Y_SIZE*sizeof(int)
Input   :   result - empty image array of size X_SIZE*Y_SIZE*sizeof(int) to
hold
           result.
Output  :   result filled with signed result of subtraction.
*/

subtract_8_16 (result, image_addr1, image_addr2)
char *image_addr1;
int *image_addr2;
int *result;

{
long i;
```

analysis.c

```
    for (i=0; i < (X_LSIZE*Y_LSIZE); i++)
        result[i] = (int)image_addr1[i] - image_addr2[i];
    }

/*
   get_comp_image uses either a subtraction dialog box or a dual comparasion
   dialog
   box to load an image into dest_addr.

   Input   :   key - pointer to key file structure
   Input   :   dest_addr - pointer to destination image array
   Input   :   base - dialog base for dialog box
   Input   :   proc_slice - processed slice number from dialog box
   Input   :   raw_slice - raw slice number from dialog box
   Output  :   dest_addr filled with appropriate image
*/

get_comp_image(key, dest_addr, base, proc_slice, raw_slice)
key_file *key;
char *dest_addr;
int base;
int proc_slice, raw_slice;

{
    int alg=0, *shape_addr, *bright_addr, return_value = 0;
    long addr;
    char s[21], fn[64];
    float factor;

    if ((g.dialog[base+6]==1) || (g.dialog[base+7]==1) || (g.dialog[base+8]==1))
    {
        while ((g.dialog[base+1+alg] != 1) && (alg < NUM_ALG))
            alg++;
        if (alg >= NUM_ALG)
            return(0);
        if (g.dialog[base+6])
        {
            sprintf(fn, "%s.bright.%d", key->filename, alg);
            load_8bit_image( dest_addr, fn);
        }
        else
            if (g.dialog[base+7])
                if (subtract_mode)
                {
                    return_value = 1;
                    free(dest_addr);
                    sprintf(fn, "%s.shape.%d", key->filename, alg);
                    if ((dest_addr = (int *)calloc((X_LSIZE*Y_LSIZE),
                        (long)sizeof(int))) == NULL)
                    {
                        printf("\nUnable to allocate memory for offscreen
buffer\n");
                        exit(-1);
                    }
                    load_16bit_image (dest_addr, fn);
                }
    }
}
```

analysis.c

```
else
{
sprintf(fn, "%s.shape.%d", key->filename, alg);
if ((shape_addr = (int *)calloc((X_LSIZE*Y_LSIZE),
(long)sizeof(int))) == NULL)
{
printf("\nUnable to allocate memory for offscreen
buffer\n");
exit(-1);
}
load_16bit_image (shape_addr, fn);
factor = (float)key->num_slices / (float)256;
for (addr = 0; addr < (X_LSIZE*Y_LSIZE); addr++)
dest_addr[addr] = (char)(((int)((float)shape_addr[addr]/
factor)) & 0xff);
free (shape_addr);
}
else
if (g.dialog[base+8])
{
if ((shape_addr = (int *)calloc((X_LSIZE*Y_LSIZE),
(long)sizeof(int))) == NULL)
{
printf("\nUnable to allocate memory for offscreen
buffer\n");
exit(-1);
}
if ((bright_addr = (char *)calloc((X_LSIZE*Y_LSIZE),
(long)sizeof(char))) == NULL)
{
printf("\nUnable to allocate memory for offscreen
buffer\n");
exit(-1);
}
sprintf(fn, "%s.shape.%d", key->filename, alg);
load_16bit_image (shape_addr, fn);
sprintf(fn, "%s.bright.%d", key->filename, alg);
load_8bit_image (bright_addr, fn);
get_proc_slice (shape_addr, bright_addr, dest_addr,
proc_slice);
free (shape_addr);
free (bright_addr);
}
}
if (g.dialog[base+9]==1)
{
sprintf(fn, "%s.%d", key->filename, raw_slice);
load_8bit_image( dest_addr, fn);
}
return (return_value);
}

/*
do_key_dialog initializes an analysis dialog box

Input : key - pointer to key file structure
Input : base - dialog base item number
```


analysis.c

Input : param - pointer to hold inputs from dialog box
Output : analysis dialog box filled with key file structure information

*/

do_key_dialog(key, base, param)

key_file *key;

int base;

char *param;

```
{
g.dialog[30] = -3;
g.dialog[31] = -3;
g.dialog[base] = 0;
g.dialog[base+5] = -2;
if (check_alg ( key, base+1))
{
g.dialog[base+6] = 1;
g.dialog[base+7] = 0;
g.dialog[base+8] = 0;
g.dialog[base+9] = 0;
}
else
{
g.dialog[base+6] = -1;
g.dialog[base+7] = -1;
g.dialog[base+8] = -1;
g.dialog[base+9] = 1;
}
g.dialog[base+10] = 0;
g.dialog[base+11] = 0;
g.dialog[base+12] = 0;
g.dialog[base+13] = 0;
sprintf (param, "%-5d%-5d%-20s", 0, 0, key->filename);
}
```

/*

do_blank_dialog initializes a blank analysis dialog box

Input : base - dialog base item number

Input : param - pointer to hold inputs from dialog box

Output : analysis dialog box filled with blank key file structure

information

*/

do_blank_key_dialog(base, param)

char *param;

int base;

```
{
g.dialog[base] = 0;
g.dialog[base+5] = -2;
g.dialog[base+1] = -1;
g.dialog[base+2] = -1;
g.dialog[base+3] = -1;
g.dialog[base+4] = -1;
g.dialog[base+6] = -1;
g.dialog[base+7] = -1;
```

analysis.c

```
g.dialog[base+8] = -1;
g.dialog[base+9] = -1;
g.dialog[base+10] = 0;
g.dialog[base+11] = 0;
g.dialog[base+12] = 0;
g.dialog[base+13] = 0;
sprintf (param, "%-5d%-5d%-20s", 0, 0, " ");
}

/*
copy_key copies a src_key structure into dest_key structure.

Input   :   dest_key, src_key - pointers to key file structures
Output  :   dest_key contains src_key

*/

copy_key (dest_key, src_key)
key_file *src_key, *dest_key;

{
dest_key->xsize = src_key->xsize;
dest_key->>ysize = src_key->>ysize;
dest_key->pix_depth = src_key->pix_depth;
strcpy (dest_key->date, src_key->date);
strcpy (dest_key->time, src_key->time);
strcpy (dest_key->title, src_key->title);
strcpy (dest_key->filename, src_key->filename);
dest_key->num_slices = src_key->num_slices;
dest_key->num_steps = src_key->num_steps;
dest_key->modified = src_key->modified;
dest_key->vol_num = src_key->vol_num;
}
```

capture.c

```
/*
    capture.c
    contains :
    init_nuvista()
    setup_nuvista()
    capture()
    restore_nuvista()

    Compiled using MPW 3.0 C Compiler and linked in with Think C HERSS Modules
*/

#include "NuVistaTools.h"
#include <Resources.h>
#include <Types.h>
#include <Quickdraw.h>
#include <Windows.h>
#include <stdio.h>

#define X_SIZE 512
#define Y_SIZE 486
#define NPRM_RES_ID 264
#define SLOT 10

NVParamBlkHdl nu_vista_statehdl;
GDHandle hNVGDev;
int srcRowBytes;
NVParamBlkHdl paramBlkHdl;
RTblEntryHdl rtblHdl1, rtblHdl2;

/*
    init_nuvista()

    Initializes NuVista Card
*/

init_nuvista()
{
    int res_refnum;

    if (!CheckSlotNuVista(SLOT))
    {
        printf("\nNot a valid slot\n");
        exit(-1);
    }

    SetSlotNuVista(SLOT);
    if( (hNVGDev = GetNVGDev(SLOT)) == (GDHandle)NULL)
    {
        printf("\nError getting Nuvista Graphics Device Handle\n");
        exit(-1);
    }
}
```

capture.c

```
    }

    UseResFile ((res_refnum = OpenResFile("\pNuVistaParameters.rsrc")));
    if ((nu_vista_statehdl = GetResource ('NPRM', NPRM_RES_ID)) ==
(NVParamBlkHdl)0)
    {
        printf("\nCould not load resource\n");
        exit(-1);
    }

    LoadResource( nu_vista_statehdl);
    HNoPurge ( nu_vista_statehdl);
}

/*
    setup_nuvista()

    Sets NuVista card up for frame capture (512x486)
*/

setup_nuvista()
{
    int linesPerRow;
    RTblEntry startValue;
    unsigned short rowStart,rowCnt;
    LUTValueHdl lutBufHdl;
    LUTValue      startColor, endColor;

    HideCursor();

    paramBlkHdl = (NVParamBlkHdl)NewHandle( sizeof(NVParamBlk) );
    SaveNuVistaState( *paramBlkHdl );

    lutBufHdl = NewLUTBuffer(256);
    startColor.argb.alpha = startColor.argb.red = startColor.argb.green =
startColor.argb.blue = 0;
    endColor.argb.alpha = endColor.argb.red = endColor.argb.green =
endColor.argb.blue = 0xff;
    SpreadLUTBuffer( lutBufHdl, 0, 255, startColor, endColor );
    DumpLUTBuffer( 7, 0, lutBufHdl, 256 );
    DisposeLUTBuffer( lutBufHdl );
    FieldWait( 3 );

    LoadNuVistaState( *nu_vista_statehdl );

    rtblHdl1 = NewRTblBuffer( 800 );
    LoadRTblBuffer( 8, rtblHdl1, 800 );

    if( X_SIZE > 512 )
    {
        linesPerRow = 1;
        srcRowBytes = 4096L;
    }
    else
    {
```

capture.c

```
        linesPerRow = 2;
        srcRowBytes = 2048L;
    }
    rtblHdl2 = NewRTblBuffer( 800 );
    startValue.rtbl.row = 1024-16-(Y_SIZE/linesPerRow);
    startValue.rtbl.dsply = 1;
    startValue.rtbl.cap = 1;
    startValue.rtbl.lut = 0;
    startValue.rtbl.tap = 0;
    rowStart = 0;
    rowCnt = Y_SIZE;

    SetRTblBuffer( rtblHdl2, rowStart, rowCnt, startValue, linesPerRow );
    DumpRTblBuffer( 8, rtblHdl2, 800 );
    DisposeRTblBuffer( rtblHdl2 );

    FieldWait( 10 );
}

/*
    capture (destAddr)

    Captures 1 32-bit color frame using NuVista card and current resolution
(512x486)
    Stores result at destAddr
*/

capture(destAddr)
    long *destAddr;

    {
        long *srcAddr;
        unsigned short imrSwixel;

        FieldWait( 10 ); /* Needed to allow frame grabber to settle before capture
*/

        GrabFrame();

        imrSwixel = GetIMRVramSwixel();
        SetIMRVramSwixel( SWIXEL32 );
        srcAddr = (long *) (0xFADFD000);

        CopyRect((long)srcAddr, (long)destAddr, (long)srcRowBytes, (long)srcRowBytes,
(long)(Y_SIZE), (long)X_SIZE);

        SetIMRVramSwixel( imrSwixel );
    }

/*
    restore_nuvista()

    Takes NuVista card out of frame capture mode and sets NuVista card up for
```

capture.c

```
normal  
    use  
*/
```

```
restore_nuvista()  
{  
    LoadNuVistaState( *paramBlkHdl );  
    DumpRTblBuffer( 8, rtblHdl1, 800 );  
    DisposeRTblBuffer( rtblHdl2 );  
    if( paramBlkHdl ) DisposalHandle( paramBlkHdl );  
  
    InvalCLUTs( hNVGDev );  
    InvalGDev( hNVGDev );  
    ShowCursor();  
}
```

convert.c

```
/*
   convert.c

   Contains misc. utility routines used to convert Freiden ASCII
   output files into standard HERSS file formats.
*/

#include "herrs.h"
#include "FaceLCH.h"

#define FREID_X 32
#define FREID_Y 32

convert_freidbase(key)
    key_file *key;

{
    char *dest_addr;
    char fn[64];
    int slice, frame_num;
    FILE *in_file;

    if ((dest_addr = (char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(char))) ==
    NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }

    for (slice = 1; slice <=32; slice++)
    {
        printf("\n\nSlice #%d\n", slice);
        sprintf(fn, "The_Heap:HERSS:16bit IMAGE3D Data:IMAGE3D.%03d", slice);
        load_freid (fn, dest_addr);
        sprintf(fn, "%s.%d", key->filename, slice-1);
        save_8bit_image (dest_addr,fn);
        key->num_slices = 32;
        key->modified = 1;
    }
}

load_freid (fn, dest_addr)
    char *fn, *dest_addr;

{
    FILE *in_file;
    int x, y,i;

    if ( (in_file = fopen (fn, "r")) == NULL)
    {
        printf("\nCan't find it..\n");
        exit(1);
    }
    for ( y = 0; y < FREID_X; y++)
        for ( x = 0; x < FREID_X; x++)
```

convert.c

```
{
fscanf (in_file, "%d,", &i);
if (i < 0)
    i == 0;
dest_addr [ (long)x + (long)y * X_LSIZE] = (char) (i/60);
}
fclose (in_file);
}
```


file.c

```
/*
    file.c

    Contains routines relating to file IO.
*/

#include "herrs.h"
#include "FaceLCH.h"

/*

    new_key_file()

    Fills up a key structure with default values and opens all windows up on
    screen.

