Technical Report SL-95-21
September 1995

US Army Corps of Engineers
Waterways Experiment Station

Development of a Multispectral Signatures Database for the Camouflage, Concealment, and Deception Design and Evaluation Environment (C2D2E2)

Report 1
Technical Description

by Sandy D. Bratcher, Douglas P. Rousell, Nichols Research Corporation

Gerardo I. Velázquez, WES

Prepared for Headquarters, U.S. Army Corps of Engineers

PRINTED ON RECYCLED PAPER

Technical Report SL-95-21
September 1995

Development of a Multispectral Signatures Database for the Camouflage, Concealment, and Deception Design and Evaluation Environment (C2D2E2)

Report 1
Technical Description

by Sandy D. Bratcher, Douglas P. Rousell

Nichols Research Corporation
P.O. 820186
Vicksburg, MS 39182-0186

Gerardo I. Velázquez

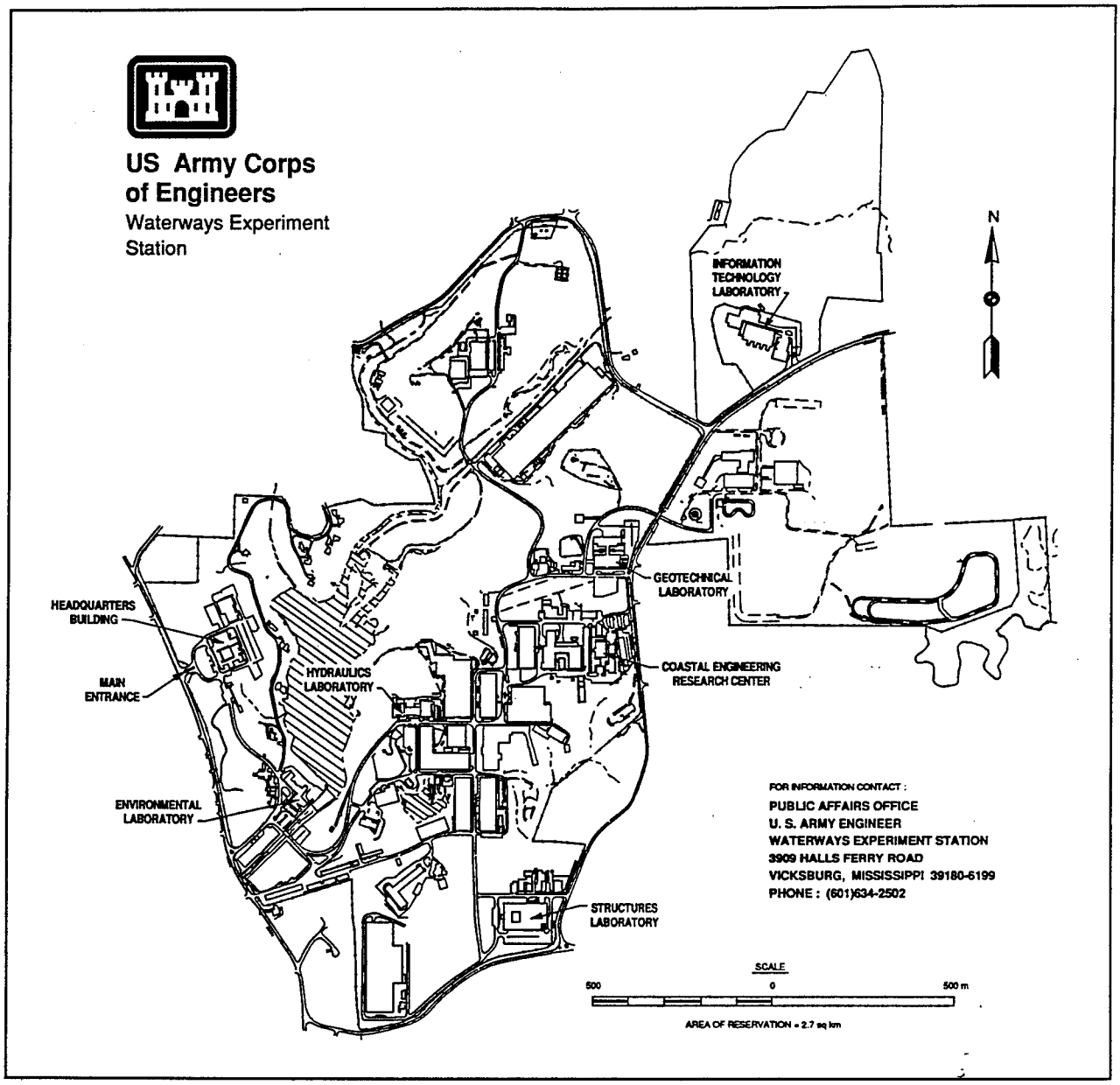
U.S. Army Corps of Engineers
Waterways Experiment Station
3909 Halls Ferry Road
Vickskburg, MS 39180-6199

Final report

Approved for public release; distribution is unlimited

Prepared for U.S. Army Corps of Engineers
Washington, DC 20314-1000

Under Contract No. DACA39-91-C-0042



Waterways Experiment Station Cataloging-in-Publication Data

Bratcher, Sandy D.

Development of a multispectral signatures database for the Camouflage, Concealment, and Deception Design and Evaluation Environment (C2D2E2). Report 1, Technical description / by Sandy D. Bratcher, Douglas P. Rousell, Gerardo I. Velazquez ; prepared for U.S. Army Corps of Engineers.

55 p. : ill. ; 28 cm. -- (Technical report ; SL-95-21 rept.1)

Includes bibliographic references.

Report 1 of a series.

1. Camouflage (Military Science) -- Computer programs. 2. Deception (Military Science) -- Computer programs. 3. Image processing -- Digital techniques. I. Rousell, Douglas P. II. Velazquez, Gerardo I. III. United States. Army. Corps of Engineers. IV. U.S. Army Engineer Waterways Experiment Station. V. Structures Laboratory (U.S. Army Engineer Waterways Experiment Station) VI. Title. VII. Series: Technical report (U.S. Army Engineer Waterways Experiment Station) ; SL-95-21 rept.1.

TA7 W34 no.SL-95-21 rept.1

Contents

Appendix A. Figures

Preface

This study was conducted by personnel of Nichols Research Corporation, Vicksburg, MS, under Contract No. DACA39-91-C-0042, and the U.S. Army Engineer Waterways Experiment Station (WES). The study comprises part of Department of the Army Project No. P4A162784AT40, Task CO, Work Unit 026, Fixed-Facility Camouflage, Concealment, Deception (CCD), Design & Evaluation Environment Technologies, which is sponsored by Headquarters, U.S. Army Corps of Engineers.

The report was prepared by Messrs. Sandy D. Bratcher and Douglas Rousell, Nichols Research Corporation, and Mr. Gerardo I. Velázquez, CCD Research Group (CCDRG), under the direct supervision of Mr. Kenneth G. Hall, Chief, CCDRG, Structures Laboratory (SL), WES. The authors were also assisted by Messrs. Gene Barnett, Runn L. Gunn, and Jerry L. Stringer, Nichols Research Corporation, and Mr. Bartley P. Durst and Ms. Eva J. Farmer, CCDRG.

The study was conducted at WES during the period May 1991 to June 1992 under the general supervision of Dr. John Harrison, Director, Environmental Laboratory (EL); Dr. Victor E. LaGarde III, Chief, Environmental Systems Division, EL; and under the direct supervision of Mr. Hall. During the period June 1992 to July 1994, the study was conducted under the general supervision of Mr. Bryant Mather, Director, SL; Dr. Jimmy P. Balsara, Chief, Geomechanics and Explosion Effects Division; and Dr. Reed L. Mosher, Chief, Structural Mechanics Division; and under the direct supervision of Mr. Hall. Technical Monitors were Mr. Al Knoch (CEMP-ET) and Mr. Bruce Walton (CEMRO).

During the preparation of this report, the Director of WES was Dr. Robert W. Whalin. Commander was COL Bruce K. Howard, EN.

This report should be cited as follows:

Bratcher, S. D., Rousell, D. P., and Velázquez, G. I. (1995). "Development of a multispectral signatures database for the camouflage, concealment, and deception design and evaluation environment (C2D2E2): Report 1, Technical description," Technical Report SL-95- , U.S. Army Engineer Waterways Experiment Station, Vicksburg, MS.

1 Introduction

The U.S. Army Engineer Waterways Experiment Station (WES) Camouflage, Concealment, and Deception (CCD) Research Group provides CCD for fixed tactical assets. Worldwide Department of Defense assets include installations such as logistics bases, supply depots, and other permanent and semimobile assets. To accomplish this mission, the CCD Research Group has been actively gathering multispectral imagery of fixed and semifixed facilities for several years. With this imagery there are ground truth and physical data collected relating to the visual, near infrared, thermal, and radar bands. Similar data have been collected within other agencies and accumulated in existing Government databases. The Camouflage, Concealment, and Deception Design and Evaluation Environment (C2D2E2) is being developed as an integrated unique group of software applications designed specifically to address this challenge. As part of this effort, WES has begun the development of a multispectral signatures database (MSD) to handle the all the data available.

CCD analysts and engineers defined data requirements (image attributes) and high-level operational requirements for this database to handle the great amount of information available at WES and other Government agencies. High-level operational requirements included: organizing and cataloging a large number of images, associating appropriate attribute data with each image, associating images of the same scene in different spectral bands, and retrieval of related images and their respective attribute data. Raima Data Manager (RDM) was used for the underlying Database Management System. The network technology provided by RDM was used to support complex relationships such as many object signatures in many scenes. These relationships are very inefficient and sometimes not configurable with typical relational technology. For image storage, a 24-bit color Tagged Image File Format was used due to the flexibility and support for this format.

The MSD database is hosted on an 486 IBM PC and has undergone development in the ANSI C language. Raima Data Manager System III (RDM) was used as the underlying Database Management System. The network technology provided by RDM was used due to its support of complex relationships that are not efficiently represented with typical relational technology. For image storage, a 24-bit color Tagged Image File Format (TIFF) was used due to the flexibility and support for this format. A single-user menu interface is provided for the management of the database. Some of the operational requirements that were defined for this user interface include: organizing and cataloging a large number of images; associating appropriate attribute, ground truth, and physical data with each image; associating images of the same scene in different spectral bands; and the ability to retrieve/display images and their respective data. See **Appendix A -- Figure 1** for an overview representation of the MSD database.

This report summarizes the technical effort performed, the results achieved, and the current status of the Multispectral Signatures Database.

Hardware and Software Requirements

The Multispectral Signatures Database software is hosted on a 486-66DX IBM-compatible PC with MS-DOS version 6.2 and 20 MB RAM. A VESA SVGA or a Paradise SVGA video card and a minimum of 4 MB of expanded memory is required only for image display.