    Input   :   new_key - pointer to key structure
    Output  :   new_key - pointer to key structure w/ new default valuesx
*/

void new_key_file( new_key )
    key_file *new_key;

{
    time_t timer;

    timer = time((time_t *)NULL);

    strcpy(new_key->filename, "Untitled");
    new_key->xsize = X_SIZE;
    new_key->ysize = Y_SIZE;
    new_key->pix_depth = DEF_PIX_DEPTH;
    strftime (new_key->date, 9L, "%m/%d/%y\0", localtime(&timer));
    strftime (new_key->time, 9L, "%H:%M:%S:0", localtime(&timer));
    strcpy( new_key->title, "Untitled");
    new_key->num_slices = 0;
    new_key->num_steps = 0;
    new_key->modified = 1;
    new_key->step_pos = (long *)NULL;
    key_dialog(new_key);
    key_notes (new_key);
    key_step (new_key);
}

/*

    open_key (key)

    This routine is the low level key file open routine which merely sets the
    default
    volume to the volume specified in key->vol_num, and reads in the key file
    named
    by key->filename. This routine then reads in all the information into the
    key
    structure.

```

file.c

Input : key - pointer to key structure
Output : key - pointer to key structure filled with values from a key
file

Function

Output : Returns 1 if successful, 0 if unsuccessful.

*/

open_key (key)

key_file *key;

```
{  
FILE *key_in_file;  
OSErr result;
```

```
result = SetVol (NULL, key->vol_num);
```

```
if ((key_in_file = fopen (key->filename, "rb")) == NULL)
```

```
{  
printf("\nKey file not found.\n");  
return (0);  
}
```

```
/* reads in key file header */
```

```
fread (key, KEY_FILE_HEADER, 1, key_in_file);
```

```
key->modified = 0;  
fclose(key_in_file);  
return (1);  
}
```

/*

open_key_file(key)

This routine calls the toolbox routine gets a filename of the key file to
open.

open_key_file then calls open_key to actually open the file.

Input : key - pointer to key structure

Output : key - pointer to key structure filled with values from a key
file

Function

Output : Returns 1 if successful, 0 if unsuccessful.

*/

int open_key_file(key)

key_file *key;

```
{  
Point loc;  
SFReply reply;  
long key_types;  
int status=0;
```

```
key_types = KEY_FILE_TYPE;
```

```
loc.v = 20;
```

```
loc.h = 20;
```

file.c

```
SFGetFile (loc, "\pEnter name of project file", NULL, 1, &key_types, NULL,
           &reply);
if (reply.good)
    {
    PtoCstr (reply.fName);
    strcpy (key->filename, reply.fName);
    key->vol_num = reply.vRefNum;
    status = open_key(key);
    }
return (status);
}

open_key_dlog (key)
key_file *key;
{
if (open_key_file(key))
    {
    key_dialog(key);
    key_notes(key);
    key_step(key);
    return (1);
    }
return(0);
}
/*
close_key_file (key)

This routine checks to see if a project file have been modified since
last
saving. If this is the case, it uses FaceIt to see if the user wants to
save
changes.

Input   :   key - pointer to key structure
Output  :   Returns 1 if close was successful, 0 if unsuccessful.

*/
int close_key_file (key)
key_file *key;

{

g.dialog[0] = 0;
g.dialog[1] = 0;
g.dialog[2] = 0;
g.dialog[3] = 0;
if (key->modified)
    {
    sprintf (g.MAC, "%-64s", key->filename);
    FaceIt(0, OpnDlg, 1001, 0, 0, 0);
    if (g.dialog[0] == 1)
        {
        FaceIt (0, EndDlg, 0, 0, 0, 0);
        save_key_file(key);
        hide_all_key();
        }
    }
}
```

file.c

```
        return(1);
    }
    if (g.dialog[2] == 1)
    {
        FaceIt (0, EndDlg, 0, 0, 0, 0);
        return(0);
    }
}
FaceIt (0, EndDlg, 0, 0, 0, 0);
hide_all_key();
return(1);
}

/*
    save_key(key)

        This is the lower level routine which sets the default volume to
whatever
        is in key->vol_num, and saves the key structure. It also saves the step
        and notes files.

        Input   :   key - pointer to key structure
        Output  :   File containing the key structure

*/

void save_key (key)
    key_file *key;

{
    OSErr  result;
    char   fn[64];

    strcpy (fn, key->filename);
    CtoPstr (fn);
    result = SetVol (NULL, key->vol_num);
    result = Create (fn, key->vol_num, CREATE_FILE_TYPE, KEY_FILE_TYPE);

    save_notes(key);
    save_step(key);
    save_key_struct(key);
}

/*
    save_key_struct  saves the actual key_file data structure to a given file.
    The file should already have been created by higher level Apple toolbox
    routines. (Creat)

        Input   :   key - pointer to key structure
        Output  :   File containing the key structure

*/

save_key_struct (key)
    key_file *key;
{
    FILE    *key_out_file;
```

file.c

```
key_out_file = fopen(key->filename, "rb+");
fwrite (key, KEY_FILE_HEADER, 1, key_out_file);
key->modified = 0;
fclose (key_out_file);
}

/*
save_notes saves the contents of the notes editor window (#1). Uses FaceIt
calls
and uses the key->filename + '.note' as the filename.

Input   :   key - pointer to key structure
Output  :   File containing the notes for the key file
*/

save_notes(key)
    key_file *key;
{
    OSErr result;
    FILE *note_file;

    sprintf(g.name, "%s.note", key->filename);
    if ((note_file = fopen (g.name, "r")) == NULL)
        result = Create (g.name, key->vol_num, CREATE_FILE_TYPE,
TEXT_FILE_TYPE);
    else
        fclose (note_file);
    FaceIt (0, ShoEd1, SelAll, SavFil, RetCtl, 0);
    FaceIt (0, ShoEd1, DesAll, RetCtl, 0, 0);
}

/*
save_step saves the contents of the steps editor window (#2). Uses FaceIt
calls
and uses the key->filename + '.key' as the filename.

Input   :   key - pointer to key structure
Output  :   File containing the step positions for the key file
*/

save_step(key)
    key_file *key;
{
    OSErr result;
    FILE *step_file;

    sprintf(g.name, "%s.step", key->filename);
    if ((step_file = fopen (g.name, "r")) == NULL)
        result = Create (g.name, key->vol_num, CREATE_FILE_TYPE,
TEXT_FILE_TYPE);
    else
        fclose (step_file);
    FaceIt (0, ShoEd2, SelAll, SavFil, RetCtl, 0);
    FaceIt (0, ShoEd2, DesAll, RetCtl, 0, 0);
}

/*
save_key_file(key)
```

file.c

This is the routine which is called to save a key file. It will call `saveas_key_file` if the filename is still `Untitled` (i.e.- it is probably a new file and should be named). It also checks to see if the project has been modified since it's last save, and if it has it will not save it again.

Input : key - pointer to key structure
Output : File containing the key structure

*/

```
save_key_file (key)
    key_file *key;

    {
    if (!key->modified)
        return;
    if (!strcmp(key->filename, "Untitled"))
        saveas_key_file(key);
    else
        save_key (key);
    update_key_dialog(key);
    }
```

/*

```
saveas_key_file(key)
```

This routine uses a toolbox call to get a filename to use to save a key file.

It then stuffs this name in `key->filename` and calls the routine `save_key(key)`.

Input : key - pointer to key structure
New filename to save key file under
Output : File containing the key structure

*/

```
saveas_key_file (key)
    key_file *key;

    {
    Point loc;
    SFReply reply;
    char fn[64];

    strcpy (fn, key->filename);
    CtoPstr (fn);
    loc.v = 10;
    loc.h = 10;
    SFPutFile (loc, "\pEnter name of project file", fn, NULL, &reply);
    if (reply.good)
        {
        key->num_slices = 0;
        PtoCstr (reply.fName);
        strcpy (key->filename, reply.fName);
        key->vol_num = reply.vRefNum;
        }
```

file.c

```
        save_key (key);
    }
    update_key_dialog(key);
}

/*
key_notes loads in the note file into editor #1. If there is no file present
for notes (i.e. - a new project) this routine will open a empty notes window

Input   :   key - pointer to key structure
          File containing the notes for the key file
*/

key_notes (key)
    key_file *key;

{
    OSErr result;
    FILE *note_file;

    sprintf( g.name, "%s.note", key->filename);
    if ((note_file = fopen (g.name, "r")) == NULL)
    {
        FaceIt (0,ShoEd1, SelAll, RetCtl,0, 0);
        FaceIt (0,ShoEd1, ClrSel, RetCtl,0, 0);
    }
    else
    {
        fclose (note_file);
        sprintf (g.name,"%s.note", key->filename);
        FaceIt (0,ShoEd1, SelAll, RetCtl,0, 0);
        FaceIt (0,ShoEd1, ClrSel, RetCtl,0, 0);
        FaceIt (0,ShoEd1, OpnFil, RetCtl,0,0);
    }
}

/*
key_step loads in the step file into editor #2. If there is no file present
for the steps (i.e. - a new project) this routine will open a empty step window.

Input   :   key - pointer to key structure
          File containing the step positions for the key file
*/

key_step (key)
    key_file *key;

{
    OSErr result;
    FILE *step_file;

    sprintf( g.name, "%s.step", key->filename);
    if ((step_file = fopen (g.name, "r")) == NULL)
    {
        FaceIt (0,ShoEd2, SelAll, RetCtl,0, 0);
        FaceIt (0,ShoEd2, ClrSel, RetCtl,0, 0);
    }
}
```

file.c

```
else
{
    fclose (step_file);
    sprintf (g.name, "%s.step", key->filename);
    FaceIt (0, ShoEd2, SelAll, RetCtl, 0, 0);
    FaceIt (0, ShoEd2, ClrSel, RetCtl, 0, 0);
    FaceIt (0, ShoEd2, OpnFil, RetCtl, 0, 0);
}
}

/*
load_steps allocates enough memory to load in the step positions from the
key->filename + '.step' file. This routine should be used before going into
ias, but should not be called until the current step editor window has been
saved.

Input   :   key - pointer to key structure
           File containing the step positions for the key file
Output  :   key->step_pos - Returns pointer to block of memory which
contains
           the step positions stored as long integers.
*/
load_steps(key)
    key_file *key;

{
    FILE *step_file;
    char fn[64];
    int j;
    OSerr result;
    long i;

    key->num_steps = 0;
    result = SetVol (NULL, key->vol_num);
    sprintf (fn, "%s.step", key->filename);
    while ( (step_file = fopen ( fn, "r" )) == NULL)
    {
        printf("\n Can't find %s \n", fn);
        exit(-1);
    }

    while ( (fscanf(step_file, "%ld", &i) != EOF))
        key->num_steps++;

    if ((key->step_pos = (long *) (malloc (sizeof(long) * key->num_steps))) ==
NULL)
    {
        printf("\n Not enough memory\n");
        exit(-1);
    }

    fseek (step_file, 0L, SEEK_SET);
    for (i=0; i < key->num_steps; i++)
        fscanf(step_file, "%ld", &key->step_pos[i] );
    fclose (step_file);
}
}
```


file.c

```
/*
save_image is used to store a offscreen bitmap. src_addr should be an array
of
X_SIZE * Y_SIZE * sizeof (long). This routine will take this array, strip
out
the 3rd byte (the green channel), and stuff this in a new array of
X_SIZE * Y_SIZE * sizeof(char). Once the entire image has been copied to the
new array, this new array is saved to disk under filename.
```

```
Input   :   src_addr - pointer to image array of size
X_SIZE*Y_SIZE*sizeof(long)
          filename - filename of file to contain the image
Output  :   image saved in filename.
*/
```

```
save_image (src_addr, filename)
    long *src_addr;
    char *filename;

    {
    char *dest_addr;
    long addr;
    FILE *pic_file;

    if ((dest_addr = (char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(char))) ==
NULL)
        {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
        }

    for (addr = 0; addr < (X_LSIZE*Y_LSIZE); addr++)
        dest_addr[addr] = (char)(((src_addr[addr] & 0xff00L) >> 8));

    pic_file = fopen (filename, "wb");
    fwrite (dest_addr, (X_LSIZE*Y_LSIZE), 1, pic_file);
    fclose (pic_file);
    free (dest_addr);
    }
/*
```

```
save_8bit_image saves a image array of size X_SIZE * Y_SIZE * sizeof(char)
to a
8 bit image file.
```

```
Input   :   src_addr - pointer to image array of size
X_SIZE*Y_SIZE*sizeof(char)
          filename - filename of file to contain the image
Output  :   image saved in filename.
*/
```

```
save_8bit_image (src_addr, filename)
    char *src_addr;
    char *filename;
```

file.c

```
{
char *dest_addr;
long addr;
FILE *pic_file;

pic_file = fopen (filename, "wb");
fwrite (src_addr, (X_LSIZE*Y_LSIZE), 1, pic_file);
fclose (pic_file);
}

/*
save_8bit_image saves a image array of size X_SIZE * Y_SIZE * sizeof(int) to
a
16 bit image file.

Input   :   src_addr - pointer to image array of size
X_SIZE*Y_SIZE*sizeof(char)
          filename - filename of file to contain the image
Output  :   image saved in filename.
*/

save_16bit_image (src_addr, filename)
int *src_addr;
char *filename;

{
long addr;
FILE *pic_file;

pic_file = fopen (filename, "wb");
fwrite (src_addr, (X_LSIZE*Y_LSIZE)*2, 1, pic_file);
fclose (pic_file);
}

/*
load_8bit_image loads an image of size X_SIZE * Y_SIZE * sizeof(char) into
dest_addr, which should be an array of size X_SIZE * Y_SIZE * sizeof(char).

Input   :   src_addr - pointer to hold image array of size
X_SIZE*Y_SIZE*sizeof(char)
          filename - filename of file that contains the image
Output  :   src_addr - pointer to block of memory filled with image array.
*/

load_8bit_image (dest_addr, filename)
char *dest_addr;
char *filename;

{
FILE *pic_file;

pic_file = fopen (filename, "rb");
fread (dest_addr, (X_LSIZE*Y_LSIZE), 1, pic_file);
fclose (pic_file);
}

/*
load_8bit_image loads an image of size X_SIZE * Y_SIZE * sizeof(char) into
```

file.c

```
dest_addr, which should be an array of size X_SIZE * Y_SIZE * sizeof(long).

Input   :   src_addr - pointer to hold image array of size
            X_SIZE*Y_SIZE*sizeof(long)
            filename - filename of file that contains the image
Output  :   src_addr - pointer to block of memory filled with image array.
*/

load_32bit_image (src_addr, filename)
long *src_addr;
char *filename;

{
char *dest_addr;
long addr;
FILE *pic_file;

if ((dest_addr = (char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(char))) ==
NULL)
{
printf("\nUnable to allocate memory for offscreen buffer\n");
exit(-1);
}
pic_file = fopen (filename, "rb");
fread (dest_addr, (X_LSIZE*Y_LSIZE), 1, pic_file);
fclose (pic_file);

for (addr = 0; addr < (X_LSIZE*Y_LSIZE); addr++)
src_addr[addr] =
(long)(
((long)dest_addr[addr]) & 0xffL );
free (dest_addr);
}

/*
load_bw_image loads a array of X_SIZE * Y_SIZE * sizeof(char) into memory
and
converts that array to a X_SIZE * Y_SIZE * sizeof(long) array suitable for
displaying by the nuvista graphics board. It makes the image a black and
white
image by stuffing one data item from the char array into the 2nd, 3rd & 4th
bytes of the long array. (the R, G, and B channels).

Input   :   src_addr - pointer to hold image array of size
            X_SIZE*Y_SIZE*sizeof(char)
            filename - filename of file that contains the image
Output  :   src_addr - pointer to block of memory filled with image array.
*/

load_bw_image (src_addr, filename)
long *src_addr;
char *filename;

{
char *dest_addr;
long addr;
FILE *pic_file;
```

file.c

```
    if ((dest_addr = (char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(char))) ==
    NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }
    pic_file = fopen (filename, "rb");
    fread (dest_addr, (X_LSIZE*Y_LSIZE), 1, pic_file);
    fclose (pic_file);

    for (addr = 0; addr < (X_LSIZE*Y_LSIZE); addr++)
        src_addr[addr] =
            (long)(
                (((long)dest_addr[addr] << 16) & 0xff0000L) |
                (((long)dest_addr[addr] << 8) & 0xff00L) |
                (((long)dest_addr[addr]) & 0xffL) );
    free (dest_addr);
}

/*
load_16bit_image loads an array of size X_SIZE * Y_SIZE * sizeof(int) into
an array of size X_SIZE * Y_SIZE * sizeof(int).

Input   :   src_addr - pointer to hold image array of size
              X_SIZE*Y_SIZE*sizeof(int)
              filename - filename of file that contains the image
Output  :   src_addr - pointer to block of memory filled with image array.
*/
load_16bit_image (dest_addr, filename)
long *dest_addr;
char *filename;

{
    FILE *pic_file;

    pic_file = fopen (filename, "rb");
    fread (dest_addr, (X_LSIZE*Y_LSIZE)*2L, 1, pic_file);
    fclose (pic_file);
}

/*
cancel

Function
Output :   1 if COMMAND-. is being held down, 0 otherwise.
*/
cancel ()
{
    KeyMap keys;

    GetKeys (&keys);
    return ( (int)((keys.Key[1] & 0x00808000L) == 0x00808000L));
}

/*
Checks to see if a file exists
```

file.c

```
Function
Output : 1 if file exists, 0 otherwise
*/
exist (fn)
char *fn;

{
FILE *check_file;

if ((check_file = fopen (fn, "rb")) == NULL)
return (0);
fclose(check_file);
return(1);
}

/*
check_alg is used to check the current directory for any shape functions
that
were created by the various algorithms. If a particular shape function is
found belonging to an algorithm, the corresponding dialog item is enabled.

Input : key - key file structure
base_dialog - base dialog item number to use to enable
particular
algorithms
Output : contents of dialog box are updated
*/
check_alg (key, base_dialog)
key_file *key;
int base_dialog;

{
int i;
char fn[64];
FILE *check_file;
int alg_found = 0;

SetVol (NULL, key->vol_num);
for (i=0; i<NUM_ALG; i++)
{
sprintf(fn, "%s.shape.%d", key->filename, i);
if ((check_file = fopen (fn, "rb")) == NULL)
g.dialog[i+base_dialog] = -1;
else
{
if (alg_found)
g.dialog[i+base_dialog] = 0;
else
{
alg_found = 1;
g.dialog[i+base_dialog] = 1;
}
}
fclose(check_file);
}
}
```

file.c

```
return(alg_found);  
}
```

graphics.c

```
/*
  graphics.c

  Contains :

      view_frame
*/

#include "herrs.h"
#include "FaceLCH.h"

/*

  view_frame()

  Displays a stored 8-bit picture stored in a block of memory called picture.
  Use defined X_SIZE and Y_SIZE to size picture

*/

view_frame(picture)
  unsigned char *picture;

  {
    long          x,y;
    PixMapHandle  map_hand;
    PixMapPtr     map_ptr,frame_ptr;
    Ptr           frame_addr;
    int           frame_width;

    FaceIt(0,NewPix,0,0,0,0); /* Start creating a new PixMap so FaceIt will
    creat an offscreen buffer */

    /* FaceIt(0,ShoFul,ShoSel,RetCtl,0,0); Display Graphics Window */

    map_hand = g.Coffscrnptr->portPixMap;
    frame_ptr = *map_hand;
    frame_addr = frame_ptr->baseAddr;
    frame_width = ((frame_ptr->rowBytes) & 0x7FFF); /* Fix for Color Grafport */

    g.frame.top=g.frame.left=0;
    g.frame.right=X_SIZE;
    g.frame.bottom=Y_SIZE;

    FaceIt(0,SetPal,1000,2,0,1);

    for (x=g.frame.left; x!=g.frame.right; x++)
    {
      for (y=g.frame.top; y!=g.frame.bottom; y++)
      {
        *(frame_addr+(long)(x-g.frame.left) +
(long)(y-g.frame.top)*frame_width)
          = picture[x + (long)y*X_SIZE];
      }
    }
  }
}
```

graphics.c

```
/*
    view_frame(key)

    Displays a stored 8-bit picture stored in a block of memory called picture.
    Use defined X_SIZE and Y_SIZE to size picture
*/

view_2frame(picture1,picture2)
    char *picture1, *picture2;

    {
        long            x,y;
        PixMapHandle    map_hand;
        PixMapPtr       map_ptr,frame_ptr;
        Ptr              frame_addr;
        int              frame_width;

        FaceIt(0,NewPix,0,0,0,0); /* Start creating a new PixMap so FaceIt will
        creat an offscreen buffer */

        /* FaceIt(0,ShoFul,ShoSel,RetCtl,0,0); Display Graphics Window */

        map_hand = g.Coffscrnptr->portPixMap;
        frame_ptr = *map_hand;
        frame_addr = frame_ptr->baseAddr;
        frame_width = ((frame_ptr->rowBytes) & 0x7FFF); /* Fix for Color Grafport */

        g.frame.top=g.frame.left=0;
        g.frame.right=X_SIZE;
        g.frame.bottom=Y_SIZE;

        FaceIt(0,SetPal,1000,2,0,1);

        for (x=g.frame.left; x!=g.frame.right; x++)
        {
            for (y=g.frame.top; y!=g.frame.bottom; y++)
            {
                *(frame_addr+(long)(x-g.frame.left) +
(long)(y-g.frame.top)*frame_width)
                = picture1[x + (long)y*X_SIZE];
            }
        }

        g.frame.top=0;
        g.frame.left=X_SIZE;
        g.frame.right=2*X_SIZE;
        g.frame.bottom=Y_SIZE;

        for (x=g.frame.left; x!=g.frame.right; x++)
        {
            for (y=g.frame.top; y!=g.frame.bottom; y++)
            {
```


graphics.c

```
        *(frame_addr+(long)(x) + (long)(y-g.frame.top)*frame_width)
          = picture2[x + (long)y*X_LSIZE];
    }
}

FaceIt(0,RetCtl,0,0,0,0); /* Copy Offscreen Bitmap into Screen memory */
}
```

hausler.c

```
/*
    hausler.c
    Contains :
        Hausler restoration algorithm.
*/

#include "herrs.h"
#include "FaceLCH.h"
#include "math.h"

hausler (key)
    key_file    *key;
{
    int product, *f;
    unsigned char *b, *i[3], *i1, *i2, *pic_file;
    unsigned char *h, test;
    char fn[64];
    int x, y, k, diff, last_frame=2, *sign[3], m, n;

    if ((f = (int *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(int))) == NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }

    if ((h = (unsigned char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(unsigned
char))) == NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }
    if ((b = (unsigned char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(unsigned
char))) == NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }
    for (k=0; k<3; k++)
        if ((i[k] = (unsigned char *)calloc((X_LSIZE*Y_LSIZE),
(long)sizeof(unsigned char))) == NULL)
        {
            printf("\nUnable to allocate memory for offscreen buffer\n");
            exit(-1);
        }
    printf("\nBeginning Hausler\n");
    sprintf(fn, "%s.%d", key->filename, 0);
    load_8bit_image (i[0], fn);
    sprintf(fn, "%s.%d", key->filename, 1);
```

hausler.c

```
load_8bit_image (i[1], fn);

for (k = 1; k < key->num_slices-1; k++)
{
    sprintf(fn, "%s.%d", key->filename, k+1);
    load_8bit_image (i[last_frame], fn);
    for (x = 0; x<3; x++)
        for (y=0; y<3; y++)
            h[k+900*x+300*y] = i[2][ref(x+100,y+100)];
/*
    for (x=0; x<X_SIZE; x++)
        for (y=0; y<Y_SIZE; y++)
            {
                switch (last_frame)
                {
                    case 0 :
                        b[ref(x,y)] = (unsigned char)
                            ((i[2][ref(x,y)] > i[0][ref(x,y)]) &&
                             (i[2][ref(x,y)] > i[1][ref(x,y)]));
                        break;
                    case 1 :
                        b[ref(x,y)] = (unsigned char)
                            ((i[0][ref(x,y)] > i[1][ref(x,y)]) &&
                             (i[0][ref(x,y)] > i[2][ref(x,y)]));
                        break;
                    case 2 :
                        b[ref(x,y)] = (unsigned char)
                            ((i[1][ref(x,y)] > i[0][ref(x,y)]) &&
                             (i[1][ref(x,y)] > i[2][ref(x,y)]));
                        break;
                }
            }
    printf("\nSubSlice %d\n", k);
    for (x=1; x<X_SIZE-1; x++)
        for (y=1; y<Y_SIZE-1; y++)
            {
                test = b[ref(x,y)];
                for (m=(x-1); m<=(x+1); m++)
                    for (n=(y-1); n<=(y+1); n++)
                        test *= b[ref(m,n)];
                if ((int)test)
                    {
                        f[ref(x,y)] = k;
                        if (last_frame != 0)
                            h[ref(x,y)] = i[last_frame - 1][ref(x,y)];
                        else
                            h[ref(x,y)] = i[2][ref(x,y)];
                    }
            }

    if (++last_frame >= 3)
        last_frame = 0;*/

}

pic_file = fopen ("test.out", "wb");
fwrite (h, 300*9, 1, pic_file);
fclose (pic_file);
pic_file = fopen ("testtext.out", "w");
```

hausler.c

```
for (k=1; k<key->num_slices-1; k++)
{
  for (x = 0; x<3; x++)
    for (y=0; y<3; y++)
      fprintf(pic_file,"%d\t", h[k+900*x+300*y]);
  fprintf(pic_file,"\n");
}
fclose (pic_file);