To run the MSD software, 600 KB hard disk space is required plus a minimum of 75 KB for each database created. The MSD source code and libraries needed for recompiling the MSD require an additional 2.2 MB hard disk space.

The following software is also required ONLY to maintain or modify the MSD software:

- Microsoft C version 6
  (compiler)
- Plink86 version 2.3 by Sage Software
  (used for overlay linker)

-Raima Data Manager System III version 3.21 --
Single user version for Microsoft C 6.0
(RDM library to link with MSD software)

- db_QUERY version 2.21
  (db_QUERY library to link with MSD software)
- db_REVISE version 1.04
  (db_REVISE utility to revise the database schema)

-Victor Image Processing Library version 2.2 by Catenary Systems
(image display library to link with MSD software)

-TLIB Version Control software version 4.12 by Burton Systems Software
(for source code configuration)

Only the Database Administrator should make changes to the database schema. Changes to the database schema will require corresponding changes to the MSD software.

2 Technical Development

Database Concepts

Raima Data Manager (RDM) is the underlying Database Management System (DBMS) used in the Multispectral Signatures Database software. There are many advantages to using RDM. Raima's database products are royalty-free when incorporated into an application such as the MSD Database. RDM provides both relational and network technologies which yields a flexible database design that can support more complex data relationships. It is very portable in that it is hardware-independent and supports multiple operating systems. It supports many standards including the structured query language (SQL) standard and standard C data types; in addition, it was developed in ANSI standard C. Raima's db_REVISE utility provides sophisticated database restructuring tools. Easy modifications to the structure of the MSD Database is an important advantage since its structure will need to be modified as the data collection process is changed or improved. For example, new field(s) must be added if types of ground data are collected other than those that are currently supported by the MSD Database structure.

The basic unit of information in a database is a "field" (or "data field"). A field is an item of data with attributes such as name, type (e.g., character or integer), and length. A "record" is a named collection of related fields, which are stored and accessed as a unit. Each "occurrence" of a record in the database contains a value for each of its fields. The definition of a record (as comprised by its fields) is called the "record type." All occurrences of a particular record type are stored in an operating system "file." Files are the primary physical storage units into which a database is organized. A "database," therefore is a collection of related files. A "key" is a field through which rapid and/or sorted access to a record is possible. An "index" (or "key file") is a file containing only keys. Raima Data Manager uses the "B-tree" method, one of the most efficient techniques for implementing an index, for maintenance of its key files. Data "relationships" often exist between record types. The "schema" is the conceptual definition of the content and organization of a database. A schema would include the definitions of all record types, with their fields and keys. The form of the schema utilized by the DBMS is called the "dictionary." In Raima Data Manager a Database Definition Language (DDL) specifies the schema.

A "database model" is a conceptual representation of interrecord relationships. Raima Data Manager supports both the "relational" and the "network" database models. The "relational database model" establishes and maintains interrecord relationships through common data fields. The "network database model" establishes interrecord relationships directly through physical links using references to memory locations between the related records (owner/member SET pointers). The primary benefit of a relational model database system is that the simplicity of the underlying data model makes it easy to use. The primary benefits of a network model database system are better performance, reduced storage requirements, and greater assurance of data integrity. Combining these technologies gives you maximum database design flexibility.

A Raima Data Manager database consists of:
- a dictionary which stores information describing the content and organization of the database

- data files which contain occurrences of one or more record types
- key files which contain an index for one or more key fields

The Database Definition Language (DDL) Processor, ddlp, is a Raima Data Manager utility that compiles a DDL specification or schema. Compilation of the DDL produces a database dictionary and a C header file. The C header file contains constants and declarations associated with the record types and fields within a specific database for use by the C programs which access the database. If database structure changes are ever made that require the modification of the DDL, the database header file must be recompiled.

After the DDL has been compiled, the database must be initialized to create the data files (*.dat) and key files (*.key). Another Raima Data Manager utility, initdb, is provided to initialize the database. If an existing database is initialized, the contents of the data and key files will be destroyed.

MSD Database Schema Definition

The Database Definition specification or schema for the MSD Database is contained in the file, "IMAGEDBS.DDL"; this file is located in the MSD root directory. A listing of the DDL is included in **Appendix B**. A schema diagram depicting the network relationships for the MSD database is included in **Appendix A -- Figure 2**.

Following is a description of each of the record types with their associated fields and all sets defined in the MSD database.

Description of basic record types
The following record types contain information about the data collection effort and the scenes associated with it.

o DATA_COLLECTION_EFFORT

* DCE_NAME
Description: Name of the Data Collection Effort, for example, "21st TAACOM", "APDI", etc.
Type: UNIQUE KEY char[81].

* BEGIN_DATE
Description: Date of first day of the data collection effort.
Type: unsigned long (YYYYMMDD).

* END_DATE
Description: Date of last day of the data collection effort.
Type: unsigned long (YYYYMMDD).

* DATE_LAST_MODIFIED
Description: Date this record was last modified (updated automatically when a record is changed).
Type: unsigned long (YYYYMMDD).

- SITE
  - * SITE_NAME_CODE
    - Description: Name of site.
    - Note: See site_name_codelist table below for an example list of values.
    - Type: KEY unsigned short (codelist).
  - * AREA_OF_INTEREST
    - Description: Description of the area of interest at a site.
    - Note: There may be multiple records with the same SITE_NAME but different AREA_OF_INTEREST's. A site would usually have multiple areas of interest when data is being collected about more than one target at the same site.
    - Type: char[81].
  - * BEGIN_DATE
    - Description: Date of first day of data collection at this site/area of interest.
    - Type: unsigned long (YYYYMMDD).
  - * END_DATE
    - Description: Date of last day of data collection at this site/area of interest.
    - Type: unsigned long (YYYYMMDD).
  - * DATE_LAST_MODIFIED
    - Description: Date this record was last modified (updated automatically when a record is changed).
    - Type: unsigned long (YYYYMMDD).

- SCENE
  - * DESCRIPTION
    - Description: Description of the scene.
    - Type: UNIQUE KEY char[81].
  - * DATE_CAPTURED
    - Description: Date that this scene was captured.
    - Type: KEY structructure
      {
      unsigned short julian_day;
      unsigned short base_year;
      }
  - * SCENE_ID
    - Description: Scene_ID to uniquely (across all MSD databases) identify a specific Scene record (eight digital

decimal number from 00000001 to 16777215 -- set automatically when a record is added).

Note: This SCENE_ID is used to construct the IMAGES filename. See note on IMAGES below. Because there can be multiple databases, the next available SCENE_ID is stored in a binary file, "imagedbs.sys". The file, IMAGEDBS.SYS, is created automatically the first time a SCENE record is added. This file must not be deleted.

Type: UNIQUE KEY char[9].

* SEASON_CAPTURED

Description: Season that this scene was captured (spring, summer, fall, or winter).

Type: KEY char[7].

* LATITUDE

Description: Latitude of the image collection platform.

Type: float.

* LONGITUDE

Description: Longitude of the image collection platform.

Type: float.

* ALTITUDE

Description: Altitude of the image collection platform.

Type: float.

* BORESIGHT_AZIMUTH

Description: Azimuth degrees (positive number from 0 to 359) of the boresight of the image collection platform (e.g. camera) from "magnetic North" not "true North".

Note: If the exact azimuth is not available, one of the eight cardinal headings (i.e., N, NE, E, SE, S, SW, W, NW) could be specified instead of the azimuth degrees. Anytime an approximation is given, the "~" character should precede the value to indicate that the value is an approximation (e.g. ~NE or ~60).

Type: char[5].

* DATE_LAST_MODIFIED

Description: Date this record was last modified (updated automatically when a record is changed).

Type: unsigned long (YYYYMMDD).

o IMAGES

*SPECTRUM_CODE

Description: Spectrum in which image was captured.
Note: See spectrum_codelist table below for the list of valid values.
Type: KEY unsigned short (codelist).

* TIME_IMAGE_CAPTURED

Description: Time that this image was captured (from SMPTE).
Note: Time is the 24-hour local time that the image was captured. A utility is available that will give the position and angle of the sun given julian day, latitude, longitude, and local time.
Type: unsigned long (HHMMSSFF - where FF is the same count).

* TYPE_RECORDING_MEDIA

Description: Type of recording media used to capture the image (e.g., 35mm, Medium-Format, VHS, HI-8).
Type: char[21].

* ANALOG_DATA_NAME

Description: Notation that designates what types of analog data are available and the location of that data. This is automatically assigned by the software when an IMAGES record is added.
Type: char[31].

* DIGITAL_DATA_VOLUME

Description: Name of the volume (of the optical disk platter) on which the image file is located. The first 10 characters represent the volume name which will correspond to a data_collection_effort and the last character represents the side number (1, 2, 3, etc.).
Type: char[12].

* DIGITAL_DATA_FILENAME

Description: Name of the image file (not compressed) including location (i.e., path\filename) stored with the database.
Note: When an Images record is added to the database, the source image file is copied to the destination image file that will be permanently stored with the database. The destination image file will not have the same file name as its source file. The destination image filename is constructed by concatenating a two-character spectrum_code with the SCENE_ID associated with the corresponding SCENE record (the unique [across all databases] 8-digital decimal number converted to a six-digit

hexadecimal number). The available spectrum codes are "VI" (VISUAL), "I3" (INFRARED 3-5), "I8" (INFRARED 8-12), "NI" (NEAR IR), and "LL" (LOW-LIGHT). The same image captured in different spectrums would have the same SCENE_ID but different spectrum codes.

Type: char[41].

* CMPRS_DATA_VOLUME

Description: This field is not currently used. Name of the volume (of the optical disk platter) on which the compressed image file is located. The first 10 characters represent the volume name which will correspond to a data_collection_effort and the last character represents the side number (1, 2, 3, 4, etc.).

Type: char[12].

* CMPRS_DATA_FILENAME

Description: This field is not currently used. Name of the image (in compressed format) including the location (i.e., path\filename).

Note: This filename would correspond to the naming schema used for the digital_data_filename.

Type: char[41].

* DATE_IMAGE_ADDED

Description: Date this image was added to the database (will be set automatically when new IMAGES record is added).

Type: unsigned long (YYYYMMDD).

* DATE_LAST_MODIFIED

Description: Date this record was last modified (updated automatically when a record is changed).

Type: unsigned long (YYYYMMDD).

o SCENE_OBJECT_INTERSECT

NOTE: This table is used to handle the many-to-many relationship between objects and scenes: 1) there can be more than one object in a scene, and 2) an object can be in more than one scene.

* SLANTRANGE_TO_IMG_COLL_PLAT

Description: Slant-range distance from this object to the image collection platform. This distance could differ for the same object in different SCENE's.

Type: float.