/* sprintf( fn, "%s.bright.%d", key->filename, HAUSLER);
save_8bit_image (h, fn);
sprintf( fn, "%s.shape.%d", key->filename, HAUSLER);
save_16bit_image (f, fn);*/

free (h);          /*512*/
free (f);          /*256*/
free (i[0]);       /*256*/
free (i[1]);       /*256*/
free (i[2]);       /*256   3.25megs! */
free (b);          /*256*/
}
```

HERRS.c

```
/* HERRS Project
*/

#include "herrs.h"
#include "herrs.e"
#include "FaceLC.h"
#include "CommIt.h"

key_file    key, aux_key1, aux_key2;
int         proj_open = 0, current_table = TABLE_1, subtract_mode;
long        Comrec[75];
FILE        *out_file;
long        *dest_addr, *src_addr;
long        table_pos[2];

main()
{
    init_nuvista();
    strcpy(g.name, "HERRS.Rsrc");
    FaceIt(0, InitLC, 0, 0, 2, 0);
    g.Fstorage[3] = (long)Comrec;
    init_commit();
    init_table(TABLE_1);
    init_table(TABLE_2);
    configure_labnb();

    while (TRUE)
    {
        FaceIt(0, 0, 0, 0, 0, 0);
        if (strcmp (g.MAC, "keydialog") == 0)
            update_key_dialog(&key);
        if (strcmp (g.MAC, "tabledialog") == 0)
            update_table_dlog();
        if (strcmp (g.MAC, "dualcomp") == 0)
            update_dual(&aux_key1, &aux_key2);
        if (strcmp (g.MAC, "subtract") == 0)
            update_subgraph(&aux_key1, &aux_key2);
        if (strcmp(g.MAC, "About") == 0)
            FaceIt(0, OpnAlt, 1009, 0, 0, 0);
        else
            DoMenus();
    }
    if (proj_open)
        close_key_file (&key);
}

DoMenus()
{
    char fn[64], st[64];
    switch(g.menuID)
    {
        case 105: /* Project */
            switch(g.menuitem)
            {
                case 1:

```

HERRS.c

```
        if (proj_open)
        {
            if (close_key_file (&key))
            {
                new_key_file(&key);
                proj_open = 1;
            }
        }
        else
        {
            new_key_file (&key);
            proj_open=1;
        }
        break;
    case 2:
        if (proj_open)
        {
            if (close_key_file (&key))
                proj_open = open_key_dlog (&key);
        }
        else
            proj_open = open_key_dlog (&key);
        break;
    case 4: if (proj_open)
            proj_open = !close_key_file(&key);
        break;
    case 5: if (proj_open)
            save_key_file(&key);
        break;
    case 6: if (proj_open)
            saveas_key_file(&key);
        break;
    }
    break;

case 106: /* IAS */
    switch(g.menuitem)
    {
        case 1: if (proj_open)
                if (key.num_slices == 0)
                {
                    save_key_file(&key);
                    ias(&key);
                }
                else
                    if (ias_alert())
                    {
                        save_key_file(&key);
                        ias(&key);
                    }
                break;
    }
    break;

case 107: /* DIPS */
    switch(g.menuitem)
    {
```

HERRS.c

```

    case 1:
        if (proj_open)
            {
                sprintf(fn, "%s.shape.%d", key.filename,
ZCONTRAST_3X3);
                if (!exist (fn))
                    zc (&key);
                else
                    if (alg_alert())
                        zc (&key);
            }
        break;
    case 2:
        if (proj_open)
            {
                sprintf(fn, "%s.shape.%d", key.filename,
ZCONTRAST_5X5);
                if (!exist (fn))
                    zc (&key);
                else
                    if (alg_alert())
                        zc (&key);
            }
        break;
    case 3: hausler(&key);
        break;
    case 4: mini (&key);
        break;
    }
break;

case 108: /* HGS */
    switch(g.menuitem)
        {
            case 1:
                hgs(&key);
                break;
        }
    break;

case 109: /* Project Analysis */
    switch(g.menuitem)
        {
            case 1:
                if (proj_open)
                    setup_dual(&key, &aux_key1, &aux_key2);
                else
                    setup_empty_dual (&aux_key1, &aux_key2);
                break;
            case 2:
                subtract_mode = 0;
                if (proj_open)
                    setup_subgraph(&key, &aux_key1, &aux_key2);
                else
                    setup_empty_subgraph (&aux_key1, &aux_key2);
                break;
        }

```

HERRS.c

```
        case 3:
            subtract_mode = 1;
            if (proj_open)
                setup_subgraph(&key, &aux_key1, &aux_key2);
            else
                setup_empty_subgraph (&aux_key1, &aux_key2);
            break;
    }
    break;

case 110: /* Options */
    switch(g.menuitem)
    {
        case 1: do_live();
                break;
        case 2: table_dlog();
                break;
        case 3: piecetog (&key);
                break;
        case 4: convert_scan (&key);
                break;
        case 5: make_data(&key);
                /*load_afc(&key);*/
                break;
        case 6: sitter (&key);
    }
    break;
}
}
```


herrs.h

```
/*
    herrs.h

    Main include file for herrs.c
*/

#include <QuickDraw.h>
#include <WindowMgr.h>
#include <MenuMgr.h>
#include <DialogMgr.h>
#include <EventMgr.h>
#include <ControlMgr.h>
#include <stdlib.h>
#include <time.h>
#include <StdFilePkg.h>
#include <FileMgr.h>
#include <stdio.h>
#include <pascal.h>
#include <String.h>
#include <Color.h>
#include "math.h"
#include "complex.h"

#define X_SIZE 512
#define Y_SIZE 486
#define X_LSIZE 512L
#define Y_LSIZE 486L
#define DEF_PIX_DEPTH 1
#define DEF_MAX_SLICE 250

#define SLM_X_SIZE 1024
#define SLM_Y_SIZE 480

#define CONE_ANG 0
#define PLANE2_RADIUS 5
#define PLANE3_RADIUS 10
#define SINGLE_RAD 0
#define DOUBLE_RAD 1

#define DEF_STEPS 25
#define DEF_AUTO_START 1000L
#define DEF_AUTO_INC 1000L

#define KEY_FILE_TYPE      'KEY!'
#define CREATE_FILE_TYPE  'HOLO'
#define TEXT_FILE_TYPE     'TEXT'

#define TABLE_1 0
#define TABLE_2 1

#define AVERAGE 1

#define SLM_1_ADDR (char *)0xFBB00000L
#define SLM_1 0
```

herrs.h

```
#define SLM_2 1
#define SLM_3 2
#define SLM_4 3

#define ZCONTRAST_3X3 0
#define ZCONTRAST_5X5 1
#define FREIDEN 3
#define HAUSLER 2
#define SERVO_ROBOT 3
#define NUM_ALG 4

typedef struct key_file
{
    int xsize;
    int ysize;
    char pix_depth;
    char date[9];
    char time[9];
    char title[80];
    int num_slices;
    int num_steps;
    char filename[64];      /* Filename */

    int vol_num;           /* Volume reference numbers */
    int modified;         /* This value should normally be a 0, unless
modifications are made to
changes the values of any
*/
    the key data structure, then the routine which
variables in the sturcture should change this to a 1
*/

    long *step_pos; /* array of Max_slice long */
} key_file;

#define KEY_FILE_HEADER (sizeof (key_file) - (sizeof(int)*2) - (sizeof(long)))
#define ref(x,y) ((long)(x) + (long)(y) * (X_LSIZE))
```

hgs.c

```
/*
    /*
    hgs.c
        Contains files relating to the Autogenerate Hologram functions.
*/

#include "herrs.h"
#include "FaceLCH.h"

/*
    hgs is the main routine which displays the processed image slices on the
    SLM, moves the
    table, and triggers the shutter.

    Input   :   key - pointer to key file structure
    Input   :   planes that go with key file structure
    Output  :   planes are displayed on the SLM(s)
*/

int use_mult;
int radius[2] = { PLANE2_RADIUS, PLANE3_RADIUS };

hgs (key)
    key_file *key;

{
    unsigned char *bright_addr, *nv_dest_addr;
    int *shape_addr;
    long *dest_addr;
    int i, slice, alg, slm2_off=1, slm3_off=2, num_sum, set_step, set_slices;
    char fn[64];
    int begin=0, end=0, use_database, use_nv, step=1, position = 0, x, y;

    if ((dest_addr = (long *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(long))) ==
    NULL)
        {
            printf("\nUnable to allocate memory for offscreen buffer\n");
            exit(-1);
        }
    if ((nv_dest_addr = (unsigned char *)calloc((X_LSIZE*Y_LSIZE),
    (long)sizeof(char))) == NULL)
        {
            printf("\nUnable to allocate memory for offscreen buffer\n");
            exit(-1);
        }
    if ((shape_addr = (int *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(int))) ==
    NULL)
        {
```

hgs.c

```
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }
    if ((bright_addr = (char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(char))) ==
    NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }

    move_table_zero (TABLE_2);
    go_home(TABLE_2);
    open_key(key);
    load_steps(key);

    if (hgs_dialog(key, &begin, &end, &step, &alg, &use_database,
    &slm2_off, &slm3_off, &num_sum, &use_mult, &set_step, &set_slices,
    &use_nv))
    {
        if (use_database)
        {
            setup_nuvista();
            sprintf(fn, "%s.shape.%d", key->filename, alg);
            load_16bit_image (shape_addr,fn);
            sprintf(fn, "%s.bright.%d", key->filename, alg);
            load_8bit_image (bright_addr,fn);
        }

        slice = begin;

        while (slice <= end)
        {
            if (cancel ())
                break;
            move_table_abs (TABLE_2, key->step_pos[position++]);
            if (use_database)
            {
                initialize_nvbuf (dest_addr);
                if (CONE_ANG)
                    open_ang (dest_addr, slice, slm2_off, slm3_off, shape_addr,
                    bright_addr, num_sum);
                shape_gen (dest_addr, slice, slm2_off, slm3_off, shape_addr,
                    bright_addr, num_sum);
                display_shape (dest_addr);
            }
            while (!done_move (TABLE_2));
            if (slice == begin)
            {
                SysBeep(10);
                while (Button());
                while (!Button());
                while (Button());
                SysBeep(10);
            }
            trigger_shutter();
            slice += step;
            if (use_mult && !(slice % set_slices))
```

hgs.c

```
        {
            slice -= set_slices;
            slice += set_step;
        }

    }
    if (use_database)
        restore_nuvista();
    }
    free (bright_addr);
    free (shape_addr);
    free (dest_addr);
    free (nv_dest_addr);
}

/*

initialize_nvbuf (buf_addr)

Initializes all three SLMs to opaque (by sending out appropriate
colors to each pixel).

Input   :   buf_addr - frame buffer
Output  :   buf_addr - initialized frame buffer

*/

initialize_nvbuf (buf_addr)
    long *buf_addr;
    {
        long i;

        for (i=0L; i<(X_LSIZE*Y_LSIZE); i++)
            buf_addr[i] = 0xFFFF00L;
    }

/*

open_ang (buf_addr, slice, slm2, slm3, shape, bright, num_sum)

open_ang is an optional feature of the HGS software which will
open cone angles as specified by Lloyd Huff for multiple SLM
cases. The desired cone angle settings are in the "herrs.h"
file as #define's.

Input   :   buf_addr - NuVista frame buffer
Input   :   slice - slice number for SLM1
Input   :   slm2 - slice number of SLM2
Input   :   slm3 - slice number of SLM3
Input   :   shape - should contain shape function for database
Input   :   bright - should contain brightness function
Input   :   num_sum - number of slices to sum
Output  :   buf_addr - holds frame information for the three SLMs
            with angles opened appropriatly.
```

hgs.c

*/

```
open_ang ( buf_addr, slice, slm2, slm3, shape, bright, num_sum)
    long *buf_addr;
    int slice, slm2, slm3, *shape, num_sum;
    char *bright;
```

```
{
    register x, y;
    register shp;
```

```
    for (y=0; y < Y_SIZE; y++)
        for (x = 0; x < X_SIZE; x++)
        {
            shp = shape[ref(x,y)];
            if ((shp >= slice) &&
                (shp < slice + num_sum))
                open_cone (buf_addr, SLM_1, x, y);
            else
                if ((shp >= slice+slm2) &&
                    (shp < slice + slm2 + num_sum))
                    open_cone (buf_addr, SLM_2, x, y);
                else
                    if ((shp >= slice+slm3) &&
                        (shp < slice + slm3 + num_sum))
                            open_cone (buf_addr, SLM_3, x, y);
        }
}
```

/*

```
open_cone (buf_addr, slm, x, y)
```

open_cone is a lower level routine that calls the routines which clear a radius on the appropriate SLMs adjacent to the current SLM.

```
Input   :   buf_addr - destination NuVista frame buffer
Input   :   slm - current slm being processed (1,2,3)
Input   :   x - x pixel coordinate
Input   :   y - y Pixel coordinate
Output  :   buf_addr - will contain the modified frame buffer.
```

*/

```
open_cone (buf_addr, slm, x, y)
    long *buf_addr;
    int slm, x, y;

{

switch (slm)
    {
    case SLM_1 :
        clear_radius (buf_addr, SLM_2, x, y, SINGLE_RAD);
        clear_radius (buf_addr, SLM_3, x, y, DOUBLE_RAD);
        break;
    case SLM_2 :
```

hgs.c

```
        clear_radius (buf_addr, SLM_1, x, y, SINGLE_RAD);
        clear_radius (buf_addr, SLM_3, x, y, SINGLE_RAD);
        break;
    case SLM_3 :
        clear_radius (buf_addr, SLM_1, x, y, DOUBLE_RAD);
        clear_radius (buf_addr, SLM_2, x, y, SINGLE_RAD);
        break;
    }
}

/*

clear_radius (buf_addr, slm, x, y, rad_size)

clear_radius clears the radius on a particular SLM.

Input   :   buf_addr - NuVista frame buffer
Input   :   slm - slm number to clear the radius on
Input   :   (x,y) - coordinates of pixel to clear frame around
Input   :   rad_size - size of radius to clear in pixels
Output  :   buf_addr - Updated with cleared radius on a given SLM.

*/

clear_radius (buf_addr, slm, x, y, rad_size)
    long *buf_addr;
    int slm, x, y, rad_size;

{
    register i, j, min_x, min_y, max_x, max_y;
    register r;

    r = radius [rad_size];
    if ( (min_x = x - r) < 0)
        min_x = 0;
    if ( (max_x = x + r) >= X_SIZE)
        max_x = X_SIZE-1;
    if ( (min_y = y - r) < 0)
        min_y = 0;
    if ( (max_y = y + r) >= Y_SIZE)
        max_y = Y_SIZE-1;

    for (i = min_x; i<max_x; i++)
        for (j = min_y; j < max_y; j++)
            clear_pix (buf_addr, slm, i, j);
}

/*

shape_gen (buf_addr, slice, slm2, slm3, shape, bright, num_sum)

shape_gen displays the auctual image brightness data by using the
shape function.

Input   :   buf_addr - NuVista frame buffer
Input   :   slice - slice to be displayed on SLM1
```

hgs.c

```
Input   :   slm2 - slice to be displayed on SLM2
Input   :   slm3 - slice to be displayed on SLM3
Input   :   shape - shape function for current database
Input   :   bright - brightness function for current database
Input   :   num_sum - number of planes to use in addition to current plane
Output  :   buf_addr - frame buffer containing all 3 SLM information to be
            displayed by the NuVista card.

*/

shape_gen ( buf_addr, slice, slm2, slm3, shape, bright, num_sum)
long *buf_addr;
int slice, slm2, slm3, *shape, num_sum;
char *bright;

{
register x, y;

if (!use_mult)
{
for (y = 0; y < Y_SIZE; y++)
for (x = 0; x < X_SIZE; x++)
if ((shape[ref(x,y)] >= slice) && (shape[ref(x,y)] <
slice + num_sum))
set_l_i (buf_addr, bright[ref(x,y)], x, y);
}
else
{
for (y=0; y < Y_SIZE; y++)
for (x = 0; x < X_SIZE; x++)
{
if ((shape[ref(x,y)] >= slice) &&
(shape[ref(x,y)] < slice + num_sum))
set_inten (buf_addr, SLM_1, bright[ref(x,y)], x, y);
else
if ((shape[ref(x,y)] >= slice+slm2) &&
(shape[ref(x,y)] < slice + slm2 + num_sum))
set_inten (buf_addr, SLM_2, bright[ref(x,y)], x, y);
else
if ((shape[ref(x,y)] >= slice+slm3) &&
(shape[ref(x,y)] < slice + slm3 + num_sum))
set_inten (buf_addr, SLM_3, bright[ref(x,y)], x,
y);
}
}
}

/*

set_inten (buf_addr, slm, inten, x, y)

set_inten sets an pixel intensity value for a given x,y coordinate for a
given plane - for multiple SLM cases only

Input   :   buf_addr - NuVista frame buffer
Input   :   slm - SLM to be displayed on
```


hgs.c

Input : inten - intensity value (256 possible levels)
Input : (x,y) - coordiante to set intensity.
Output : buf_addr - containing modified (x,y) pixel

*/

```
set_inten (buf_addr, slm, inten, x, y)
long *buf_addr;
int slm, x, y;
unsigned char inten;

{
long old_inten;

old_inten = buf_addr[ref(x,y)];
switch (slm)
{
case SLM_1 :
buf_addr [ref(x,y)] = (old_inten & 0x00ffffL) |
(((long)(255 - inten) << 16) & 0xff0000L);
clear_pix (buf_addr, SLM_2, x, y);
clear_pix (buf_addr, SLM_3, x, y);
break;

case SLM_2 :
buf_addr [ref((X_SIZE - x),y)] = (buf_addr [ref((X_SIZE -
x),y)]
& 0xff00ffL) | (((long)(255 - inten) << 8) & 0xff00L);
clear_pix (buf_addr, SLM_1, x, y);
clear_pix (buf_addr, SLM_3, x, y);
break;

case SLM_3 :
buf_addr [ref(x,y)] = (old_inten & 0xffff00L) | (long)inten;
clear_pix (buf_addr, SLM_1, x, y);
clear_pix (buf_addr, SLM_2, x, y);
break;
}
}
```

/*

set_l_i (buf_addr, inten, x, y)

set_l_i is the same as set_inten but for the one SLM case.

Input : buf_addr - NuVista frame buffer
Input : inten - intensity value (256 possible levels)
Input : (x,y) - coordiante to set intensity.
Output : buf_addr - containing modified (x,y) pixel

*/

```
set_l_i (buf_addr, inten, x, y)
long *buf_addr;
int x, y;
```

hgs.c

```
    unsigned char inten;

    {
    long old_inten;

    buf_addr [ref(x,y)] = (((long)(inten))<<16) & 0xff0000L |
                          (((long)(inten))<<8) & 0xff00L |
                          (((long)(255 - inten))) & 0xffL;

/* buf_addr [ref(x,y)] = (((long)(255 - inten))<<16) & 0xff0000L |
                          (((long)(255 - inten))<<8) & 0xff00L |
                          (((long)(inten))) & 0xffL;*/

    }

/*
display_shape (src_addr)

display_shape copies a NuVista frame buffer into the NuVista video ram.

Input   :   src_addr - NuVista frame buffer
Output  :   NuVista frame buffer displayed on NuVista card

*/

display_shape (src_addr)
    long *src_addr;

    {
    long *dest_addr;
    dest_addr = (long *) (0xFADFD000);
    CopyRect(((long)src_addr, (long)dest_addr, (long)2048, (long)2048,
              (long)(Y_SIZE), (long)X_SIZE);
    }

/*
clear_pix (buf_addr, slm, x, y)

clear_pix clears sets an (x,y) pixel on a given slm to a transparent state

Input   :   buf_addr - NuVista frame buffer
Input   :   slm - slm number to clear
Input   :   (x,y) - coordiante to clear
Output  :   modified buf_addr

*/

clear_pix (buf_addr, slm, x, y)
    long *buf_addr;
    int slm, x, y;

    {
    switch (slm)
    {
        case SLM_1:
```

hgs.c

```
        buf_addr [ref(x,y)] &= 0x00ffffL;
        break;
    case SLM_2:
        buf_addr [ref((X_SIZE - x),y)] &= 0xff00ffL;
        break;
    case SLM_3:
        buf_addr [ref(x,y)] |= 0x0000ffL;
        break;
    }
}
```

/*

NVIEW

The following routines are for addressing the nView SLM.

*/

```
display_nv_shape (slm_num, key, slice_num, shape_addr, bright_addr, dest_addr)
```

```
    int slm_num, slice_num;
    unsigned char *dest_addr, *bright_addr;
    int *shape_addr;
    key_file *key;
```

```
    {
        get_proc_slice (shape_addr, bright_addr, dest_addr, slice_num);
        copy_8bit_map (dest_addr, SLM_1_ADDR, X_SIZE, Y_SIZE, SLM_X_SIZE,
SLM_Y_SIZE);
    }
```

```
get_proc_slice (shape_addr, bright_addr, dest_addr, slice_num)
    int *shape_addr, slice_num;
    char *dest_addr, *bright_addr;
```

```
    {
        int x, y;

        for (y = 0; y<Y_SIZE; y++)
            for (x=0; x<X_SIZE; x++)
                if ( shape_addr[ ((long)x + ((long)y)*X_LSIZE)] == slice_num)
                    dest_addr[ ((long)x + ((long)y)*X_LSIZE)] = bright_addr[
((long)x +
                    ((long)y)*X_LSIZE)];
                else
                    dest_addr[ ((long)x + ((long)y)*X_LSIZE)] = (char)0;
    }
```

/*

copy_8bit_map copies an 8 bit image array onto the SLM

```
Input   :   src_addr - pointer to image array which contains source image
Input   :   dest_addr - pointer to image array which source image is copied
into
```

hgs.c

```
Input   :   x_size - number horizontal pixels in the source image
Input   :   y_size - number of vertical pixels in the source image
Input   :   slm_x_size - number of horizontal pixels in the SLM
Input   :   slm_y_size - number of vertical pixels in the SLM
Output  :   dest_addr will contain a copy of the image in src_addr
*/

copy_8bit_map (src_addr, dest_addr, x_size, y_size, slm_x_size, slm_y_size)
char *src_addr, *dest_addr;
int x_size, y_size, slm_x_size, slm_y_size;

{
int max_x, max_y;
register i, j;

if (x_size <= slm_x_size)
    max_x = x_size;
else
    max_x = slm_x_size;

if (y_size <= slm_y_size)
    max_y = y_size;
else
    max_y = slm_y_size;
for (j = 0; j<max_y; j++)
    for (i = 0; i<max_x; i++)
        dest_addr[((long)i + ((long)j) * ((long)slm_x_size))] =
            src_addr[((long)i + (long)j * ((long)x_size))];
}
```

hgs_dialog.c

```
/*
   hgs_dialog.c contains routines relating to the hgs dialog boxes.
*/

#include "herrs.h"
#include "FaceLCH.h"

/*
   hgs_dialog brings up the dialog box to prompt the user for various
information
   needed before starting the hgs routine.

   Input   :   key - pointer to key file structure
   Input   :   User entered dialog box information
   Output  :   begin - slice to begin generating hologram at
   Output  :   end - ending slice
   Output  :   step - number of slices to skip between exposures
   Output  :   alg - algorithm code to use for shape/brightness functions
   Output  :   slm2off - number of slices offset from SLM#1 (for SLM #2)
   Output  :   slm3off - number of slices offset from SLM#1 (for SLM #3)
   Output  :   numsum - number of slices to sum starting with current slice
   Output  :   use_mult - 1 if multiple SLMs should be used
   Output  :   set_step - Total slices to skip in a multi-SLM configuration
   Output  :   set_slices - how many slices are contained in a set
   Output  :   use_nv - 1 if nView SLM should be used
   Output  :   begin - contains starting frame number to be displayed
   Output  :   end - contains last frame number to be displayed
   Output  :   alg - contains which brightness & shape arrays to use
   Output  :   use_database - 1 if a database should be used, 0 otherwise.
*/
```

```
hgs_dialog (key, begin, end, step, alg, use_database,
            slm2off, slm3off, numsum, use_mult, set_step, set_slices,
            use_nv)
key_file *key;
int *begin, *end, *step, *use_nv;
int *alg, *use_database;
int *slm2off, *slm3off, *numsum, *use_mult;
int *set_step, *set_slices;
{
int i, a = 0;
char fn[64];

g.dialog[0] = 0;
g.dialog[1] = -2;
g.dialog[2] = -2;
g.dialog[3] = 0;
g.dialog[4] = 0;
g.dialog[10] = 0;
g.dialog[11] = 1;
g.dialog[12] = -2;
g.dialog[13] = 0;
g.dialog[14] = 0;
g.dialog[15] = -2;
g.dialog[16] = -2;
```

hgs_dialog.c

```
g.dialog[17] = 0;
g.dialog[18] = 0;
g.dialog[19] = -2;
g.dialog[20] = 0;
g.dialog[21] = -2;
g.dialog[22] = -2;
g.dialog[23] = 0;
g.dialog[24] = 0;
g.dialog[25] = 0;

check_alg (key, 5);
if (key->num_slices)
    sprintf (g.name, "%-4d%-4d%-4d%-4d%-4d%-4d%-4d", 0,
key->num_slices-1, 1,0,0,1,0,0);
else
    sprintf (g.name, "%-4d%-4d%-4d%-4d%-4d%-4d%-4d", 0, 0, 1,0,0,1,0,0);

FaceIt(0,OpnDlg,1007,0,0,0);
sscanf (g.name, "%4d%4d%4d%4d%4d%4d%4d", begin, end, step, slm2off,
slm3off,
    numsum, set_step, set_slices);
*use_database = g.dialog[11];
*use_mult = g.dialog[14];
*use_nv = g.dialog[25];
while ((g.dialog[a+5] != 1) && (a < NUM_ALG))
    a++;
*alg = a;

if ((g.dialog[0] == 1) && (a < NUM_ALG))
{
    FaceIt (0, EndDlg, 0, 0, 0, 0);
    return(1);
}
else
    if (g.dialog[0] == 1)
    {
        FaceIt (0, EndDlg, 0, 0, 0, 0);
        return(!(*use_database));
    }
FaceIt (0, EndDlg, 0, 0, 0, 0);

return(0);
}
```

ias.c

/*

ias.c

contains the routines used to acquire an image automatically.

*/