* LOOK_ANGLE
  Description: The angle formed by the line from the image collection platform to the object and the line from the image collection platform parallel to the ground.
  Type: float.

oOBJECT
* OBJ_CLASS_CODE
  Description: Object Class to which this object belongs.
  Note: See obj_class_codelist table below for the list of valid values.
  Type: KEY unsigned short (codelist).

* OBJ_GROUP_CODE
  Description: Object Group to which this object belongs.
  Note: See obj_group_codelist table below for a list of example values.
  Type: KEY unsigned short (codelist).

* OBJECT_NAME
  Description: Unique name/description of this object. An object is defined as a target or background with unique characteristics including the location of the object and CCD application.
  Note: If a target/background is included in some SCENE's with CCD and some SCENE's without CCD, this target/background would be added as two separate OBJECT's; for example, "building #1001 with Brunswick Ultralite net" and "building #1001 with Tracor Lightweight net" and "building #1001 without CCD". If a mobile target/background is included in some SCENEs at one location and some SCENE's at a different location, this target/background would be added as two separate OBJECT's; for example "apache 1 near treeline" and "apache 1 at FARP".
  Type: UNIQUE KEY char[81].

* OBJECT_SIZE
  Description: Size/Dimensions of the object (HxWxL in meters).
  Type: char[31].

* OBJ_LATITUDE
  Description: Latitude of the object.
  Type: float.

* OBJ_LONGITUDE
  Description: Longitude of the object.
  Type: float.

* OBJ_ALTITUDE
  - Description: Altitude of the object.
  - Type: float.

* DATE_LAST_MODIFIED
  - Description: Date this record was last modified (updated automatically when a record is changed).
  - Type: unsigned long (YYYYMMDD).

o CCD
* CCD_TYPE_CODE
  - Description: Type of CCD applied to object.
  - Note: See ccd_type_codelist table below for a list of example values.
  - Type: KEY unsigned short (codelist).

* CCD_IDENTIFIER_CODE
  - Description: Specific name of CCD applied to object.
  - Note: See ccd_identifier_codelist table below for a list of example values.
  - Type: KEY unsigned short (codelist).

* VISUAL_PROPERTIES
  - Description: String describing the visual properties associated with the CCD applied to an object.
  - Type: char[81].

* THERMAL_PROPERTIES
  - Description: String describing the thermal properties associated with the CCD applied to an object.
  - Type: char[81].

* RADAR_PROPERTIES
  - Description: String describing the radar properties associated with the CCD applied to an object.
  - Type: char[81].

* OTHER_PROPERTIES
  - Description: String describing other miscellaneous properties associated with the CCD applied to an object.
  - Type: char[81].

* DATE_LAST_MODIFIED
  - Description: Date this record was last modified (updated automatically when a record is changed).
  - Type: unsigned long (YYYYMMDD).

Description of data record types

The following record types contain information about the physical and ground truth data associated with objects:

- MET_DATA
  - * MET_DESCRIPTION
    - Description: Description of met data file.
    - Type: UNIQUE KEY char[81].
  - * MET_FILENAME
    - Description: Filename assigned to meteorological data file.
    - Note: When a MET_DATA file is added to the database, the source file is copied to the database directory with a new name. The new filename is constructed by concatenating "MET" with a five-character unique number (per database) and a file extension of ".MET" (e.g., MET00001.MET).
    - Type: char[41].

- CAL_PANEL_DATA
  - * CAL_PANEL_DESCRIPTION
    - Description: Description of cal panel data file.
    - Type: UNIQUE KEY char[81].
  - * CAL_PANEL_FILENAME
    - Description: Filename assigned to calibration panel data file.
    - Note: When a CAL_PANEL_DATA file is added to the database, the source file is copied to the database directory with a new name. The new filename is constructed by concatenating "CAL" with a five-character unique number (per database) and a file extension of ".CAL" (e.g., CAL00001.CAL).
    - Type: char[41].

- CHROMO_DATA
  - * CHROMO_DESCRIPTION
    - Description: Description of chromatic data file.
    - Type: UNIQUE KEY char[81].
  - * CHROMO_FILENAME
    - Description: Filename assigned to chromatic data file.
    - Note: When a CHROMO_DATA file is added to the database, the source file is copied to the database directory with a new name. The new filename is constructed by concatenating "CHR" with a five-character unique number (per database) and a file

extension of ".CHR" (e.g., CHR00001.CHR).

Type: char[41].

- GLOSS_DATA
  - GLOSS_DESCRIPTION

    Description: Description of gloss data file.

    Type: UNIQUE KEY char[81].

  - GLOSS_FILENAME

    Description: Filename assigned to gloss data file.

    Note: When a GLOSS_DATA file is added to the database, the source file is copied to the database directory with a new name. The new filename is constructed by concatenating "GLO" with a five character unique number (per database) and a file extension of ".GLO" (e.g., GLO00001.GLO).

    Type: char[41].

- SPECTRAD_DATA
  - SPECTRAD_DESCRIPTION

    Description: Description of the spectral radiometer data file.

    Type: UNIQUE KEY char[81].

  - SPECTRAD_FILENAME

    Description: Filename assigned to the spectral radiometer data file.

    Note: When a SPECTRAD_DATA file is added to the database, the source file is copied to the database directory with a new name. The new filename is constructed by concatenating "SPE" with a five-character unique number (per database) and a file extension of ".SPE" (e.g., SPE00001.SPE).

    Type: char[41].

Description of Codelist Record Types

The following record types support the use of "codelists." Codelist record types are used to decrease the amount of storage required for repetitive data and to store a domain of possible values for a field in a separate record type. A codelist number is associated with each codelist value. The number instead of the value is then stored in the data record type. The data record type is then "linked" to the codelist record type through a common field, the codelist number. Some of the codelist record types in the MSD database are loaded with values by the MSD software upon creation of a new database. Others are loaded with values when records (of the record type with which the codelist record type is linked) are added to the MSD database using the MSD software.

- SITE_NAME_CODELIST
  - CODENUM
    - Description: Codelist number
    - Type: UNIQUE KEY unsigned short
    - Linked to: SITE.SITE_NAME_CODE
  - CODEVAL
    - Description: Site name codelist value
    - Type: UNIQUE KEY char[51];
  - LOCATION_CODEVAL
    - Description: Site location codelist value
    - Type: KEY char[51]
    - Example values:

      1 = Germersheim Depot
      Germersheim, Germany
      2 = Husterhoeh POMCUS Site
      Pirmasens, Germany
      3 = Kaiserslautern Ammo Depot
      Kaiserslautern, Germany
      4 = Waterways Experiment Station
      Vicksburg, Mississippi

- OBJ_CLASS_CODELIST
  - CODENUM
    - Description: Codelist number
    - Type: UNIQUE KEY unsigned short
    - Linked to: OBJECT.OBJ_CLASS_CODE
  - CODEVAL
    - Description: Object class codelist value
    - Type: UNIQUE KEY char[51]
    - Valid values (preloaded):

      1 = Data Collection Activity
      2 = Fixed Facility With Camouflage
      3 = Fixed Facility Without Camouflage

4 = Mobile Systems With Camouflage
5 = Mobile Systems Without Camouflage

* SECOND_CODEVAL
  Description: Abbreviated object class codelist value
  Type: UNIQUE KEY char[6]
  Valid values (preloaded):
  1 = DCACT
  2 = FFWIC
  3 = FFWOC
  4 = MSWIC
  5 = MSWOC

o OBJ_GROUP_CODELIST
  * CODENUM
    Description: Codelist number
    Type: UNIQUE KEY unsigned short
    Linked to: OBJECT.OBJ_GROUP_CODE

  * CODEVAL
    Description: Object group codelist value
    Type: UNIQUE KEY char[51]
    Example values :
    1 = Apache
    2 = Pomcus Warehouses
    3 = Hemtt Tanker

o CCD_TYPE_CODELIST
  * CODENUM
    Description: Codelist number
    Type: UNIQUE KEY unsigned short
    Linked to: CCD.CCD_TYPE_CODE

  * CODEVAL
    Description: CCD type codelist value
    Type: UNIQUE KEY char[31]
    Example values:
    1 = Net
    2 = Coating
    3 = Decoy

o CCD_IDENTIFIER_CODELIST
  * CODENUM
    Description: Codelist number
    Type: UNIQUE KEY unsigned short
    Linked to: CCD.CCD_IDENTIFIER_CODE

* CODEVAL
  Description: CCD identifier codelist value
  Type: UNIQUE KEY char[51]
  Example values:
    1 = Standard DoD
    2 = Brunswick Ultralite
    3 = Tracor Lightweight

o SPECTRUM_CODELIST
  * CODENUM
    Description: Codelist number
    Type: UNIQUE KEY unsigned short
    Linked to: IMAGES.SPECTRUM_CODE

  * CODEVAL
    Description: Spectrum codelist values
    Type: UNIQUE KEY char[21]
    Valid values (preloaded):
      1 = VISUAL
      2 = INFRARED 3-5
      3 = INFRARED 8-12
      4 = NEAR IR
      5 = LOW-LIGHT

  * SECOND_CODEVAL
    Description: Abbreviated spectrum codelist values
    Type: UNIQUE KEY char[3]
    Valid values (preloaded):
      1 = VI
      2 = I3
      3 = I8
      4 = NI
      5 = LL

Description of Sets

The following are the description of the SET's established to support the network structure of the MSD Database with the type of relationship supported by each SET:

o DCE_TO_SITE_LINK

- Relates a SITE record to the DATA_COLLECTION_EFFORT (DCE) record for which it is associated.

OWNER data_collection_effort
MEMBER site

This is a one-to-many relationship:

* A DCE can have multiple SITE's associated with it.
* A SITE can only be associated with one DCE. However, a SITE can be associated with multiple DCE's, by creating a new SITE record with the same SITE name. This

capability is inherent in the MSD software.

- SITE_TO_SCENE_LINK

- Relates a SCENE record to the SITE record where the SCENE was captured

OWNER site
MEMBER scene

This is a one-to-many relationship:

* A SITE can have multiple SCENE's associated with it.
* A SCENE can only be associated with one and only one SITE.

- SCENE_TO_IMAGES_LINK

- Relates an IMAGES record to the SCENE record that describes the scene of that image.

OWNER scene
MEMBER images

This is a one-to-many relationship:

* A SCENE can have multiple IMAGES associated with it. There will be one IMAGES record for each spectrum in which this SCENE was captured.
* An IMAGES record can only be associated with one and only one SCENE.

See **Appendix A -- Figure 3** for a schema diagram depicting the "SCENE - IMAGES Relationship."

- SCENE_TO_OBJECT_LINK

- Relates (indirectly thru an "intersection" record) a SCENE record to the OBJECT record(s) that are included in this SCENE.

OWNER scene
MEMBER scene_object_intersect

- OBJECT_TO_SCENE_LINK

- Relates (indirectly thru an "intersection" record) an OBJECT record to the SCENE record(s) in which this OBJECT is included.