```
#include "herrs.h"
#include "FaceLCH.h"
extern FILE *out_file;
```

/*

ias - main routine which acquires an image database.

Input : key - pointer to key structure containing database information
Input : Captured frames using the NuVista graphics board
Output : Plane files corresponding to each captured frame

*/

ias(key)

key_file *key;

```
{
long *dest_addr_1, *dest_addr_2, *dest_addr_3, step;
char fn[64];
int slice, frame_num;
```

```
if ((dest_addr_1 = (long *)calloc((X_LSIZE*Y_LSIZE),
(long)sizeof(long))) == NULL)
```

```
{
printf("\nUnable to allocate memory for offscreen buffer\n");
exit(-1);
}
```

```
if (AVERAGE > 1)
```

```
if ((dest_addr_2 = (long *)calloc((X_LSIZE*Y_LSIZE),
(long)sizeof(long))) == NULL)
```

```
{
printf("\nUnable to allocate memory for offscreen buffer\n");
exit(-1);
}
```

```
if (AVERAGE > 2)
```

```
if ((dest_addr_3 = (long *)calloc((X_LSIZE*Y_LSIZE),
(long)sizeof(long))) == NULL)
```

```
{
printf("\nUnable to allocate memory for offscreen buffer\n");
exit(-1);
}
```

```
go_home(TABLE_2);
open_key(key);
load_steps(key);
```

ias.c

```
key->num_slices = key->num_steps;
save_key_struct (key);
move_table_abs (TABLE_2, key->step_pos[0]);
setup_nuvista();

for (slice = 0; slice < key->num_steps; slice++)
{
while (!done_move (TABLE_2));
capture(dest_addr_1);
if (AVERAGE > 1)
    capture(dest_addr_2);
if (AVERAGE == 3)
    {
    capture(dest_addr_3);
    average_3 (dest_addr_1, dest_addr_2, dest_addr_3);
    }
if (AVERAGE == 2)
    average_2 (dest_addr_1, dest_addr_2);

if (slice < (key->num_steps-1))
    move_table_abs (TABLE_2, key->step_pos[slice+1]);
sprintf(fn, "%s.%d", key->filename, slice);
save_image (dest_addr_1,fn);
}
if (AVERAGE > 1)
    free(dest_addr_2);
if (AVERAGE > 2)
    free(dest_addr_3);

restore_nuvista();
free (dest_addr_1);
free (key->step_pos);
}

average_2 (addr_1, addr_2)
long *addr_1, *addr_2;

{
int x, y;

for (x=0; x<X_SIZE; x++)
    for (y=0; y<Y_SIZE; y++)
        addr_1[ref(x,y)] = ((addr_1[ref(x,y)] + addr_2[ref(x,y)]) / 2L);
}

average_3 (addr_1, addr_2, addr_3)
long *addr_1, *addr_2, *addr_3;

{
int x, y;

for (x=0; x<X_SIZE; x++)
    for (y=0; y<Y_SIZE; y++)
        addr_1[ref(x,y)] =
            ((addr_1[ref(x,y)] + addr_2[ref(x,y)] + addr_3[ref(x,y)]) / 3L);
}
```


las.c

```
char *c2pstr(word)
char *word;
{
    *CtoPstr(word);
}
```

ias_dialog.c

```
/*
   ias_dialog.c

   contains dialog routines relating to the ias system.
*/

#include "herrs.h"
#include "FaceLCH.h"

/*
   ias_alert displays and alert box if the user is about to re-acquire an image
   database
   over a database which has already been acquired.

   Input   :   Response from user if database should be re-acquired.
   Function
   Output  :   1 if database should be reaquired, 0 if not.
*/
ias_alert()
{
  g.dialog[0] = 0;
  g.dialog[1] = 0;
  g.dialog[2] = 0;
  FaceIt(0,OpnDlg,1008,0,0,0);
  if (g.dialog[0] == 1)
  {
    FaceIt (0, EndDlg, 0, 0, 0, 0);
    return(0);
  }
  if (g.dialog[1] == 1)
  {
    FaceIt (0, EndDlg, 0, 0, 0, 0);
    return(1);
  }
}
```

key_dialog.c

```
/*
    key_dialog.c

    Contains all the routines involved in processing the key_file entry dialog
    box and
    related dialog boxes.
*/

#include "herrs.h"
#include "FaceLCH.h"

/*
    key_dialog brings up the key_file dialog box and fills it with the
    appropriate
    data from the key data structure. This dialog contains such information as
    date
    created, time created, title of project, etc. This should be called
    everytime any
    items in the key data structure are changed.

    Input   : key - pointer to key file structure
    Output  : updated key dialog box according to key file structure
*/

key_dialog (key)
    key_file *key;

{
    int i, done = 0;

    g.dialog[0] = 0;
    g.dialog[1] = 0;
    g.dialog[2] = 0;
    g.dialog[3] = 0;
    g.dialog[4] = 0;
    g.dialog[5] = -2;
    g.dialog[6] = -2;
    g.dialog[7] = -2;
    g.dialog[8] = -2;
    g.dialog[9] = -2;
    g.dialog[10] = 0;
    g.dialog[11] = 0;

    sprintf (g.name, "%-64s%-5d%-8s%-8s%-80s",
             key->filename, key->num_slices, key->date, key->time, key->title);
    FaceIt(0,SetDlg,1002,RetCtl,0,0);
    FaceIt(0,ShoDgl,RetCtl,0,0,0);
}

/*
    update_key_dialog processes the main key dialog box. It reads any
    information
    that may have been changed out of it, and it also checks to see what buttons
    if any have been hit in the dialog box, and calls any routines needed to
    take
    care of these buttons.
*/
```

key_dialog.c

```
Input   :   key - pointer to key file structure
Input   :   Current status of key dialog box
Output  :   updated key file structure according to key dialog box status
*/

update_key_dialog(key)
    key_file *key;

    {
    char s[64];

    if (g.dialog[99] == 12)
        auto_step(key);
    FaceIt (0, GetDlg, 1002, 0, 0, 0);
    strncpy (key->date, &g.name[69], 8);
    strncpy (key->time, &g.name[77], 8);
    strncpy (key->title, &g.name[85], 80);

    key->modified = 1;
    key_dialog(key);
    }

/*
    hide_all_key hides the Key dialog box and the notes and step editor windows.
*/
hide_all_key()
    {
    FaceIt(0, HidDg1, RetCtl, 0, 0, 0);
    FaceIt(0, HidEd1, RetCtl, 0, 0, 0);
    FaceIt(0, HidEd2, RetCtl, 0, 0, 0);
    }

/*
    show_all_key shows the Key dialog box and the notes and step editor windows.
*/
show_all_key()
    {
    FaceIt(0, ShoDg1, RetCtl, 0, 0, 0);
    FaceIt(0, ShoEd1, RetCtl, 0, 0, 0);
    FaceIt(0, ShoEd2, RetCtl, 0, 0, 0);
    }

/*
    auto_step brings up a dialog box which will automatically fill up Editor #2
(step
position window) with step positions according to a start position, end
position,
and number of positionsrequested. All of this data is inputted from the used
by
the dialog box.

    Input   :   key - pointer to key file structure
    Input   :   User inputs from dialog box step, increment and start
    Output  :   Step positions auto-generated from user inputs
*/
auto_step (key)
    key_file *key;
```

key_dialog.c

```
{
long step, increment, start;
int i, num_steps;

g.dialog[0] = 0;
g.dialog[1] = 0;
g.dialog[2] = 0;
g.dialog[3] = 0;
g.dialog[4] = 0;
g.dialog[5] = -2;
g.dialog[6] = -2;
if (key->modified)
{
    sprintf (g.name, "%-7ld%-7ld%-5d", DEF_AUTO_INC, DEF_AUTO_START,
DEF_STEPS);
    FaceIt(0, OpnDlg, 1004, 0, 0, 0);
    sscanf (g.name, "%7ld%7ld%5d", &increment, &start, &num_steps);
    if (g.dialog[0] == 1)
    {
        FaceIt (0, EndDlg, 0, 0, 0, 0);
        for( i=0; i < num_steps; i++)
        {
            sprintf(g.MAC, "%ld", (start + increment * ((long)i)));
            FaceIt (-2, RetCtl, 0, 0, 0, 0);
        }
    }
}
FaceIt (0, EndDlg, 0, 0, 0, 0);
}
```

lab_nb.c

```
/*  
 lab_nb.c    contains all routines directly relating to the lab_nb I/O board.  
*/
```

```
#include <stdio.h>  
extern int LDSysError;
```

```
/*  
    configure_labnb()  
  
    sets up an output port for controlling the shutter, and an input port  
which  
    is used to detect the status of the shutter and the two stepper motors.  
*/
```

```
configure_labnb()  
{  
    DIG_Prt_Config(1, 1, 0, 0);  
    DIG_Prt_Config(1, 0, 1, 0);  
    DIG_Out_Line (1, 0, 0, 0);  

```

```
/*  
    table_active checks either input line #0 or #1, corresponding on table, to  
check  
    to see if a table is done moving or not.
```

```
    Input   :   table - table number  
    Function  
    Output  :   1 if table is still moving, 0 if it has reached its desired  
location.
```

```
*/  
table_active (table)  
    int table;
```

```
{  
    int state;  
  
    DIG_In_Line (1, 1, table, &state);  
    return (!state);  
}
```

```
/*  
    trigger_shutter strobes the output line which controls the shutter  
controller.
```

```
    It then waits until the shutter_open routine returns a 0.
```

```
*/  
trigger_shutter()  
{  
    DIG_Out_Line (1, 0, 0, 1);  
    sleep(1);  
    DIG_Out_Line (1, 0, 0, 0);  
    while (shutter_open());  
}
```

lab_nb.c

```
/*
 shutter_open returns a 1 if the shutter is open, and a 0 if the shutter is
 closed.

 Function
 Output : 1 if shutter is open, 0 if closed
*/
shutter_open()
{
 int state;

 DIG_In_Line (1, 1, 2, &state);
 return (!state);
}
```

laser_data.c

```
/*
    laser_data.c

    Contains :

    convert_scan
    make_data
    load_afc

*/

#define THICK      2
#define WIDTH     10
#define NUM_SLICES 120
#define SIZE      15
#define CHECK     1
#define BACKGND   0
#define INTEN     255

#include "herrs.h"
#include "FaceLCH.h"

/*
    convert_scan(key)

    converts a Servo-Robot ASCII database into standard HERSS shape and
    brightness file formats.

*/

convert_scan (key)
    key_file *key;

{
    unsigned int *i;
    int *f;
    int slice, min_slice, max_slice, x, y, m, n;
    unsigned char fn[64],s[100];
    float contrast, *cont_max;
    unsigned char i_max, i_min, *h;
    long *dest_addr;
    FILE *scan_file;

    if ((i = (int *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(unsigned int))) ==
    NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }

    if ((h = (unsigned char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(unsigned
    char))) == NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }
}
```


laser_data.c

```
if ((f = (int *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(int))) == NULL)
{
    printf("\nUnable to allocate memory for offscreen buffer\n");
    exit(-1);
}

sprintf( fn, "%s.3D", key->filename);
scan_file = fopen (fn, "r");
min_slice = 32767;
max_slice = -32767;
for (y=0;y<63;y++)
{
    for (x=0;x<20;x++)
        fgets (s,100, scan_file);
    for (x=0; x<512; x+=2)
    {
        fscanf (scan_file, "%d", &slice);
        f[ref(x,y*6)] = slice;
        f[ref(x,y*6+1)] = slice;
        f[ref(x,y*6+2)] = slice;
        f[ref(x,y*6+3)] = slice;
        f[ref(x,y*6+4)] = slice;
        f[ref(x,y*6+5)] = slice;
        f[ref(x+1,y*6)] = slice;
        f[ref(x+1,y*6+1)] = slice;
        f[ref(x+1,y*6+2)] = slice;
        f[ref(x+1,y*6+3)] = slice;
        f[ref(x+1,y*6+4)] = slice;
        f[ref(x+1,y*6+5)] = slice;
        if (slice > max_slice)
            max_slice = slice;
        if (slice < min_slice)
            min_slice = slice;
    }
}
for (y=0; y<63*3*2; y++)
    for (x=0; x<256*2; x++)
    {
        f[ref(x,y)] -= min_slice;
        h[ref(x,y)] = (unsigned char)128;
    }
fclose (scan_file);

key->num_slices = key->num_steps = max_slice - min_slice;
save_key_struct (key);

sprintf( fn, "%s.bright.%d", key->filename, 2);
save_8bit_image (h, fn);
sprintf( fn, "%s.shape.%d", key->filename, 2);
save_16bit_image (f, fn);
}