OWNER object
MEMBER scene_object_intersect

This is a many-to-many relationship:

* A SCENE can have multiple OBJECT's associated with it.
* An OBJECT can be associated with multiple SCENE's.

See **Appendix A -- Figure 4** for a schema diagram depicting the "SCENE - OBJECT Relationship".

- OBJECT_TO_CCD_LINK

- Relates a CCD record to the OBJECT record to which the CCD was applied.

OWNER object
MEMBER ccd

This is a one-to-one relationship:

* An OBJECT can have one and only one CCD associated with it. This one-to-one relationship is enforced by the MSD software.

- MET_TO_SITE_LINK

- Relates a SITE record to the MET_DATA record containing the met data that were collected at this SITE.

OWNER met_data
MEMBER site

This is a one-to-many relationship:

* A MET_DATA can have multiple SITE's associated with it.
* A SITE can be associated with one and only one MET_DATA.

- CAL_TO_SITE_LINK

- Relates a SITE record to the CAL_PANEL_DATA record containing the calibrated panel data that were collected at this SITE.

OWNER cal_panel_data
MEMBER site

This is a one-to-many relationship:

* A CAL_PANEL_DATA can have multiple SITE's associated with it.
* A SITE can be associated with one and only one CAL_PANEL_DATA.

- CAL_TO_SCENE_LINK

- Relates a SCENE record to the CAL_PANEL_DATA record containing the calibrated panel data included in this SCENE.

OWNER cal_panel_data
MEMBER scene

This is a one-to-many relationship:

* A CAL_PANEL_DATA can have multiple SCENE's associated with it.
* A SCENE can be associated with one and only one CAL_PANEL_DATA record. NOTE that a CAL_PANEL_DATA file will contain the data for all three calibrated panels (low, medium, and high emissivity).

- SITE_TO_CHROMO_LINK

- Relates a CHROMO_DATA record to the SITE record where the chromatic data were collected.

OWNER site
MEMBER chromo_data

This is a one-to-many relationship:

* A SITE can have multiple CHROMO_DATA records associated with it.
* A CHROMO_DATA record can be associated with one and only one SITE.

- CHROMO_TO_OBJECT_LINK

- Relates an OBJECT to the CHROMO_DATA record that contains chromatic data pertaining to that OBJECT.

OWNER chromo_data
MEMBER object

This is a one-to-many relationship:

* A CHROMO_DATA record can have multiple OBJECT's associated with it.
* An OBJECT can be associated with one and only one CHROMO_DATA record.

- SITE_TO_GLOSS_LINK

- Relates a GLOSS_DATA record to the SITE record where the gloss data were collected.

OWNER site
MEMBER gloss_data

This is a one-to-many relationship:

* A SITE can have multiple GLOSS_DATA records associated with it.
* A GLOSS_DATA record can be associated with one and only one SITE.

- GLOSS_TO_OBJECT_LINK

- Relates an OBJECT to the GLOSS_DATA record that contains gloss data pertaining to that OBJECT.

OWNER gloss_data
MEMBER object

This is a one-to-many relationship:

* A GLOSS_DATA record can have multiple OBJECT's associated with it.
* An OBJECT can be associated with one and only one GLOSS_DATA record.

- SITE_TO_SPECTRAD_LINK

- Relates a SPECTRAD_DATA record to the SITE record where the spectral radiometer data were collected.

OWNER site
MEMBER spectrad_data

This is a one-to-many relationship:

* A SITE can have multiple SPECTRAD_DATA records associated with it.
* A SPECTRAD_DATA record can be associated with one and only one SITE.

- SPECTRAD_TO_OBJECT_LINK

-Relates an OBJECT to the SPECTRAD_DATA record that contains spectral radiometer data pertaining to that OBJECT.

OWNER spectrad_data
MEMBER object

This is a one-to-many relationship:

* A SPECTRAD_DATA record can have multiple OBJECT's associated with it.
* An OBJECT can be associated with one and only one SPECTRAD_DATA record.

- SITE_TO_RADTHERM_LINK

-Relates a RAD_THERM_DATA record to the SITE record where the radiometer / thermistor data was collected.

OWNER site
MEMBER rad_therm_data

This is a one-to-many relationship:

* A SITE can have multiple RAD_THERM_DATA records associated with it.
* A RAD_THERM_DATA record can be associated with one and only one SITE.

- OBJECT_TO_RADTHERM_LINK

-Relates a RAD_THERM_DATA record to the OBJECT to which the radiometer / thermistor data pertains.

OWNER object
MEMBER rad_therm_data

This is a one-to-many relationship:

* An OBJECT can have multiple RAD_THERM_DATA records associated with it.

* A RAD_THERM_DATA record can be associated with one and only one OBJECT.

Compilation and Initialization of the MSD Database

The steps taken to compile and initialize the MSD Database are specified below and are supplied ONLY for informational purposes. The only reason for recompiling the DDL would be if the database structure was being modified using db_REVISE. Only the Database Administrator should compile the DDL and only in limited circumstances. NOTE that changes to the MSD database schema will require corresponding changes to the MSD software. An MSD database should NEVER be initialized; doing so destroys all existing data.

Compilation of the MSD DDL produces the database dictionary (imagedbs.dbd) and a C header file (imagedbs.h). The following command line was used to compile the DDL:

```
ddlp -d -amsc imagedbs.ddl
```

The "-d" option allows duplicate field names within different record types. The "-amsc" option is used for word alignment for Microsoft C. The DDLP command is contained in the RDM321\BIN directory.

The following command line was used to initialized the database:

```
initdb imagedbs
```

Using the "-y" option with initdb would initialize the entire database (that is, create all of the data and key files) without prompting the user for confirmation of each file.

Image Files

Image files to be stored in the MSD database must be in TIFF (Tagged Image File Format) 24-bit RGB uncompressed format. When an Image is added to the MSD database, the user is prompted for the image filename. This image file is copied from the specified location to the "Images directory"; that is, "<last_drive>:\IMAGES." Images for all MSD databases are stored in the Images directory. Images are stored on the last drive because the "last_drive" is always the optical disk drive on the C2D2E2 system. Images are stored on an optical disk drive platter due to the large amount of disk space required to store images.

The volume name of the optical disk platter (or last drive) where the Image is stored is also stored in the MSD database along with the Image filename. When querying for an Image, the

user will be prompted to insert the correct optical disk platter if it is not already inserted. The image is displayed using a 256-color video display mode at a resolution of 640 x 480.

Unique image filenames are generated as follows:

<2_character_spectrum><6_digit_Hex_Scene_ID>.TIF

An example image filename is "VI00001F.TIF." The two-character spectrum values are stored in the MSD database in the SECOND_CODEVAL field of the SPECTRUM_CODELIST record. This codelist record is loaded with values by the MSD software upon creation of a new database. Currently, the available values for the SPECTRUM_CODELIST record are: "VI" for Visual, "I3" for Infrared 3-5, "I8" for Infrared 8-12, "NI" for Near IR, and "LL" for Low-Light. The Scene ID portion of the filename is taken from the Scene record for which an Image is associated. The SCENE_ID field in the SCENE record contains a unique value (ranging from 1 to 16777215). This value is converted to a six-digit hexadecimal format for use in the image filename (ranging from 000001 to FFFFFF). An example of a full file specification of an image file stored in an MSD database would be "F:\IMAGES\VI0000F1.TIF."

3 Source Code

There are approximately 40,000 lines of source code in the entire MSD system. **Appendix C** includes the actual source code. The source files, the supporting directory structure, and general information on maintaining this source code are documented below.

Maintenance of the MSD Source Code

The source code for the MSD system was written in the C programming language. All source code used to create the MSD System executable, UI.EXE, was compiled with Microsoft C version 6.0 using the LARGE memory model. Overlays were necessary because the executable was so large; therefore, the PLINK overlay linker was used to link UI.EXE.

Object libraries are used to organize the MSD source code. Each library defines a distinct level of functionality within the MSD software. The object libraries and their associated source code are located in subdirectories under the <MSD_ROOT>\LIB subdirectory while the source code for the MAIN function is located in the <MSD_ROOT>\MAINSRC subdirectory.

Each directory containing MSD source code contains a MAKEFILE. The Microsoft NMAKE utility can be used to build the libraries and/or executables by setting the default path to the appropriate directory and issuing the following command:

NMAKE

There are environment variables within the MAKEFILE's that specify the location of certain files such as object library files, include files, etc. If any of these files are moved to a different location, the values of these environment variables may need to be modified before using NMAKE to reflect the new location of the files. As specified in the MAKEFILE's, by default, all source code will be compiled and linked with the DEBUG option set. To compile and link without debug, issue the following command:

NMAKE "NO_DEBUG="

As stated earlier, the system file limit must be increased by setting "files=100" in your CONFIG.SYS file because of the large number of database files. The maximum number of file handles and streams that the MSD program (UI.EXE) can handle also had to be increased. This was achieved by changing some constants in the C startup source files (provided by Microsoft C/C++ Version 6.0), compiling the startup files, and linking the new startup code with the MSD program. This procedure is documented in the Run-Time Library Reference document of the Microsoft C/C++ version 6.0 document set on pages 40 - 42. Using this procedure, the following changes were made:

To increase the maximum number of file handles:
- Edited the C startup source file CRT0DAT.ASM (found in the <C600_ROOT>\SOURCE\STARTUP directory) and changed the line:_NFILE_ = 20
so that _NFILE_ is set to the desired maximum (100) as shown below:

_NFILE = 100

- Also uncommented are three lines of code in CRT0DAT.ASM so that a section of conditional code is enabled whenever the value of _NFILE_ is changed. These lines of code are denoted with comments in the CRT0DAT.ASM file (i.e., ; Increase File Handle Count).

To increase the maximum number of streams:
-Edited the C source file _FILE.C and changed the line:

#define _NFILE_ 20

to set _NFILE_ to the desired maximum (100) as shown below:

#define _NFILE_ 100

After modifying the C startup source files, the batch file STARTUP.BAT was used to recompile the files. Before executing the STARTUP.BAT batch file, however, the macro compiler MASM was installed to compile the CRT0DAT.ASM macro, and the MAKEFILE for the startup code was edited to add "-FPi87" to the CFLAGS options (this option must be used to indicate that an 80x87 coprocessor will be used instead of using an emulator library). The following command was used to execute the STARTUP.BAT batch file:

```
STARTUP DOS L
```

The large-model object files (CRT0DAT.OBJ and _FILE.OBJ) were created by the MAKEFILE in the subdirectory <C600_ROOT>\SOURCE\STARTUP\L\DOS. These object files were then replaced in the large-model C library, LLIBC7.LIB using the following commands:

```
LIB LLIBC7.LIB -+ ...\source\startup\l\dos\_file
LIB LLIBC7.LIB -+ ...\source\startup\l\dos\crt0dat
```

Then the MSD executable was linked with the new library.

TLIB Version Control software was used as the source code configuration management system. In every source code directory, there is a \VERSIONS subdirectory that contains the TLIB library files (one for each source file). The following command was used to store the latest working version of a source file in its appropriate TLIB library file:

```
tlib p versions
> Update
> file_to_update.c
```

The TLIB command is contained in the TLIB installation directory. For information on how to store and retrieve files from the TLIB library files, reference the TLIB User's Guide and Reference Manual.

MSD Source File Listing

See **Appendix A -- Figure 5** for a diagram of the MSD directory structure and source code. A description of each directory and a listing of the files/function names contained in each directory are listed below.

MSD Root Directory

Below is a list of the files included in the top level of the MSD directory structure:

ui.exe - MSD executable
imagedbs.ddl - MSD schema file
imagedbs.dbd - MSD dictionary file
preproc.exe - MSD preprocessor
browse.com - File browsing utility

MAIN Function

The main function for the MSD system and other related files are found in the MAINSRC directory.

<MSD_ROOT>\MAINSRC

Below is a list of the C source files / function names and other files included in this directory:

ui.c - C source file containing main()
- main()
- User_login()

ui.obj - Object file produced when UI.C is compiled
ui.lnk - Link file used by the Plink86 linker to set up the overlay structure
ui.map - Map file created by the Plink86 linker

The executable produced by linking UI.C is UI.EXE; this file is moved to the MSD Root Directory where all executables are located.

Library Functions

Object libraries are used to organize the MSD source code. The files contained in each library are listed below.

<MSD_ROOT>\LIB\UI

This directory consists of two object libraries:

UI1.LIB
UI2.LIB

These libraries contain functions for implementing the user interface (UI) for the MSD system. These functions handle all input/output for the MSD system, such as accepting keyboard input from user and displaying output to the screen. The MAKEFILE in this directory will build both of these libraries.

Below is a list of the C source files / function names included in the UI1.LIB object library:

addcal.c
- UI_add_cal_panel_record()
addchrom.c
- UI_add_chromo_record()
addata.c
- UI_add_data()
addce.c
- UI_add_dce_record()
addgloss.c
- UI_add_gloss_record()
addimags.c
- UI_add_images_record()
addmet.c
- UI_add_met_record()
addobjct.c
- UI_add_object_record()
addrad.c
- UI_add_radtherm_record()
addscene.c
- UI_add_scene_record()
addsite.c
- UI_add_site_record()
addspect.c
- UI_add_spectrad_record()
chek_vol.c
- UI_check_volume()
disp_err.c
- UI_display_error()
dispdata.c
- UI_display_data_file()

files.c
- Draw_files_screen()
- Handle_files()
- Database_error()
- Directory_error()
- Post_directory()
- Open_DB_files()
- Create_DB_files()
- Close_DB_files()
ops.c
- Draw_operations_screen()
- Handle_operations()
- Draw_records_screen()
- Handle_records()
seldimag.c
- UI_select_data_for_images()
selobscn.c
- UI_select_objects_in_scenes()
seldobj.c
- UI_select_data_for_object()
seldsite.c
- UI_select_data_for_site()
- UI_get_data_type_string()
seldscen.c
- UI_select_data_for_scene()
selscnob.c
- UI_select_scenes_with_objects()
selunobj.c
- UI_select_unique_objects()
sel_cal.c
- UI_select_calpaneldata_record()
sel_chro.c
- UI_select_chromo_record()
sel_data.c
- UI_select_data()
sel_dce.c
- UI_select_dce_record()
sel_glos.c
- UI_select_gloss_record()
sel_imgs.c
- UI_select_images_record()
sel_met.c
- UI_select_metdata_record()
sel_obj.c
- UI_select_object_record()
sel_rad.c
- UI_select_radtherm_record()
sel_rec.c
- UI_select_record()
sel_scen.c
- UI_select_scene_record()
sel_site.c
- UI_select_site_record()
sel_spec.c
- UI_select_spectrad_record()
utils.c
- Erase_command_line()
- Erase_message_line()
- Erase_query_display()
- Erase_operation()
- Erase_DB_filename()
- Draw_login_screen()
- Draw_season_line()
- Handle_seasons()
- UI_handle_subrecord_screen()

- UI_handle_data_screen()
- UI_handle_ground_data_screen()
- UI_handle_physical_data_screen()
- UI_handle_met_screen()
- UI_handle_cal_panel_screen()
- UI_handle_chromo_screen()
- UI_handle_spectrad_screen()
- UI_handle_radtherm_screen()
- UI_handle_gloss_screen()
- UI_prompt_for_confirm()

viewops.c
- UI_view_operation()

writview.c
- UI_write_view_file()

Below is a list of the C source files / function names included in the ui2.lib object library:

delete.c
- UI_delete_record()
- UI_delete_data_file()
- UI_prompt_for_copy_or_delete()

query.c
- Query_DB()
- UI_display_images_info()

<MSD_ROOT>\LIB\IMAGEDBS

This directory consists of the following object library:

IMAGEDBS.LIB

This library contains functions specifically for manipulating an MSD database. These functions handle making specific queries to an MSD database, adding specific MSD data to the database, and other functions written specifically for the database structure supported by the MSD database.

Below is a list of the C source files / function names included in the IMAGEDBS.LIB object library:

add_ccd.c
- I_add_ccd_record()

add_cal.c
- I_add_cal_panel_record()

add_chro.c
- I_add_chromo_record()

add_dce.c
- I_add_dce_record()

add_glos.c
- I_add_gloss_record()

add_imgs.c
- I_add_images_record()

add_met.c
- I_add_met_record()

add_obj.c
- I_add_object_record()

add_rad.c
- I_add_radtherm_record()

add_scen.c
- I_add_scene_record()

add_site.c
- I_add_site_record()

add_spec.c
- I_add_spectrad_record()

analognm.c
- I_create_analog_data_name()

cpdbsrec.c
- I_copy_dbs_record()

get_cal.c
- I_get_calpaneldata_records()

get_chro.c
- l_get_chromo_records()

get_glos.c
- l_get_gloss_records()

get_rad.c
- l_get_radtherm_records()

get_spec.c
- l_get_spectrad_records()

get_dce.c
- l_get_dce_records()

get_imag.c
- l_get_one_images_record()

get_imgs.c
- l_get_images_records()

get_obj.c
- l_get_object_records()

get_met.c
- l_get_metdata_records()

get_scen.c
- l_get_scene_records()

get_site.c
- l_get_site_records()

getcalsn.c
- l_get_owner_cal_panel_for_scene()

getobscn.c
- l_get_objects_in_scenes()

getradob.c
- l_get_radtherm_records_for_object()

getscnid.c
- l_get_scene_id()

getscnob.c
- l_get_scenes_with_objects()

getunobj.c
- l_get_unique_objects()

gtavlobj.c
- l_get_available_objects()

gtavlspc.c
- l_get_available_spectrums()

imgfilsp.c
- l_create_image_filespec()

imgspect.c
- l_get_image_spectrums()

nextcal.c
- l_get_next_cal_panel_filename()

nextchro.c
- l_get_next_chromo_filename()

nextglos.c
- l_get_next_gloss_filename()

nextmet.c
- l_get_next_met_filename()

nextrad.c
- l_get_next_radtherm_filename()

nextspec.c
- l_get_next_spectrad_filename()

ownrcal.c
- l_get_owner_cal_panel_record()

ownrchro.c
- l_get_owner_chromo_record()

ownrglos.c
- l_get_owner_gloss_record()

ownrmet.c
- l_get_owner_met_record()

ownrscen.c
- l_get_owner_scene_record()

ownrsite.c
- l_get_owner_site_record()

ownrspec.c
- l_get_owner_spectrad_record()

proc_sel.c
- l_process_select()

updscnid.c
- l_update_scene_id()

<MSD_ROOT>\LIB\RDMDBS

This directory consists of the following object library:

RDMDBS.LIB

This library contains functions for manipulating any Raima Data Manager (RDM) database. These functions handle opening and closing a database, adding records, deleting records, and other basic database functions.

Below is a list of the C source files / function names included in the RDMDBS.LIB object library:

add_cdvl.c
- R_add_codelist_value()

add_rec.c
- R_add_record()

all_cods.c
- R_get_all_codelists()

chkmembr.c
- R_check_for_member()

chkowner.c
- R_check_for_owner()

clos_dbs.c
- R_close_database()

conn_rec.c
- R_connect_record()

dbs_err.c
- R_database_error()

dcon_rec.c
- R_disconnect_record()

del_rec.c
- R_delete_record()

get_cdnm.c
- R_get_codelist_number()

get_cdvl.c
- R_get_codelist_value()

log_err.c
- R_log_error()

mod_rec.c
- R_modify_record()

nuloutpt.c
- R_set_output_null()

open_dbs.c
- R_open_database()

<MSD_ROOT>\LIB\UTIL

This directory consists of the following object library:

UTIL.LIB

This library contains utility functions. These functions handle copying files, getting system time and date and other general utility functions. Most of the function make calls to DOS functions.

Below is a list of the C source files / function names included in the UTIL.LIB object library:

dattojul.c
- U_convert_date_to_julian()

jultodat.c
- U_convert_julian_to_date()

copyfile.c
- U_copy_file()

datetime.c
- U_get_date_time_string()
freespac.c
- U_get_free_space()
lastdriv.c
- U_get_last_drive()

<MSD_ROOT>\LIB\VICTOR

This directory contains the library modules required for displaying 24-bit uncompressed TIFF images using the VESA SVGA or Paradise SVGA 256-color video display mode that support a resolution of 640 x 480. There are two .LIB files located in this directory: MSVIC.LIB, the Victor Image Processing library for Microsoft C compilers, supplied by Catenary Systems, and VICTORIF.LIB, which makes calls to Victor functions to display TIFF images, and handles errors encountered when attempting to display an image.

This directory consists of the following object libraries:

MSVIC.LIB
VICTORIF.LIB

The MSVIC.LIB library is the Victor Image Processing library for Microsoft C compilers supplied by Catenary Systems. The VICTORIF.LIB library contains functions for displaying 24-bit uncompressed TIFF images by making calls to Victor functions.

Below is a list of the C source files / function names included in the VICTORIF.LIB object library:

disp_tif.c
- V_display_tiff_image()
vic_err.c
- V_get_viclib_error()

Header Files

In addition to the object libraries, there are C header files containing preprocessor definitions, the declaration of data structures and types, definition of constants, and function prototypes.