/*
make_data (key)
```

laser_data.c

Utility to make a test hgs database. Uses #define's at top of this file
for setting parameters for the new database.

*/

```
make_data (key)
    key_file *key;

    {
    unsigned int *i;
    int *f, val=0;
    int slice, min_slice, max_slice, x, y, m, n, start;
    unsigned char fn[64],s[100];
    float contrast, *cont_max;
    unsigned char i_max, i_min, *h;
    long *dest_addr;
    FILE *scan_file;

    if ((h = (unsigned char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(unsigned
char))) == NULL)
        {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
        }

    if ((f = (int *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(int))) == NULL)
        {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
        }

    for (y=0; y<486; y++)
        for (x=0; x<512; x++)
            {
            f[ref(x,y)] = NUM_SLICES;
            h[ref(x,y)] = (unsigned char)BACKGND;
            }
        SysBeep(1);
        SysBeep(1);
        for (slice = 0; slice<NUM_SLICES; slice++)
            for (x = ((-1*(WIDTH/2)) - THICK -
slice*THICK);x<(slice*THICK+THICK+(WIDTH/2)); x++)
                for (y=0; y<THICK; y++)
                    {
                    f[ref(244 + x,244 - y - slice*THICK - WIDTH/2)] = NUM_SLICES
- slice - 1;
                    h[ref(244 + x,244 - y - slice*THICK - WIDTH/2)] = (unsigned
char)INTEN;
                    f[ref(244 + x,244 + y + slice*THICK + WIDTH/2)] = NUM_SLICES
- slice - 1;
                    h[ref(244 + x,244 + y + slice*THICK + WIDTH/2)] = (unsigned
char)INTEN;
                    f[ref(244 - y - slice*THICK - WIDTH/2,244 + x)] = NUM_SLICES
- slice - 1;
                    h[ref(244 - y - slice*THICK - WIDTH/2,244 + x)] = (unsigned
```

laser_data.c

```
char)INTEN;
        f[ref(244 + y + slice*THICK + WIDTH/2,244 + x)] = NUM_SLICES
- slice - 1;
        h[ref(244 + y + slice*THICK + WIDTH/2,244 + x)] = (unsigned
char)INTEN;
    }
    start = (WIDTH/2)+(NUM_SLICES*THICK);
    if (CHECK)
        for (x = (-1*start); x<start; x+=SIZE*2)
            for (y = (-1*start); y<start; y+=SIZE*2)
                {
                    for (m = 0; m<SIZE; m++)
                        for (n = 0; n<SIZE; n++)
                            {
                                h[ref(244 + x + m, 244 + y + n)] = (unsigned char) 0;
                                h[ref(244 + x + m + SIZE, 244 + y + n + SIZE)] =
(unsigned char) 0;
                            }
                }

    key->num_slices = key->num_steps = NUM_SLICES;
    save_key_struct (key);

    sprintf( fn, "%s.bright.%d", key->filename, 2);
    save_8bit_image (h, fn);
    sprintf( fn, "%s.shape.%d", key->filename, 2);
    save_16bit_image (f, fn);
}

/*
load_afc(key)

Utilitiy that loads an afc test chart from a raw video frame and stores it as
a brightness function for use in HGS testing

*/

load_afc (key)
    key_file *key;

{
    unsigned int *i;
    int *f, val=0;
    int slice, min_slice, max_slice, x, y, m, n, start;
    unsigned char fn[64],s[100];
    float contrast, *cont_max;
    unsigned char i_max, i_min, *h;
    long *dest_addr;
    FILE *scan_file;

    if ((h = (unsigned char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(unsigned
char))) == NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }
}
```

laser_data.c

```
    }

    if ((f = (int *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(int))) == NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }
    sprintf( fn, "%s.0", key->filename);
    load_8bit_image (h,fn);

    for (y=0; y<486; y++)
        for (x=0; x<512; x++)
            {
                if( h[ref(x,y)] > 192)
                    h[ref(x,y)] = (unsigned char)255;
                else
                    h[ref(x,y)] = 0;
                f[ref(x,y)] = 0;
            }

    key->num_slices = key->num_steps = NUM_SLICES;
    save_key_struct (key);

    sprintf( fn, "%s.bright.%d", key->filename, 1);
    save_8bit_image (h, fn);
    sprintf( fn, "%s.shape.%d", key->filename, 1);
    save_16bit_image (f, fn);
}
```

live.c

```
/*
live.c

do_live goes into live mode to enable focusing of camera
*/

do_live()

{
  setup_nuvista();
  Live();
  while (Button());
  while (!Button());
  while (Button());
  restore_nuvista();
}
```

mini.c

```
/*
    mini.c

    Contains : mini.c
*/

#include "herrs.h"
#include "FaceLCH.h"

/*

    mini(key)

    mini is the mini-frieden maximum intensity algorithm
*/

mini (key)
    key_file *key;

    {
    unsigned char *i;
    int *f;
    int slice, x, y, m, n;
    unsigned char fn[64];
    float contrast, *cont_max;
    unsigned char i_max, i_min, *h;
    long *dest_addr;
    time_t timer;

    if ((i = (unsigned char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(unsigned
char))) == NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }

    if ((cont_max = (float *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(float))) ==
NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }

    if ((h = (unsigned char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(unsigned
char))) == NULL)
    {
        printf("\nUnable to allocate memory for offscreen buffer\n");
        exit(-1);
    }
    }
```

mini.c

```
if ((f = (int *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(int))) == NULL)
{
    printf("\nUnable to allocate memory for offscreen buffer\n");
    exit(-1);
}

for (slice = 0; slice < key->num_slices; slice++)
{
    sprintf(fn, "%s.%d", key->filename, slice);
    load_8bit_image (i, fn);
    printf("\nSlice %d\n", slice);
    for (y = 0; y < (Y_SIZE); y++)
    {
        for (x = 0; x < (X_SIZE); x++)
        {
            if (i[ref(x,y)] > cont_max[ref(x,y)])
            {
                cont_max[ref(x,y)] = i[ref(x,y)];
                f[ref(x,y)] = slice;
                h[ref(x,y)] = i[ref(x,y)];
            }
        }
    }
}

sprintf( fn, "%s.bright.%d", key->filename, FREIDEN);
save_8bit_image (h, fn);
sprintf( fn, "%s.shape.%d", key->filename, FREIDEN);
save_16bit_image (f, fn);
free (cont_max);
free (i);
free (h);
free (f);
}
```

motor.c

/*

motor.c

Contains all routines pertaining to the CompuMotor/Tables

Note : Command to initialize CompuMotor :

1LD0 1MN These settings enable both limit switches

MPA Sets Compumotor to absolute positioning

1001 Sets output port to a high state

1000 Sets output port to low state

1PR Requests position report- Nothing will be sent back after this
command until
table is finished moving.

*/

#include "herrs.h"
#include "FaceLCH.h"
#include "CommIt.h"

extern long table_pos[2];
extern long Comrec;

/*

init_commit

Calls CommIt initialization procedures to initialize the printer and
modem port.

*/

init_commit()

```
{  
FaceIt (0, CoInit, 0, 0, 0, 0);  
FaceIt (0, CmDflt, 0, 0, 0, 0);  
FaceIt (0, CpDflt, 0, 0, 0, 0);  
  
}
```

/*

sends contents of g.MAC to either printer or serial port (depending on table

-

TABLE_1 = Modem, TABLE_2 = Printer

Input : table - table number to send command to
g.MAC : contains command to send

*/

motor.c

```
send_command(table)
    int table;

    {
    if (table)
        FaceIt (0, CpDoIO, -1, 0, 0, 0);
    else
        FaceIt (0, CmDoIO, -1, 0, 0, 0);
    }

/*
    Reads input buffer from either printer or serial port depending on table,
    and puts the
    characters recieved into g.MAC

    Input   :   table - table number to use
    Output  :   g.MAC - characters read from table
*/

get_command(table)
    int table;

    {
    if (table)
        FaceIt (0, CpDoIO, 0, -1, 0, 0);
    else
        FaceIt (0, CmDoIO, 0, -1, 0, 0);
    }

/*
    init_table initializes the Compumotor controller, sends table to home
    position, and
    resets global variable table_pos[table] to 0. It also sets the table to
    absolute
    position mode and turns on the output port and sets it to a low state.

    Input   :   table - table number to use
*/
init_table(table)
    int table;
    {
    configure_labnb();
    table_pos[table] = 0L;
    strcpy (g.MAC, "1LDO 1MN MPA 1000 ");
    send_command (table);
    get_command (table);
    }

/*
    motor_on sends a command which turns the stepper motor power on. This should
    be
    called before any commands are sent that require the motor for movement.

    Input   :   table - table number to use
*/
motor_on (table)
    {
```

motor.c

```
    strcpy (g.MAC, "ST0 ");
    send_command (table);
    get_command (table);
    sleep(1);
}
/*
    motor_off sends a command which turns off the stepper motor. This is used to
cut
down on RF interference produced by having the motor powered while acquiring
images.

    Input   :   table - table number to use
*/
motor_off (table)
{
    strcpy (g.MAC, "ST1 ");
    send_command (table);
    get_command (table);
    sleep(1);
}

/*
    Sends table to it's home position. This is accomplished by sending a GH+5.1
to the
    Compumotor controller, which in turn moves the table until the home position
switch
    is activated.

    Note that this routine will not return until the routine is assured that the
table is at the home position.

    Input   :   table - table number to use
*/
go_home(table)
int table;
{
    int found = 0, i, j;

    table_pos[table] = 0L;
    motor_on(table);
    strcpy (g.MAC, "1LD0 1MN MPA 1000 1000 A11 GH+5.1 1011 ");
    send_command (table);
    get_command (table);
    while (table_active (table))
    {
        sleep(1);
        strcpy (g.MAC, "1011 ");
        send_command (table);
        get_command (table);
    }
    strcpy (g.MAC, "1000 ");
    send_command (table);
    motor_off(table);
}

/*
```

motor.c

move_table_abs moves the given table to an absolute position. It also sends commands which turn off the output port until the table is done moving, at which

time the output port is then brought to a high level. This is used by done_move to signal when a move is complete.

Input : table - table number to use
Input : pos - absolute position to move table to

*/

```
move_table_zero (table)
    int table;
```

```
{
    long step;
```

```
    motor_on (table);
    sprintf(g.MAC, "1LDO 1MN MPA 1000 A10 V10 D%d G 1011 ", 0);
    send_command(table);
    get_command(table);
}
```

```
move_table_abs (table, pos)
    int table;
    long pos;
```

```
{
    long step;
```

```
    motor_on (table);
    table_pos[table] = pos;
    sprintf(g.MAC, "1LDO 1MN MPA 1000 A10 V1 D%d G 1011 ", pos*-1);
    send_command(table);
    get_command(table);
}
```

/*

done_move checks to see if a table is active by checking the compumotor output port.

If the output port is in a high state, it means that the motor has stopped. If it is low, the motor is still moving to the desired position.

Input : table - table number to use
Function

Output : returns a 1 if the table is done moving, 0 if it is still moving to the desired location.

*/

```
done_move (table)
    int table;
```

```
{
    int i;
    if ( !(i = table_active (table)) )
```

motor.c

```
{
    sprintf(g.MAC, "1000 ");
    send_command(table);
    motor_off(table);
}
return (li);
}
```

motor_dialog.c

```
/*
    motor_dialog

    contains routines relating to the motor control dialog box.
*/

#include "herrs.h"
#include "FaceLCH.h"

extern int current_table;
long extern table_pos[2];

table_dlog()
{
    long rel_step=0L, abs_step;

    g.dialog[0] = 0;
    g.dialog[1] = 0;
    g.dialog[2] = 0;
    g.dialog[3] = 0;
    g.dialog[4] = 0;
    g.dialog[5] = 0;
    g.dialog[6] = 1;
    g.dialog[7] = 0;
    g.dialog[8] = -2;
    g.dialog[9] = -2;

    sprintf (g.name, "%-8ld%-8ld", table_pos[current_table], rel_step);
    FaceIt(0,SetDlg,1003,RetCtl,0,0);
    FaceIt(0,ShoDg2,RetCtl,0,0,0);
}

update_table_dlog()
{
    long rel_step=0L, abs_step;
    int dialog_item;

    dialog_item = g.dialog[99];
    FaceIt (0, GetDlg, 1003, 0, 0, 0);
    current_table = g.dialog[7];

    sscanf (g.name, "%8ld%8ld", &abs_step, &rel_step);
    switch (dialog_item)
    {
        case 1:
            FaceIt (0, HidDg2, 0, 0, 0, 0);
            return;
        case 2:
            move_table_abs (current_table, (abs_step + rel_step));
            while (!done_move (current_table));
            break;
        case 3:
    }
```

motor_dialog.c

```
        move_table_abs (current_table, abs_step);
        while (!done_move (current_table));
        break;
    case 6:
        go_home (current_table);
        break;
    }
    sprintf (g.name, "%-8ld%-8ld", table_pos[current_table], rel_step );
    FaceIt(0,SetDlg,1003,RetCtl,0,0);
}
```

sitter.c

```
/*
    sitter.c
    Contains :
        scale
*/

#include "herrs.h"

void    FFT(cmplx *data,int nn, int isign);
cmplx   *cvectr( int size);

/*
    scale ()

    scale is the Daivd Sitter algorithm that converts a N.A.T. database into
    a telecentric database. This is the core of this algorithm (scaling routine)
*/

void    scale (long double **I, int msize, int nsize,int Mpad, int Npad, long
double fctr)
{
    int i,j,m,n,min,max,Nmax;
    long double j_scaled,i_scaled,mag,alpha;
    cmplx      *s,*t;

    Nmax = maxi(Mpad,Npad);

    if ((s=cvectr(Nmax)) == NULL)
        printf("\nerror!\n");
    if ((t=cvectr(Nmax)) == NULL)
        printf("\nError\n");
    mag=1.0+fctr;

    for (m=0;m<msize;m++)
    {
        for (j=0;j<nsize/2;j++)
        {
            s[j].re=I[m][j+nsize/2];
            s[j].im=0.0;
        }

        for (j=Npad-nsize/2;j<Npad;j++)
        {
            s[j].re=I[m][j-Npad+nsize/2];
            s[j].im=0.0;
        }
    }
}
```

sitter.c

```
for (j=ns/2;j<Npad-ns/2;j++)
    s[j].re = s[j].im = 0.0;

FFT (s, Npad, 1);

for (j=0;j<=(Npad/2);j++)
{
    j_scaled=mag*((long double)j);

    min=(int)(j_scaled);

    max=min+1;

    if (min <= Npad/2)
        {
            alpha = j_scaled - (long double)(min);
            if (min != Npad/2)
                {
                    t[j].re = s[min].re*(1.0-alpha) + s[max].re*alpha;
                    t[j].im = s[min].im*(1.0-alpha) + s[max].im*alpha;
                }
            else
                {
                    t[j].re = s[min].re*(1.0-alpha);
                    t[j].im = s[min].im*(1.0-alpha);
                }
        }
    else
        t[j].re = t[j].im = 0.0;
}

for (j=Npad-1; j>Npad/2; j--)
{
    t[j].re = t[Npad-j].re;
    t[j].im = -t[Npad-j].im;
}

FFT (t,Npad, -1);

for (j=0; j<ns/2; j++)
    I[m][j] = t[j+Npad-ns/2].re;
for (j=ns/2; j<ns; j++)
    I[m][j] = t[j-ns/2].re;
}

for (n=0; n<ns; n++)
{
    for (i=0; i<ms/2; i++)
        {
            s[i].re = I[i+ms/2][n];
            s[i].im = 0.0;
        }
    for (i=Mpad-ms/2; i<Mpad;i++)
        {
            s[i].re = I[i-Mpad+ms/2][n];
            s[i].im = 0.0;
        }
}
```


sitter.c

```
for (i=msize/2; i<Mpad-msize/2; i++)
    s[i].re = s[i].im = 0.0;

FFT (s, Mpad, 1);

for (i=0; i<= Mpad/2; i++)
{
    i_scaled = mag * ((long double)i);
    min = (int)i_scaled;
    max = min + 1;
    if (min <= Mpad/2)
    {
        alpha = i_scaled - (long double)min;
        if (min != Mpad/2)
        {
            t[i].re = s[min].re*(1.0 - alpha) + s[max].re*alpha;
            t[i].im = s[min].im*(1.0 - alpha) + s[max].im*alpha;
        }
        else
        {
            t[i].re = s[min].re*(1.0 - alpha);
            t[i].im = s[min].im*(1.0 - alpha);
        }
    }
    else
        t[i].re = t[i].im = 0.0;
}
for (i=Mpad-1; i>Mpad/2; i--)
{
    t[i].re = t[Mpad-i].re;
    t[i].im = -t[Mpad-i].im;
}

FFT (t, Mpad, -1);
for (i=0; i<msize/2; i++)
    I[i][n] = t[i+Mpad - msize/2].re;
for (i=msize/2; i<msize; i++)
    I[i][n] = t[i-msize/2].re;
}
free(s);
free(t);
}
```

sitter_driver.c

/*

sitter_drive.c

Contains code which interfaces sitter's scaling algorithm to HERSS standard databases.

*/

```
#include "herrs.h"
#define SIZE 256
#define HSIZE (256)
sitter (key)
key_file *key;
```

```
{
long double **image;
int i, x, y, slice;
FILE *in_file, *out_file;
long double xim, i_max;
unsigned char *im;
char fn[64];
long double fctr=-(0.007875/2.03125)*10.0;
```

```
if ((image = (long double **)calloc((SIZE), (long)sizeof(long double *))) ==
NULL)
```

```
{
printf("\nUnable to allocate memory for offscreen buffer\n");
exit(-1);
}
```

```
for (i=0; i < SIZE; i++)
if ((image[i] = (long double *)calloc(SIZE, (long)sizeof(long double)))
== NULL)
```

```
{
printf("\nUnable to allocate memory for complex buffer %d\n",i);
exit(-1);
}
```

```
if ((im = (unsigned char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(unsigned
char))) == NULL)
```

```
{
printf("\nUnable to allocate memory for offscreen buffer\n");
exit(-1);
}
```

```
printf("\n Memory Allocated \n");
```

```
for (slice = 0; slice < 20; slice++)
```

```
{
sprintf(fn, "%s.%d", key->filename, slice*10);
load_8bit_image (im, fn);
for (x=0; x<HSIZE; x++)
for (y=0; y<HSIZE; y++)
image[x][y] = (long double) im[ref(x,y)];
```

sitter_driver.c

```
fctr = (0.007875/2.03125) * ((long double) (slice - 10));
printf("\nScaling...\n");
scale (image, 256, 256, 512, 512, fctr);
printf("\nDone Scaling...\n");
i_max = 0.0;
for (x=0; x<SIZE; x++)
    for (y=0; y<SIZE; y++)
        if (image[x][y] > i_max)
            i_max = image[x][y];
for (x=0; x<SIZE; x++)
    for (y=0; y<SIZE; y++)
        if (image[x][y] < 0.0)
            im[ref(x,y)] = (unsigned char)(0);
        else
            im[ref(x,y)] = (unsigned char) image[x][y];
sprintf(fn, "%s.%d", key->filename, slice+key->num_slices);
save_8bit_image (im, fn);
printf("\nimax = %f\n", i_max);
}

free (im);
free (image);
}
```

sitter_fft.c

```
/*
    sitter_fft.c
    Contains standard FFT program acquired from Georgia Tech
*/

#include "herrs.h"

#define SWAP(a,b)    tempr=a;a=b;b=tempr

void FFT (cmplx *data, int nn, int isign)
{
    int    n, mmax, m, j, istep;
    int    i, k, l;
    long double wtemp, wr, wpr, wpi, wi, theta;
    long double tempr, tempi;

    n = nn<<1;
    j = 1;
    for (i=1; i<n; i+=2)
        {
            if (j>i)
                {
                    k = j/2;
                    l = i/2;
                    SWAP(data[k].re,data[l].re);
                    SWAP(data[k].im,data[l].im);
                }
            m=n>>1;
            while (m>=2 && j>m)
                {
                    j -= m;
                    m >>= 1;
                }
            j += m;
        }
    mmax = 2;
    while (n > mmax)
        {
            istep = 2*mmax;
            theta = 6.28318530717959/(isign*mmax);
            wtemp = sin (0.5*theta);
            wpr = -2.0*wtemp*wtemp;
            wpi = sin (theta);
            wr = 1.0;
            wi = 0.0;
            for (m=1; m<mmax; m+=2)
                {
                    for (i=m; i<=n; i+=istep)
                        {
                            j = i + mmax;
                            k = j/2;
```

sitter_fft.c

```
        l = i/2;
        tempr = wr*data[k].re - wi*data[k].im;
        tempi = wr*data[k].im + wi*data[k].re;

        data[k].re = data[l].re - tempr;
        data[k].im = data[l].im - tempi;
        data[l].re += tempr;
        data[l].im += tempi;
    }
    wr = (wtemp = wr)*wpr - wi*wpi + wr;
    wi = wi*wpr + wtemp*wpi + wi;
}
mmax = istep;
}
if (isign == -1)
{
    for (k=0; k<=nn-1; k++)
    {
        data[k].re /= nn;
        data[k].im /= nn;
    }
}
}
```

z-contrast.c

```
/*
   zc.c

   Contains Z-Contrast Restoration Algorithm - (Both 3x3 & 5x5)
*/

#include "herrs.h"
#include "FaceLCH.h"

zc(key, kern)
   key_file *key;
   int kern;

   {
   unsigned char *i;
   int *f;
   int slice, x, y, m, n;
   unsigned char fn[64];
   float contrast, *cont_max;
   unsigned char i_max, i_min, *h;
   long *dest_addr;
   time_t timer;

   if ((i = (unsigned char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(unsigned
char))) == NULL)
   {
   printf("\nUnable to allocate memory for offscreen buffer\n");
   exit(-1);
   }

   if ((cont_max = (float *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(float))) ==
NULL)
   {
   printf("\nUnable to allocate memory for offscreen buffer\n");
   exit(-1);
   }

   if ((h = (unsigned char *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(unsigned
char))) == NULL)
   {
   printf("\nUnable to allocate memory for offscreen buffer\n");
   exit(-1);
   }

   if ((f = (int *)calloc((X_LSIZE*Y_LSIZE), (long)sizeof(int))) == NULL)
   {
   printf("\nUnable to allocate memory for offscreen buffer\n");
   exit(-1);
   }

   switch (kern)
```

z-contrast.c

```
{
  case 3 :

    for (slice = 0; slice < key->num_slices; slice++)
    {
      sprintf(fn, "%s.%d", key->filename, slice);
      load_8bit_image (i, fn);
      printf("\nSlice %d\n", slice);
      for (y = 1; y < (Y_SIZE-1); y++)
      {
        for (x = 1; x < (X_SIZE-1); x++)
        {
          i_max = i[ref(x,y)];
          i_min = i_max;

          for (m = (x-1); m <= (x+1); m++)
            for (n = (y-1); n <= (y+1); n++)
            {
              if ( i[ref(m,n)] > i_max )
                i_max = i[ref(m,n)];
              if ( i[ref(m,n)] < i_min )
                i_min = i[ref(m,n)];
            }
          if ((i_max != 0) || (i_min != 0))
            contrast =
              (((float)i_max - (float)i_min) /
               ((float)i_max + (float)i_min));
          else
            contrast = 0.0;
          if (contrast > cont_max[ref(x,y)])
          {
            cont_max[ref(x,y)] = contrast;
            f[ref(x,y)] = slice;
            h[ref(x,y)] = i[ref(x,y)];
          }
        }
      }
      sprintf( fn, "%s.bright.%d", key->filename, ZCONTRAST_3X3);
      save_8bit_image (h, fn);
      sprintf( fn, "%s.shape.%d", key->filename, ZCONTRAST_3X3);
      break;
    }
  case 5 :
    for (slice = 0; slice < key->num_slices; slice++)
    {
      sprintf(fn, "%s.%d", key->filename, slice);
      load_8bit_image (i, fn);
      printf("\nSlice %d\n", slice);
      for (y = 1; y < (Y_SIZE-1); y++)
      {
        for (x = 1; x < (X_SIZE-1); x++)
        {
          i_max = i[ref(x,y)];
          i_min = i_max;

```

ROUTING NAME	MODULE NAME	PAGE
alg_alert	alg_dialog.c	B-1
auto_step	key_dialog.c	B-64
average_2	ias.c	B-60
average_3	ias.c	B-60
cancel	file.c	B-32
capture	capture.c	B-17
check_alg	file.c	B-33
clear_pix	hgs.c	B-54
clear_radius	hgs.c	B-51
close_key_file	file.c	B-23
configure_labnb	lab_nb.c	B-66
convert_freidbase	convert.c	B-19
convert_scan	laser_data.c	B-68
copy_8bit_map	hgs.c	B-56
copy_key	analysis.c	B-14
display_nv_shape	hgs.c	B-55
display_shape	hgs.c	B-54
do_blank_key_dialog	analysis.c	B-13
do_key_dialog	analysis.c	B-12
do_live	live.c	B-73
domenus	herss.c	B-41
done_move	motor.c	B-79
exist	file.c	B-33
FFT	sitter_fft.c	B-88
get_command	motor.c	B-77
get_comp_image	analysis.c	B-11
get_proc_slice	hgs.c	B-55
go_home	motor.c	B-78
hausler	hausler.c	B-38
hgs	hgs.c	B-47
hgs_dialog	hgs_dialog.c	B-57
hide_all_key	key_dialog.c	B-64
histogram	analysis.c	B-2
ias	ias.c	B-59
ias_alert	ias_dialog.c	B-62
init_commit	motor.c	B-76
init_nuvista	capture.c	B-15
init_table	motor.c	B-77
initialize_nvbuf	hgs.c	B-49
key_dialog	key_dialog.c	B-63
key_notes	file.c	B-27
key_step	file.c	B-27
load_8bit_image	file.c	B-30
load_16bit_image	file.c	B-32
load_32bit_image	file.c	B-31