<MSD_ROOT>\INCLUDE

All of the C header files for the MSD system are located in the INCLUDE directory and are listed below:

i_define.h
i_proto.h
imagedbs.h
r_define.h
r_extern.h
r_global.h
r_proto.h
u_proto.h

uicon.h
uiconex.h
uiconst.h
uiopex.h
uiops.h
uiproto.h
v_proto.h

Preprocessor

A preprocessing utility, PREPROC.EXE, has been provided for formatting raw physical and ground truth data files for entry into the MSD. This utility was implemented utilizing Turbo Pascal version 5.5 by Borland International and the TechnoJock's Turbo Toolkit version 5.0 by TechnoJock Software, Inc. The Pascal source code for the preprocessor is located in the PREPROC directory.

<MSD_ROOT>\PREPROC

Below is a list of files containing the Pascal source code for the preprocessor utility:

ccd_sort.pas
dup_line.pas
enterdat.pas
globals.pas
preproc.pas
selectid.pas

Also located in this directory are a MAKEFILE for building the preprocessor executable and the textfile PREPROC.DOC containing detailed instructions on using the utility.

Browser

For the MSD to display physical and ground truth data files, the program must make calls to a stand-alone file browsing utility. A public domain browser, BROWSE.COM, has been included in the MSD root directory. Assembler source code and technical notes about the browser are included in the BROWSER directory.

<MSD_ROOT>\BROWSER

Below is a list of files contained in this directory:

browse.com - Browser utility
browse.asm - Assembler source code
browse2.txt - Programming credits and technical notes

MSD Utilities

Several utility programs have been provided for use with the MSD software. These programs are in the UTILITY directory.

<MSD_ROOT>\UTILITY

Below is a list of the programs included in this directory:

- **volume.exe** - Utility for labeling optical disks
  disptiff.exe - Utility for displaying a TIFF image file (24-bit uncompressed)
  imagesys.exe - Utility for restoring imagedbs.sys file if corrupted

4 Summary

Design and development of the MSD database have been completed. The MODIFY command was not implemented due to memory and time constraints. Currently plans are to port the database code to the Silicon Graphics Inc. (SGI) platform and integrate it into the Input Module of the C2D2E2 application which is hosted on the SGI platform.

5 References

Raima Corporation. (1991). "db_VISTA reference manual, version 3.2.1," Bellevue, WA.

Raima Corporation. (1991). "db_VISTA user's guide, version 3.2.1," Bellevue, WA.

Raima Corporation. (1991). "db_QUERY user's guide, version 3.2.1," Bellevue, WA.

Raima Corporation. (1991). "db_REVISE user's guide, version 3.2.1," Bellevue, WA.

Microsoft Corporation. (1990). "Microsoft C reference manual, version 6.0," Redmond, WA.

Catenary Systems. (1992). "Victor Library reference manual," St. Louis, MO.

APPENDIX A:

FIGURES

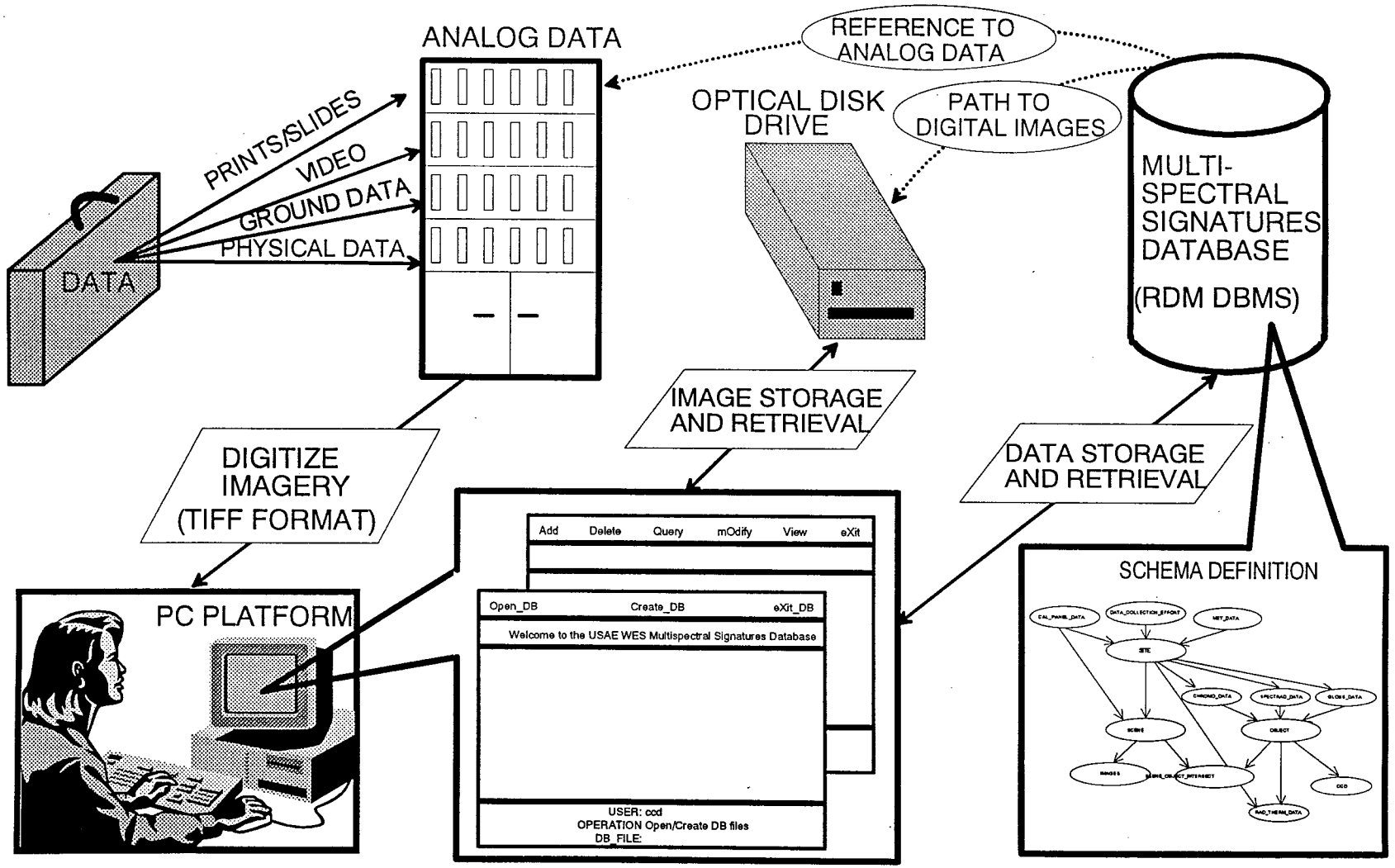


Figure 1. Multispectral Signature Database

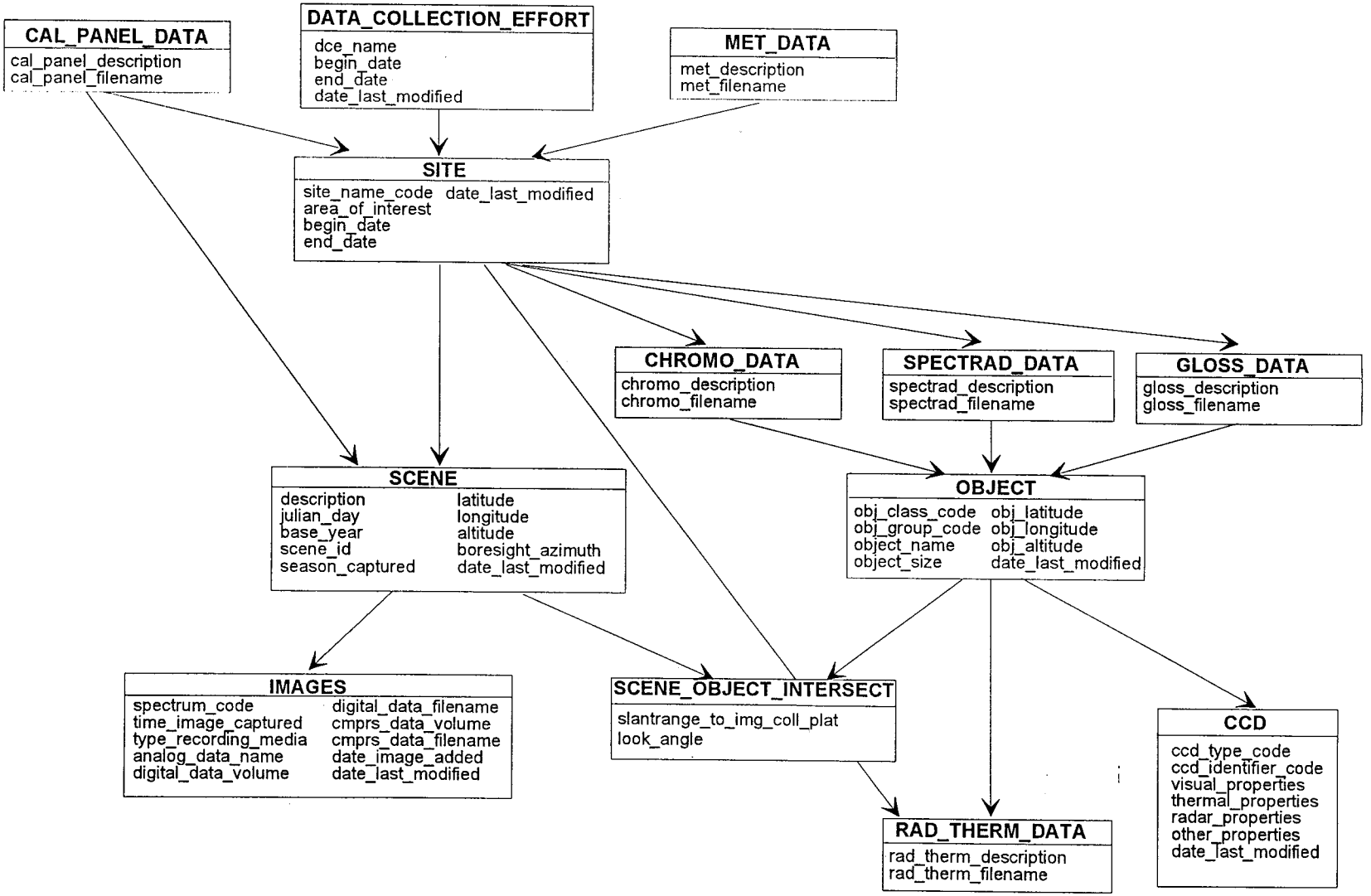


Figure 2. MSD Schema Definition

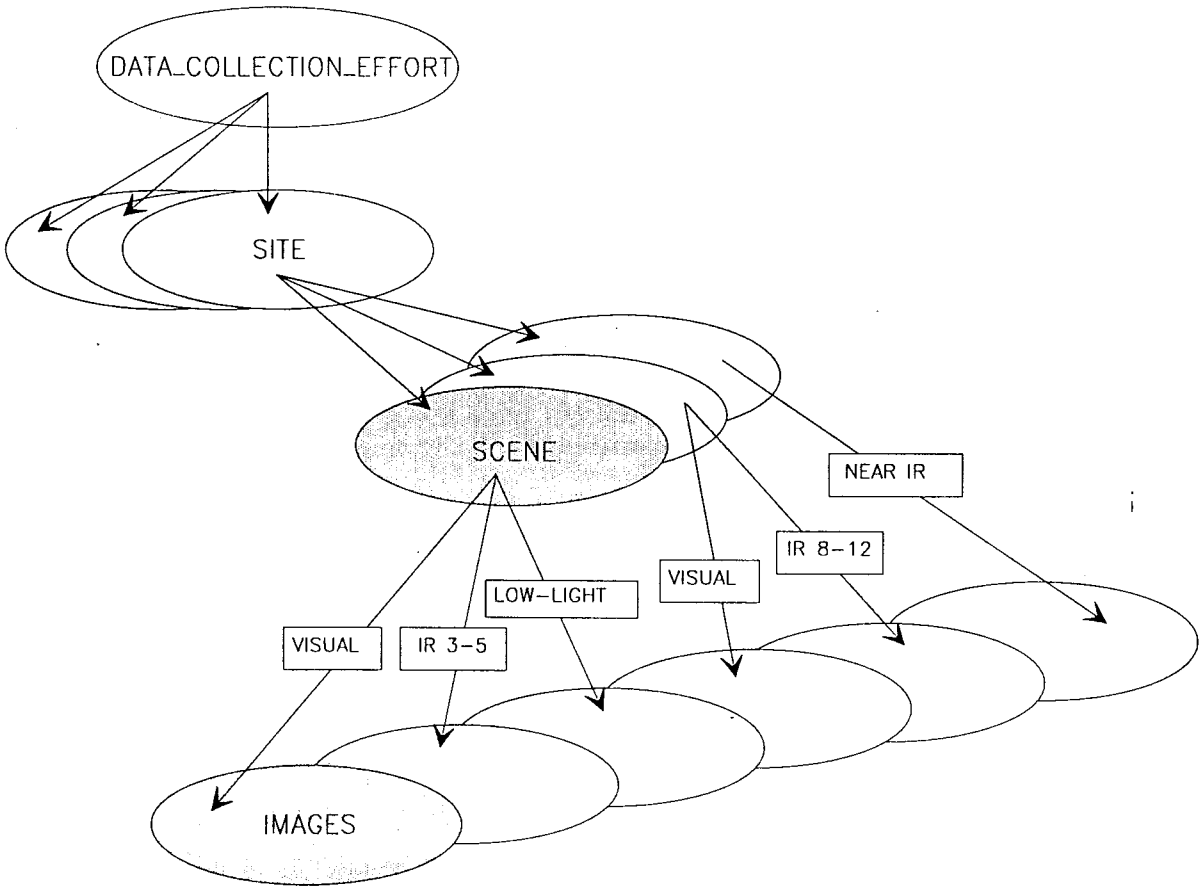


Figure 3. Scene-Images Relationship

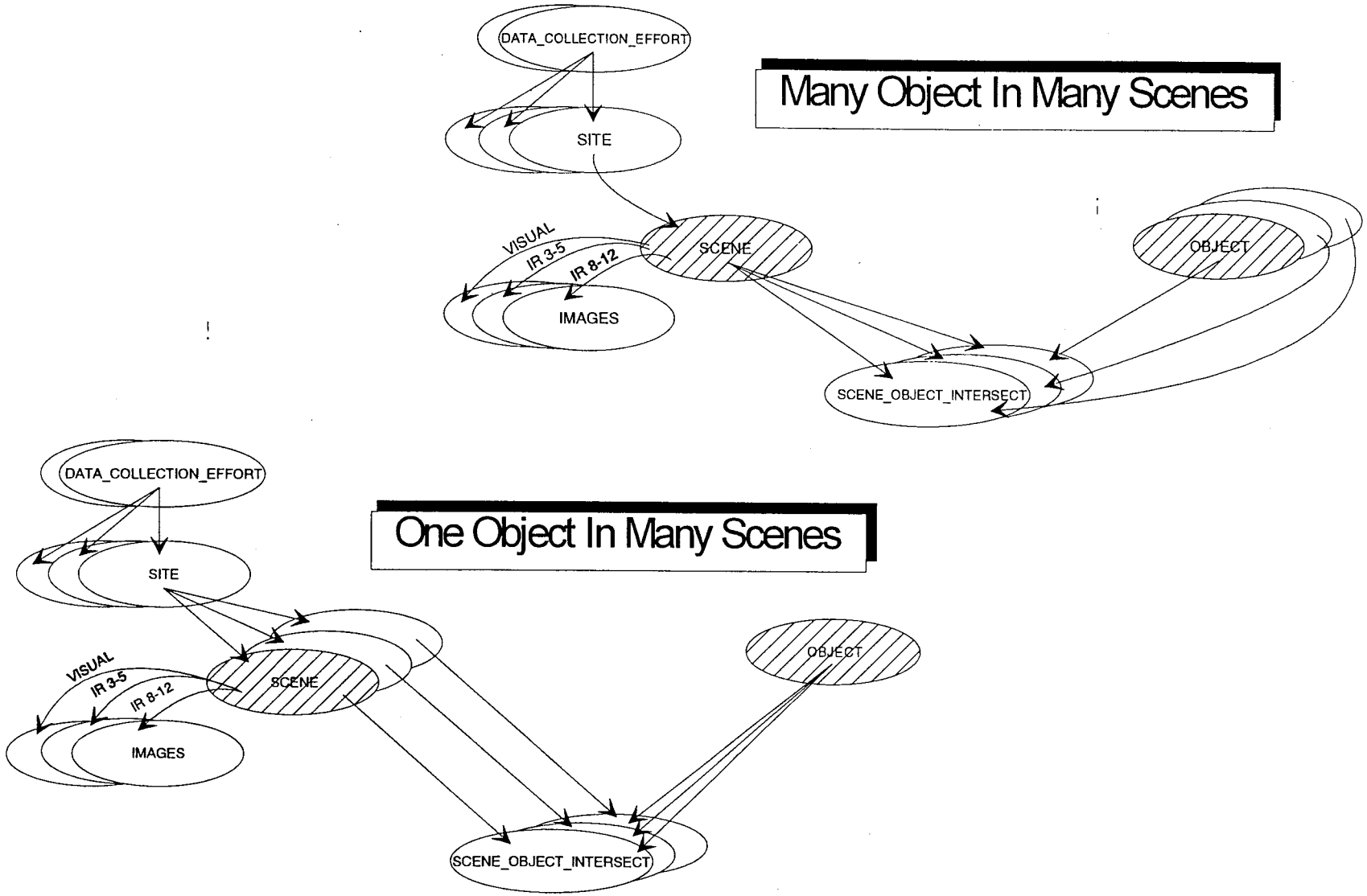


Figure 4. Scene-Object Relationship

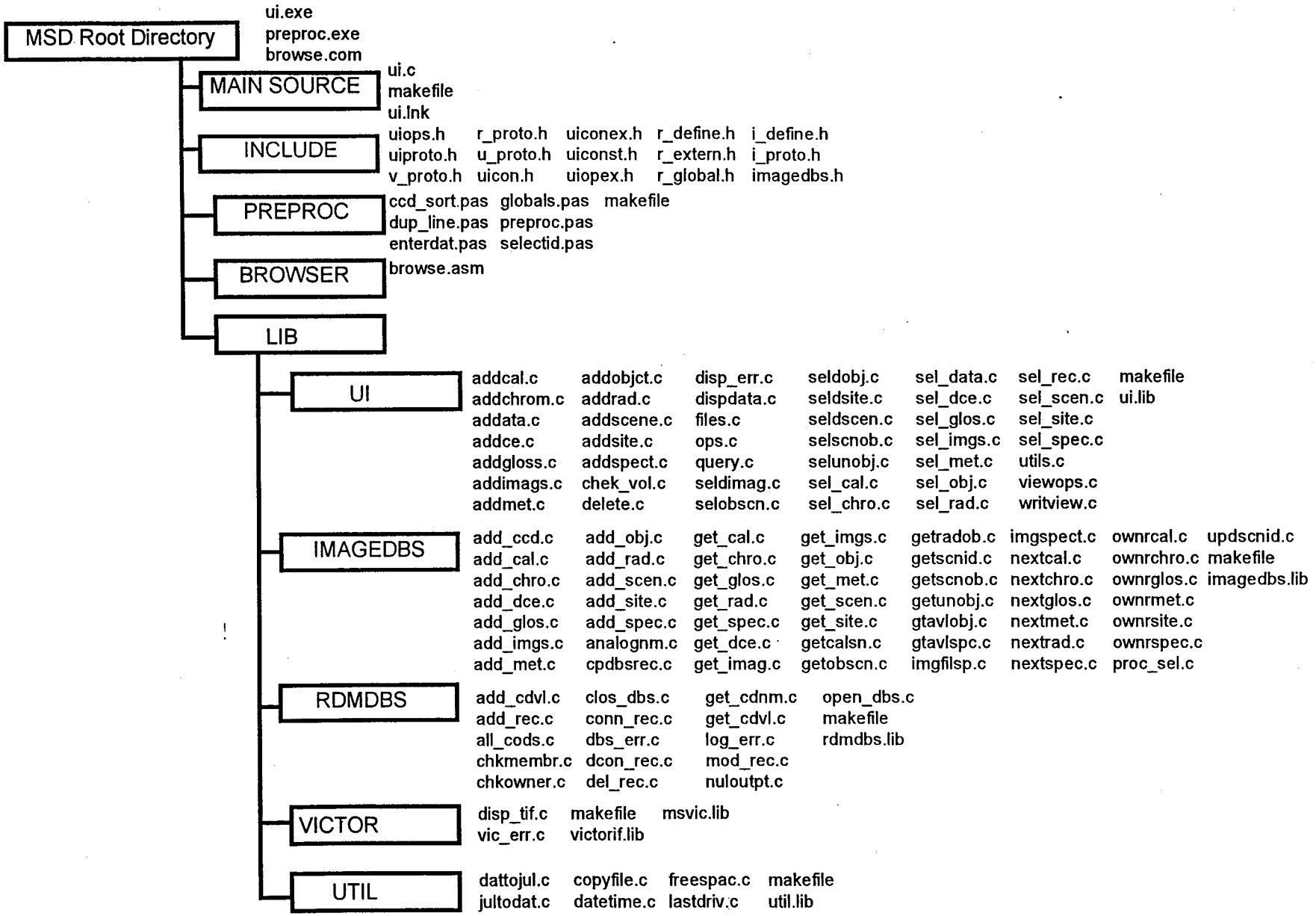


Figure 5. MSD Directory Structures & Source Code

APPENDIX B:

MSD DATABASE SCHEMA DEFINITION

```
/*************************************************************************/
/* WES CCD Research Group Image Database - imagedbs.ddl */
/*************************************************************************/

database imagedbs
{
/*********************/
/* File Statements */
/*********************/

   data file [1024] ddceffrt = "ddceffrt.dat" contains
            data_collection_effort;
   data file [2048] dsite    = "dsite.dat"    contains
            site;
   data file [4096] dscene   = "dscene.dat"   contains
            scene;
   data file [4096] dimages  = "dimages.dat"  contains
            images;
   data file [4096] dscenobj = "dscenobj.dat" contains
            scene_object_intersect;
   data file [4096] dobject  = "dobject.dat"  contains
            object;
   data file [4096] dccd     = "dccd.dat"     contains
            ccd;
   data file [1024] dmetdata = "dmetdata.dat" contains
            met_data;
   data file [1024] dcalpanl = "dcalpanl.dat" contains
            cal_panel_data;
   data file [1024] dgloss   = "dgloss.dat"   contains
            gloss_data;
   data file [1024] dspectrd = "dspectrd.dat" contains
            spectrad_data;
   data file [1024] dchromo  = "dchromo.dat"  contains
            chromo_data;
   data file [1024] dradthm  = "dradthm.dat"  contains
            rad_therm_data;
   data file [2048] cd_site  = "cd_site.dat"  contains
            site_name_codelist;
   data file [512]  cd_objcl = "cd_objcl.dat" contains
            obj_class_codelist;
   data file [1024] cd_objgr = "cd_objgr.dat" contains
            obj_group_codelist;
   data file [1024] cd_ccdtp = "cd_ccdtp.dat" contains
            ccd_type_codelist;
   data file [2048] cd_ccdid = "cd_ccdid.dat" contains
```

```
            ccd_identifier_codelist;
data file [512]  cd_spect = "cd_spect.dat" contains
            spectrum_codelist;
key file  [1024] kdcename = "kdcename.key" contains
            dce_name;
key file  [1024] kmisc    = "kmisc.key"    contains
            spectrum_code,
      spectrum_codelist.codenum,
      spectrum_codelist.codeval,
      spectrum_codelist.second_codeval;
key file  [1024] ksite    = "ksite.key"    contains
            site_name_code,
      site_name_codelist.codenum,
      site_name_codelist.codeval,
      site_name_codelist.location_codeval;
key file  [1024] kobjclas = "kobjclas.key" contains
            obj_class_code,
      obj_class_codelist.codenum,
      obj_class_codelist.codeval,
      obj_class_codelist.second_codeval;
key file  [1024] kobjgrp  = "kobjgrp.key"  contains
            obj_group_code,
      obj_group_codelist.codenum,
      obj_group_codelist.codeval;
key file  [1024] kobjname = "kobjname.key" contains
            object_name;
key file  [1024] kccdtype = "kccdtype.key" contains
            ccd_type_code,
      ccd_type_codelist.codenum,
      ccd_type_codelist.codeval;
key file  [1024] kccdid   = "kccdid.key"   contains
            ccd_identifier_code,
      ccd_identifier_codelist.codenum,
      ccd_identifier_codelist.codeval;
key file  [1024] kscndesc = "kscndesc.key" contains
            scene.description;
key file  [1024] kdatecap = "kdatecap.key" contains
            date_captured;
key file  [1024] kseason  = "kseason.key"  contains
            season_captured;
key file  [1024] ksceneid = "ksceneid.key" contains
            scene_id;
key file  [1024] kmetdesc = "kmetdesc.key" contains
            met_description;
key file  [1024] kcaldesc = "kcaldesc.key" contains
```

```
            cal_panel_description;
   key file  [1024] kchrdesc = "kchrdesc.key" contains
            chromo_description;
   key file  [1024] kglsdesc = "kglsdesc.key" contains
            gloss_description;
   key file  [1024] kspcdesc = "kspcdesc.key" contains
            spectrad_description;
   key file  [1024] kraddesc = "kraddesc.key" contains
            rad_therm_description;


/*********************************/
/* Record Type Declarations */
/*********************************/

   record data_collection_effort
   {
    unique key  char          dce_name[81];
            unsigned long  begin_date;
            unsigned long  end_date;
            unsigned long  date_last_modified;
   }

   record site
   {
         key  unsigned short site_name_code;
            char          area_of_interest[81];
            unsigned long  begin_date;
            unsigned long  end_date;
            unsigned long  date_last_modified;
   }

   record scene
   {
    unique key  char          description[81];
         key  struct
              {
               unsigned short   julian_day;
               unsigned short   base_year;
              }            date_captured;
    unique key  char          scene_id[9];
         key  char          season_captured[7];
            float         latitude;
            float         longitude;
            float         altitude;
```

```
        char        boresight_azimuth[5];
        unsigned short calibrated_data;
        unsigned long  date_last_modified;
}

record images
{
    key  unsigned short spectrum_code;
        unsigned long  time_image_captured;
        char        type_recording_media[21];
        char        analog_data_name[31];
        char        digital_data_volume[12];
        char        digital_data_filename[41];
        char        cmprs_data_volume[12];
        char        cmprs_data_filename[41];
        unsigned long  date_image_added;
        unsigned long  date_last_modified;
}

record scene_object_intersect
{
        float   slantrange_to_img_coll_plat;
        float   look_angle;
}

record object
{
    key  unsigned short obj_class_code;
    key  unsigned short obj_group_code;
 unique key  char        object_name[81];
        char        object_size[31];
        unsigned long  date_last_modified;
        float obj_latitude;
        float obj_longitude;
        float obj_altitude;
}

record ccd
{
    key  unsigned short ccd_type_code;
    key  unsigned short ccd_identifier_code;
        char        visual_properties[81];
        char        thermal_properties[81];
        char        radar_properties[81];
        char        other_properties[81];
```

```
        unsigned long  date_last_modified;
}

record met_data
{
 unique key  char        met_description[81];
        char        met_filename[41];
}

record cal_panel_data
{
 unique key  char        cal_panel_description[81];
        char        cal_panel_filename[41];
}

record chromo_data
{
 unique key  char        chromo_description[81];
        char        chromo_filename[41];
}
record gloss_data
{
 unique key  char        gloss_description[81];
        char        gloss_filename[41];
}

record spectrad_data
{
 unique key  char        spectrad_description[81];
        char        spectrad_filename[41];
}

record rad_therm_data
{
 unique key  char        rad_therm_description[81];
        char        rad_therm_filename[41];
}

/* Codelist Record Type Declarations */

record site_name_codelist
{
 unique key  unsigned short codenum;
 unique key  char        codeval[51];
      key  char        location_codeval[51];
```

```
}

record obj_class_codelist
{
 unique key  unsigned short codenum;
 unique key  char         codeval[51];
 unique key  char         second_codeval[6];
}

record obj_group_codelist
{
 unique key  unsigned short codenum;
 unique key  char         codeval[51];
}

record ccd_type_codelist
{
 unique key  unsigned short codenum;
 unique key  char         codeval[31];
}

record ccd_identifier_codelist
{
 unique key  unsigned short codenum;
 unique key  char         codeval[51];
}

record spectrum_codelist
{
 unique key  unsigned short codenum;
 unique key  char         codeval[21];
 unique key  char         second_codeval[3];
}


/***********************/
/* Set declarations */
/***********************/

 set dce_to_site_link
 {
  order  last;
  owner  data_collection_effort;
  member site;
 }
```

```
set site_to_scene_link
{
 order  last;
 owner  site;
 member scene;
}

set scene_to_images_link
{
 order  last;
 owner  scene;
 member images;
}

set scene_to_object_link
{
 order  last;
 owner  scene;
 member scene_object_intersect;
}

set object_to_scene_link
{
 order  last;
 owner  object;
 member scene_object_intersect;
}

set object_to_ccd_link
{
 order  last;
 owner  object;
 member ccd;
}

set met_to_site_link
{
 order  last;
 owner  met_data;
 member site;
}

set cal_to_site_link
{
 order  last;
```

```
 owner  cal_panel_data;
 member site;
}

set cal_to_scene_link
{
 order  last;
 owner  cal_panel_data;
 member scene;
}

set site_to_chromo_link
{
 order  last;
 owner  site;
 member chromo_data;
}

set chromo_to_object_link
{
 order  last;
 owner  chromo_data;
 member object;
}

set site_to_gloss_link
{
 order  last;
 owner  site;
 member gloss_data;
}

set gloss_to_object_link
{
 order  last;
 owner  gloss_data;
 member object;
}

set site_to_spectrad_link
{
 order  last;
 owner  site;
 member spectrad_data;
}
```

```
    set spectrad_to_object_link
    {
     order   last;
     owner   spectrad_data;
     member  object;
    }

    set site_to_radtherm_link
    {
     order   last;
     owner   site;
     member  rad_therm_data;
    }

    set object_to_radtherm_link
    {
     order   last;
     owner   object;
     member  rad_therm_data;
    }

} /* End IMAGEDBS.DDL */
```

APPENDIX C:

MSD SOURCE CODE

The MSD Source Code is not reprinted in this technical report due to the large size of the program. Copies will be provided, upon request, from the following address:

Commander and Director
USAE Waterways Experiment Station
ATTN: CEWES-SS-C/Gerardo I. Velazquez
3909 Halls Ferry Road
Vicksburg, Mississippi 39180-6199

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1995	Report 1 of a series

4. TITLE AND SUBTITLE
Development of a Multispectral Signatures Database for the Camouflage, Concealment, and Deception Design and Evaluation Environment (C2D2E2); Report 1, Technical Description

5. FUNDING NUMBERS
Contract No.
DACA39-91-C-0042

6. AUTHOR(S)
Sandy D. Bratcher, Douglas P. Rousell, Gerardo I. Velázquez

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Nichols Research Corporation, Vicksburg, MS 39182-0186;
U.S. Army Engineer Waterways Experiment Station
3909 Halls Ferry Road, Vicksburg, MS 39180-6199

8. PERFORMING ORGANIZATION REPORT NUMBER
Technical Report SL-95-21

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Corps of Engineers, Washington, DC 20314-1000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Available from National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This study provides a technical description of a unique multispectral signature database system develop by USAE Waterways Experiment Station (WES). The database will handle the great amount of multispectral imagery, ground truth and physical data collected by WES. The Multispectral Signatures Database will organize and catalog large number of images, associating appropriate attribute data with each image, associating images of the same scene in different spectral bands, and retrieval of related images and their respective attribute data. The Raima Data Manager (RDM) was used for the underlying Database Management System. This report summarizes the technical effort and the results achieved of the Multispectral Signatures Database.

14. SUBJECT TERMS
Camouflage Materials
Database
Database Definition Language
Database Management
Database Model
Materials Properties
TIFF Images

15. NUMBER OF PAGES
55

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED		