(200)
R290
no 96-56

U.S. DEPARTMENT OF THE INTERIOR
U.S. GEOLOGICAL SURVEY

Remote Video Monitoring Systems

by

John W. Haines and Bill Townsley

Open File Report 9656

This report is preliminary and has not been reviewed for conformity with U.S. Geological Survey editorial standards

U.S. Geological Survey, Center for Coastal Geology
St. Petersburg, Florida 33701

Remote Video Monitoring Systems

John W. Haines and William W. Townsley

SUMMARY

Remote Video Monitoring Systems are a cost effective means of acquiring video data from remote locations and returning them to a central computer for processing and analysis. Video data has proven to be a valuable source of information in a variety of coastal studies. Guidelines and procedures used in developing Remote Video Monitoring (RVM) systems and the usefullness of video data are described in this document including specific examples from the RVM sytem in operation at the Center for Coastal Geology in St. Petersburg, FL.

REMOTE VIDEO MONITORING SYSTEMS
TABLE OF CONTENTS

LIST OF FIGURES

LIST OF IMAGES

INTRODUCTION

Remote Video Monitoring (RVM) Systems provide a means of automatically acquiring video data from remote locations and returning them to a central location for processing and analysis. Since 1991, the U.S. Geological Survey's Center for Coastal Geology has been developing RVM capabilities in cooperation with coastal researchers at Oregon State University (OSU). The Coastal Imaging Lab (CIL) at OSU has pioneered the use of video data acquisition for studies of coastal processes. Video monitoring systems, when used to observe beach and nearshore systems, have successfully provided lengthy continuous records of change in a variety of coastal features such as shorelines and offshore sand bars (Lippmann and Holman, 1989). Video data have also been utilized in the study of a wide range of nearshore processes including wave propagation and breaking, and the dynamics of the swash zone (Lippmann and Holman, 1991).

The USGS/OSU cooperative has focussed on further application of video methods to the collection of continuous records in remote locations. The goal has been to provide an economical, rugged, and reliable data collection and analysis package requiring minimal maintenance for use in areas that are not suitable for frequent oversight. The results of this cooperative effort are described in the following document.

The video data collection platform, or field station, is described in Figure 1. These low-cost, PC-based data acquisition systems return digital video data (and data from other sensors) from the monitoring location over commercial or cellular telephone links. Power is provided by either commercial power or solar/battery systems. Field stations designed using the guidelines described in this document have been in continual operation for several years through environmental extremes ranging from sub-freezing temperatures to near hurricane storm conditions.

Data retrieval, archiving, and collection control is performed by the RVM central laboratory computer, currently any Unix based workstation (Figure 2). The laboratory computer also provides a flexible platform for the analysis and distribution of the resulting data. From the onset, the objectives of this effort have included the development of streamlined data distribution paths and widely available analysis tools. Combined, the lab computer and field station(s) form a wide area network linked by standard communications protocols. This network provides many of the conveniences of local computer networks and is much less expensive to implement.

The structure of the complete RVM system is based on a number of systems of specified functionality, rather than on specific hardware products. Using general and specific examples from the RVM system used by the USGS Coastal Center, this report will provide guidance to anyone interested in either using available data from existing systems, or in developing similar systems for specific applications, video data and its application to coastal research will be reviewed, hardware and software solutions are discussed, and processing techniques described.

VIDEO DATA

Video provides a 2-dimensional image of a limited spatial region at a particular moment in time. Any feature which has a visual signature is amenable to video monitoring (Holman *et al*, 1993). Quantification of the data is based on the magnitude of the returned signal and on the location within the field of view.

Figure 1: Central Laboratory Computer Schematic

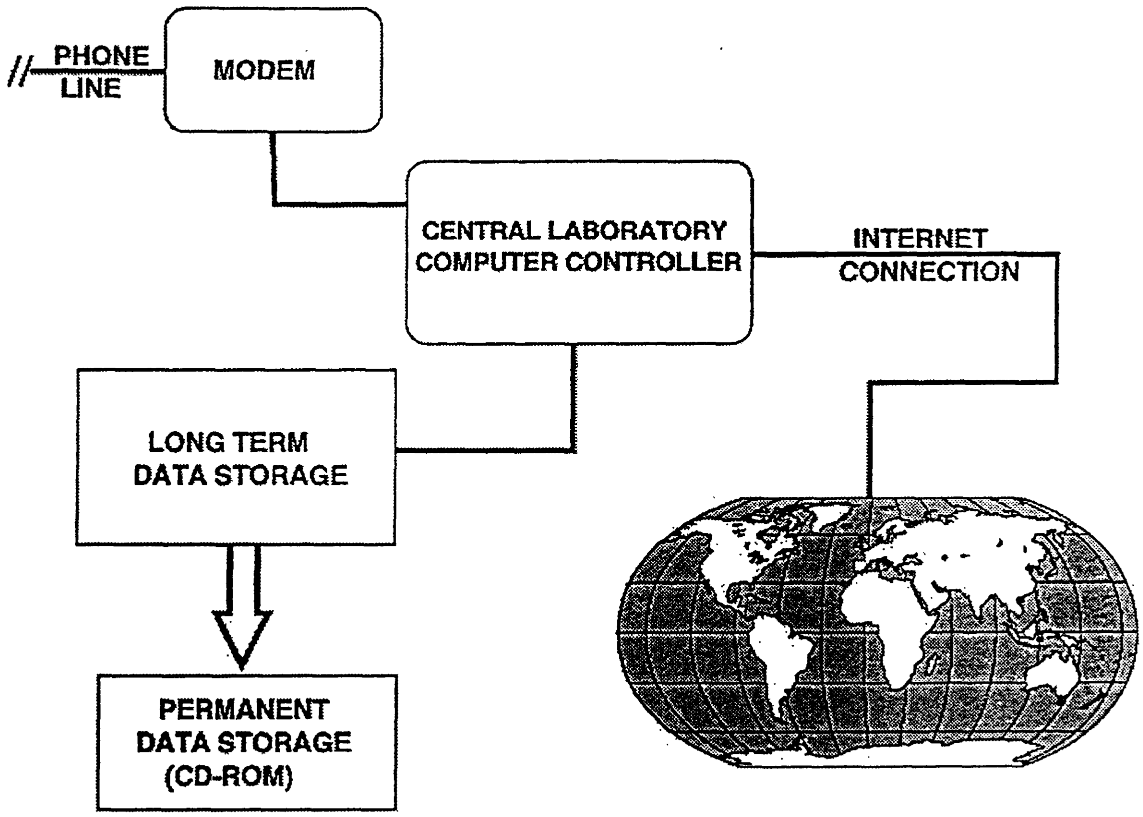
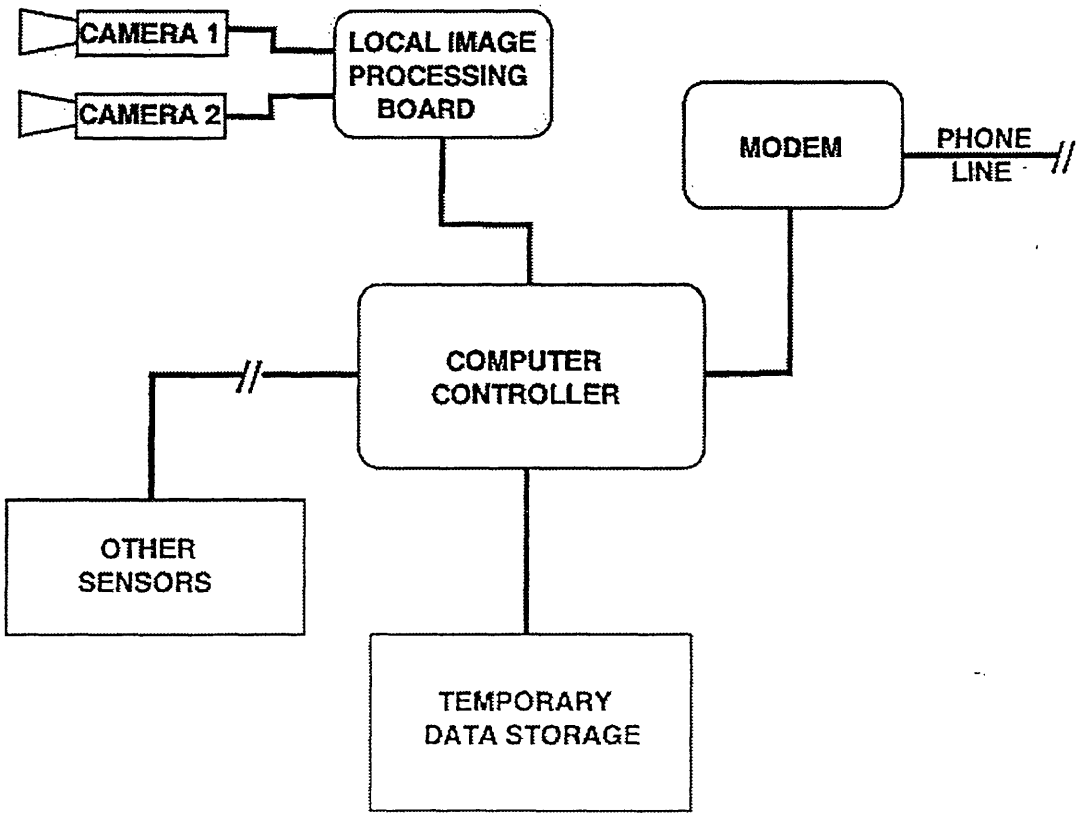

PHONE LINE
MODEM
CENTRAL LABORATORY COMPUTER CONTROLLER
INTERNET CONNECTION
LONG TERM DATA STORAGE
PERMANENT DATA STORAGE (CD-ROM)

Figure 2: Remote Video Monitoring Field Station Schematic



A digital image consists of a matrix of numeric values representing the individual picture elements (pixels). The numeric value of each pixel represents the intensity (lightness/darkness or color) of the captured feature. Thus features which exhibit high contrast, such as shorelines and breaking waves, may be located and followed in time (Holman and Guza, 1984; Lippmann and Holman, 1989). As an example, Image 1 shows the marked difference between the highly reflective swash zone and the underlying sand on a barrier beach with a section of the swash limit marked by the black diamonds. A series of such images allows the instantaneous shoreline position to be traced as it varies with time.

Video imagery may also be averaged creating a digital "time exposure" that identifies persistant features in the image frame. Images 2a and 2b show the contrasting information from the video snapshots (single images) and time exposure images (averaged over 12 minutes). The time exposure approach allows the identification the breaking wave zone, which corresponds to the location of sand bars, submerged shoals and channels (Lippmann and Holman, 1989).

Mapping the video image to a meaningful coordinate system is accomplished by tranforming the pixel (image) coordinates to a relevent ground coordinate system (e.g., Lippmann and Holman, 1989). The rectification of the image depends on the camera lens, camera position, local topography, and control points visible in the scene. Specifiction of the appropriate geometry variables allows the geometric transformation of the video image to a two-dimensional 'mapping' in ground coordinates. In practice, analagous to aerial photography analysis, ground control points with known locations within the image field are used to solve for the transformation geometry. Three or more known points permit a least squares solution for the transformation. Subsequently, image features may be viewed in a plan or map view. The results of such image rectification are shown in Images 3a & 3b. Successful transformation requires the specification of one coordinate dimension, thus mapping is possible for 2-dimensional features where the third dimension is constant. For example, shorelines and water surface features may be mapped provided the water level is known.

SYSTEM HARDWARE

Remote Video Monitoring field stations, called Argus stations, were originally developed at Oregon State University's Coastal Imaging Lab (Holman *et al*, 1993). They include video cameras, an IBM PC or PC clone with an add-on image processing circuit board, and a modem. Controlled by the computer, images are acquired by the image processor from the video cameras and stored on the PC's hard drive. Data transfer from the field station to the central laboratory computer is facilitated by the modem.

The first Argus station deployed by the USGS Coastal Center, voyeur02, is still active and utilizes a Dipix P360F Power Grabber installed in a 80486 PC clone. Two (2) Sony XC-77 compact, black & white video cameras provide video input to the PC through the Dipix board*. A 120 MB internal hard disk drive enables the station to store up to 3,000 images before data retrieval becomes necessary. Communications with the outside world are facilitated by a Telebit T2500 modem featuring automatic baud rate adjustment from 2400-19200 baud, error detection and correction, and effective handling of data transfers. Eight (8) MB of memory accommodate the software tools which allow the station to run automatically. Though none of the Argus stations presently in operation are identical, they typically use Dipix P360F Power Grabber image processing cards. This allows the exchange of image acquisition software and also simplifies trouble-shooting among the system operators.

The central laboratory computer can be any Unix based workstation with ample on-line storage and a modem. The laboratory computer located at the Coastal Center is a Sun SPARCstation2 running the Solaris 2.0 operating system. Like the field station, this workstation utilizes a Telebit modem for its aforementioned features and to take advantage of the low error/high transfer rates possible between Telebit modems. Though not required, a real time Internet connection allows the Internet community access to video data and related information. Four (4) GB of online storage accommodate many months of video data and processed results. The Coastal Center uses a Compact Disk - Write Once Read Many (CD-WORM) back-up unit so that data can be cycled onto CD media for permanent storage and distribution.

SYSTEM SOFTWARE

There are many software products available to aid in automating and monitoring a RVM system. A bulletin board system (BBS) program is an ideal method of controlling and automating a field station. BBSs provide an interface for remote users connecting to the field station over telephone lines and for the central laboratory computer when it connects to retrieve video data. Waffle, the BBS used by the USGS, can schedule data acquisition, provide e-mail service, perform system accounting, provide file upload/download capabilities, and maintain logs of station usage. Waffle also provides an administrative environment to enable the system operator to manage system resources, limit system access, and create user accounts. The BBS also enables the system operator to reprogram the field station remotely to adapt to changing conditions and acquisition schemes.

Video image acquisition programs can be written using the image processing software that generally comes bundled with add-on image processors. The image acquisition software used by the USGS was originally developed at the CIL in the programming language C and draws upon a library of image processing functions that provide control over the Dipix Power Grabber. The program, called average.c (see Appendix A), allows 1 to 24000 video frames to be averaged together. During the frame average, the pixel values in each successive video frame are added together. At the end of the summation, these cumulative values are divided by the total number of frames acquired. Averaging 1 frame produces the equivalent of a snapshot, while averaging multiple frames creates the effect of a photographic time exposure (see Image 2b).

The images acquired by average.c must be named, compressed and then spooled by the remote computer for retrieval by the laboratory computer. These functions are controlled by a program named snapsend.pl (Appendix B) written in the public domain programming language Perl, which permits manipulation of text strings and system commands. Snapsend.pl first executes average.c, to which it supplies the command line arguments of which camera to use, the gain and offset levels, the number of frames to average, and a 7 digit encoded output filename representing the time of acquisition. The script then executes cjpeg4, a C program which applies the JPEG compression algorithm to the image files produced by the system. Finally, snapsend.pl spools the data for UUCP retrieval by the lab computer (see NETWORKING).

The field station PC is further equipped with a TurboC compiler, a Perl compiler, and a full screen editor. These tools enable the system operator to create and compile new programs remotely in order to adapt station operation to changing conditions and needs. The use of TurboC and Perl is a common thread among Argus stations allowing the exchange of software tools among the system operators.

The central laboratory computer is primarily responsible for data retrieval and processing. Data retrieval can be automated using UUCP (described in the next section) or performed interactively using the conventional X, Y, or Zmodem file transfer protocols. Once data are retrieved, they are placed in an online storage location for processing. At the USGS, this storage location is also the FTP directory allowing the Internet community immediate access to the data.

The FTP directory hierarchy is created dynamically when data are moved from the receiving directory. A Perl script called voyeur02.move_data (see APPENDIX C) decodes the 7 digit encoded image file names provided by snapsend.pl and creates more intuitive and descriptive file names. The script then parses the long-term storage directory and creates a new sub-directory, named according to the date and camera of acquisition, into which it moves the images (see DATA AVAILABILITY).

NETWORKING

The central computer and field station(s) can be viewed as a wide area network connected using serial lines, modems, the telephone service, and the Unix feature UUCP. UUCP, which stands for Unix to Unix CoPy, is an ensemble of programs that create a dial-up network between machines. When one machine on the network requires data or services from another machine, it makes a UUCP request for those data or services. The UUCP request is then processed when connections are established between those machines via the telephone service. UUCP requests, such as to copy image data from the field station to the lab computer or to send e-mail, are processed during those connections. This process forms the basis for automated data retrieval from the field station. Though a full explanation of configuring UUCP is beyond the scope of this report, a detailed description can be found in O'Reilly and Associate's <u>Managing UUCP and Usenet</u> and in the Waite Groups <u>UNIX Communications</u>. To put it simply, the field station is described to the lab computer in terms of a telephone number, baud rate, login name, and password. The lab computer is then instructed to dial that number at specified times, log into the field station, and access any UUCP requests waiting to be processed. Though traditionally a UNIX feature, UUCP capabilities can be installed on a field station PC and may be available as part of the BBS package. When a field station acquires images, the acquisition programs make a UUCP request to copy the images to the central computer. The UUCP program on the field station PC places the images in a special spooling directory where they await transfer during the next connection made by the UUCP software on the lab computer. UUCP provides e-mail service in the same fashion. When e-mail is sent from a user or process on one machine to a recipient on another, the message is spooled until the next connection is made when it will be transferred and routed to its final destination. In this fashion, logs of station usage can be e-mailed to the system operator. Connections between lab computers and field stations typically occur early in the morning to take advantage of lower long distance telephone rates. Other UUCP features, such as 'tip' and 'cu' allow users of the central computer to quickly connect with the field station for an interactive session. There is virtually no limit to the number of remote field stations that can be UUCP-linked to the RVM network.

DEPLOYMENT

Once an area of interest has been selected, an appropriate location for the field station must be chosen. Generally, it is advantageous to place the camera as high over the area of interest as possible to increase the accuracy of image quantification algorithms. The location should have a clear unobstructed view of the area and, if being deployed outdoors, flat space large enough for the field station housings. Power and telephone connections are another concern. Direct power from a local utility company is more reliable than solar power. Likewise, telephone land lines are more reliable than cellular for data transfer.

The area of interest must be large enough to accomodate 2 or more ground control points (GCPs). GCPs can be any permanent, stationary feature, such as a building, landform, or installed marker, and are a necessary component of a RVM system. When attempting to quantify video images, GCPs provide reliable, permanent image and ground coordinate markers.

The RVM field station voyeur02 was deployed by the USGS Coastal Center on the southwestern shore of Lake Erie at Painesville, OH on August 17, 1993. Of primary concern was the severe erosion that had resulted in the destruction of many homes along the stretch of coast and threatened many more. After speaking with local home owners to get their approval and cooperation, a location on the edge of a bluff overlooking the lake was selected.

The suburban location made it a simple matter for local utility companies to install power and telephone connections. The site also permitted the cameras to be mounted in a position overlooking the area of interest from a vantage point on top of the bluff. Additionally, the area of interest gathered by the video cameras was large enough that three large, round, white signs could be installed within the camera view so serve as ground control points. The white color ensures their visibility under most conditions and the round shape enables their exact center to be accurately surveyed in the real world and precisely located within the image for accuracy in the geometry correction algorithms.

The diverse weather conditions of the area combined with the well populated, suburban setting required that Argus station be contained in rugged, secure housings. The main housing is a liquid tight, lockable, aluminum alloy box measuring 3' by 4' by 1.3' and is secured to a concrete form laid into the earth. A 1/4" aluminum shield fixed over and around the housing serves as a sun shield and provides additional protection from tampering (Image 4). The housing contains the PC, monitor, modem, power conditioner, and some tools used during site visits. Air vents and directional fans cool the system in the hot summer months. During the winter, the vents are sealed, the fans turned off, and heat is generated by the electronic components. The video cameras are contained in an environmental camera housing measuring 1.5' by 6" by 8" mounted on a pole overlooking the area of interest (AOI) (see Image 5). This commercially available housing features a thermostat controlled heater, cooling fans, and a window defroster allowing it to operate in extreme environmental conditions. Power, telephone, and video cables run through liquid tight conduit and fittings.

The Argus stations currently deployed are Yaquina Head, OR (Images 2a, 2b, 3a, 3b), Painesville, OH (Image 5), Duck, NC (Image 6), La Jolla Shores Beach, CA (Image 7), and Waimea Bay, HI (Image 8).

AREA SURVEY

A precise survey of the area of interest must be performed before any image quantification can be performed. Subsequent surveys are also a good idea since the area may be geologically dynamic. A survey is necessary to determine the angles and distances between the cameras and the GCPs. These data are then used in the geometry correction algorithms. Precision in the survey is extremely important to ensure the accuracy of the quantification algorithms. Sub-centimeter accuracy is required. GCPs should be surveyed to their exact center and cameras to their focal point. Benchmarks should be installed near the site for use in future surveys. Subsequent surveys of the area must be performed each time the cameras or GCPs are moved and should be made in the event of extreme change in the topography of the area of interest. Periodic surveys are also helpful to check the integrity of the data produced by the quantification programs.

DATA AVAILABILITY AND DISTRIBUTION

Properly configured, a RVM system can make its data and services available to the user community in ways that do not threaten the security of the system. When not acquiring data, the field station voyeur02 will accept calls from remote users. After connecting to the field station using a computer and modem, and entering a valid login name and password provided by the system operator, a user can acquire and download video data, learn more about the station and its usage, and send e-mail to recipients both on the field station as well as the Internet. The requirements are a computer or terminal, a modem capable of running between 2400 and 19200 baud, and some type of terminal program. All connections to the field station and the actions of the users are logged and reported to the system operator.

The phone number for the Argus station voyeur02 on Lake Erie is (216)350-9126. After dialing the number, wait about 30 seconds while the station's modem receives the call and adjusts to the correct baud rate. Next, the station will issue a login prompt. Login with the name "guest" and the password "video". At this point, a banner message will appear followed by a command line prompt. Help is available on a variety of subjects by typing in the command "help". To acquire an image, first read the help available on the "average" program by typing "help average". To execute the average program type, for example, "average -c 1 -f 1 snapshot" on the command line. This will instruct the image processing board to acquire one frame from camera #1 and write the image to disk in a file named "snapshot", which will reside in the current working directory. The image will be in SunRaster format and should be JPEG compressed before any file transfers take place. Do this by typing the command "compress snapshot snapshot.jpg". To download a file to your local machine, enter the file download/upload area of the system by typing in the command "files". In this mode, all commands are recognized by the first letter of the command. First, "L"og to the directory in which the compressed image resides. Next select the "P"rotocol to use - "Z"Modem. Finally "S"end the file to initiate the ZModem download. Transfer times vary based on baud rates and line conditions. "Q"uit will exit the "files" mode. The command "bye" will log the user out of the station and break the telephone connection.

If the central laboratory computer has a real-time Internet connection, the Internet community can access video data via the ARPANET File Transfer Protocol (FTP). This is the most efficient way to access video data. FTP software has been written for virtually every computer platform and is freely available in the public domain.

A typical FTP session to access video data from the USGS laboratory computer, rocky, would begin by executing FTP with the command "ftp rocky.er.usgs.gov" or "ftp 131.247.143.106." Log in with username "anonymous" and use a valid Internet or e-mail address as the password. Once the FTP session is started, users should change directory (cd) to the /pub directory, which contains subdirectories for each of the field stations deployed by the Center, with the command "cd pub". Help is available by tying in a question mark "?" and banner messages will guide the user as they peruse the file system.

Image files are placed in a directory structure organized according to the date and camera of acquisition. For example, images acquired by camera #1 of the RVM field station voyeur02, on November 14, 1993 would be stored in the subdirectory "/pub/voyeur02/93/318_Nov.14/cam_1". The 318 preceeding the date is the Julian day of acquisition. Similarly, images acquired on the same day, but from camera #2, would be in the subdirectory "/pub/voyeur02/93/318_Nov.14/cam_2".

In each of the above examples, the subdirectories would contain image files named according to the following convention:

UNIXTIME.WDAY.MON.MDAY_TIME.ZONE.YEAR.TYPE.CAM.FORMAT

UNIXTIME, consisting of 9 digits, is a UNIX method of referring to time as the number of non-leap seconds since January 1, 1970, the generally accepted inception date of the UNIX operating system. Decoded, this number yields the date and time fields of the image file name. TYPE represents whether the image is a snapshot (snap) or time exposure (timex). CAM symbolizes the camera number used to acquire the image. FORMAT is a 3 character extension labeling the type of image file format used.

eg. 753282056.Sun.Nov.14_13:00:56.GMT.1993.snap.1.jpg

The above example is the filename of an image acquired at 13:00:56 Greenwich Mean Time on Sunday, November 14, 1993. The image is a snapshot acquired from camera #1 and is in JPEG image file format.

To download an image first change directory to the file structure location for the desired field station, date and camera. Next, type "get filename" where filename is the name of the desired image file. For example, to download the image in the above example, first type "cd /pub/voyeur02/93/318_Nov.14/cam_1" to move into the proper directory. Then type "get 753282056.Sun.Nov.14_13:00:56.GMT.1993.snap.1.jpg". The command "bye" or "quit" will end the FTP session. All FTP sessions are logged and reported to the system operator daily.

OSU maintains a database of image data from all Argus stations presently deployed. To access this archive, ftp ruby.oce.orst.edu (128.193.64.54) with login name "anonymous" and a valid e-mail address as a password. In the /pub directory are sub-directories called LakeErieArgus, DuckArgus, etc. Within these directories, the directory structure follows the same conventions as described above.

VIEWING RVM IMAGERY

Imagery downloaded from either the field station(s) or the FTP sites is in JPEG image file format. This is currently the industry standard compression algorithm and many public domain, shareware, and commercial image viewers for all computer platforms will readily decompress and display JPEG images. Example shareware programs are LView for PCs running Microsoft Windows, JPEGView for Macintosh computers, and xv for Unix workstations using X11 based interfaces (OpenLook, Motif, etc).

DATA PROCESSING

Image processing tasks will vary with the areas and processes being observed. There is a large base of software available to lay a foundation for most image processing tasks. Processing of the images returned by voyeur02 begins by converting the images to SunRaster image file format. Though images on the field station are JPEG compressed to reduce storage space and increase data transfer rates, the format is inappropriate for image processing. SunRaster's simple, uncompressed image file format readily lends itself to image processing and is quite suitable for scientific analysis. All data quantification and image processing tools developed for the RVM system are written specifically around the SunRaster image file format.

The next processing step is to produce a solution file for the image geometry using a software tool such as geomtool, which was developed at OSU's Coastal Imaging Lab. Geomtool is a graphical software tool that allows the user to interactively select the image locations of GCPs. Geomtool then solves for the image geometry using the topographical locations of the GCPs and produces a least squares solution for photogrammetric coordinate transformation (Holman *et al*, 1993).

Ideally, the cameras should be secured in a position that will not fluctuate. However, voyeur02's camera positions are not constant due to buffeting winds, human interaction during site visits, and the occaisional relocation of the camera housing. These events determine how frequently a geometry solution needs to be produced. Generally, a geometry solution will have to be produced for each week's worth of data though there have been cases were a unique solution was required for each image acquired in a single day.

Geomtool produces a geometry solution which allows U,V image coordinates (U and V are used in place of X and Y to avoid confusion) to be transformed to x,y,z ground coordinates provided that the third dimension, z, is already known. For example, when transforming shoreline coordinates, the z value can be assumed to be lake level. This dimensional coordinate is obtained from water level data provided by the Great Lakes Water Level Database maintained by NOAA. Once the geometry solution is known, the location or size of features within the image can be computed from their U,V image coordinates.

A regular processing task for voyeur02's images is the digitization of the shoreline. This is achieved by running a primitive edge detector on the images to determine the point where the water and land meet in time exposure images (frame averages) provided by camera #2. Camera #2 is used because its wider angle lens captures the entire shoreline of the area of interest. Time exposures are desirable since they offer a more reliable "average shoreline location" than their snapshot counterparts and also limit the pixel values representing the water to a predictable range. This range is the key to determining shoreline location. For each row of the image, working from left to right, or from the pixels representing water towards those representing the shore, each pixel is inspected for inclusion in a range of pixel values encompassing the predicted pixel values for "average water brightness". When a pixel is encountered that is not in this range it is assumed that it represents a feature other than water and therefore is the location of the water's edge. The pixel's (x,y) image location is then highlighted in the image and recorded in an ASCII file. Since edge detection is not 100% accurate, visual inspection of the images and occasional editing of the ASCII image coordinates file is required.

Another processing task performed on the images is the digitization of the ice edge, as it grows out from the shoreline during the winter months, in order to determine growth rates, ice volume, and ice stability. A slightly different edge detecting algorithm than the one to locate shorelines aids in determining where the water and ice edge meet. Again, frame averages from camera #2 are used because camera #2 offers a view of the entire shoreline and because the frame averages limit the range of pixel values representing water. The difference between the pixel intensities of the white ice and the underlying brown water is the key to edge detection. For each row of the image, starting with the pixels representing water and working towards the shore, each pixel value is subtracted from the value of the pixel adjacent to it in the row. If the difference between the 2 pixels is greater than a defined threshold representing the predicted difference between the pixel values representing white ice and those representing average water color it is assumed that the edge of the ice has been found. The pixel's (x,y) location is then highlighted in the image and recorded in an ASCII file. Again, visual inspection of the highlighted edges and editing of the ASCII coordinates file is often required to ensure proper edge detection.

The image coordinates determined by the edge detecting programs can then be transformed to ground coordinates using the geometry solutions produced by geomtool. This essentially allows continuous digital surveys of the area without the need to physically visit the site.

CONCLUSION

Remote Video Monitoring systems are a cost effective means of collecting daily video records of truly remote areas. Video data has proven to be useful in a variety of coastal studies and offers many advantages over conventional sensing methods. RVM systems provide automatic data acquisition, collection, processing, and archiving. Administration of the systems consists mainly of monitoring daily system-generated status messages. The systems also provide several avenues of distribution for both raw and processed video data and extend their services to the user community. Agencies wishing to take advantage of Remote Video Monitoring systems can benefit from over a decade of research and development which has resulted in a wholly complete and reliable data system.

* (References to specific products are made for identification purposes only and do not indicate an endorsement on the part of the U.S. Geological Survey).

REFERENCES

Holland, K.T. and Holman, Rob .A., 1993, The statistical distribution of swash maxima on natural beaches. *Journal of Geophysical Research*, 98(C6), 10371-10278.

Holman, Rob A., and Guza, R.T., 1984, Measuring run-up on a natural beach. *Coastal Engineering*, 8, 129-140.

Holman, Rob A., Sallenger Jr., Asbury H., Lippmann, Tom C., Haines, John W., 1993, The application of video image processing to the study of nearshore processes. *Oceanography*, v.6, n.3, p. 78-85.

Lippmann, Tom C., and Holman, Rob A., 1989, Quantification of sand bar morphology: a video technique based on wave dissapation. *Journal of Geophysical Research*, 94(C1), 995-1011.

Lippmann, Tom C., and Holman, Rob A., 1991, Phase speed and angle of breaking waves measured with video techniques. In: *Coastal Sediments*, '91, N. Kraus, ed., ASCE New York, 542-556.

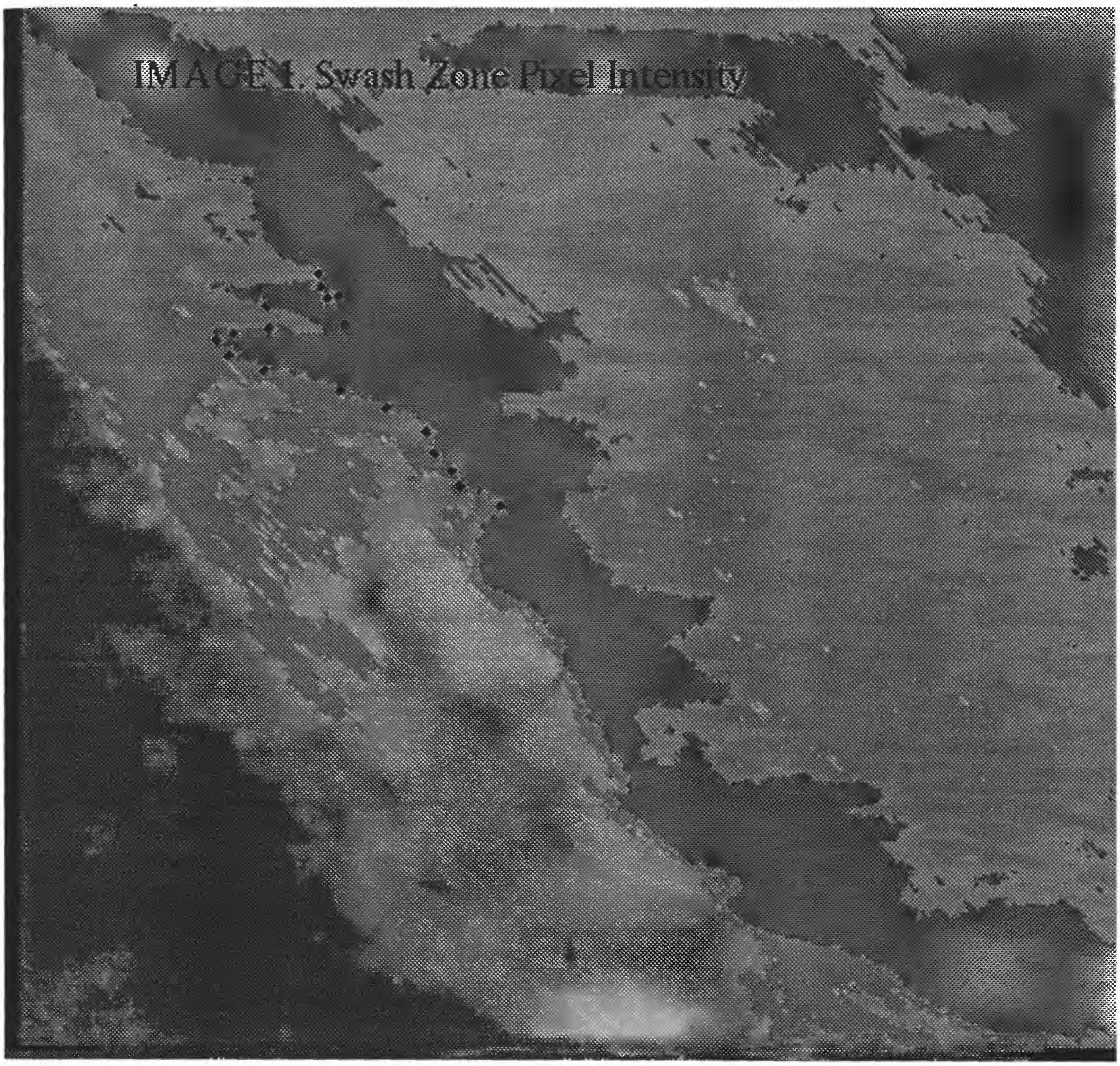


IMAGE 1. Swash Zone Pixel Intensity

IMAGE 2a. Yaquina Head, Oregon Snapshot

IMAGE 2b. Yaquina Head, Oregon Time Exposure

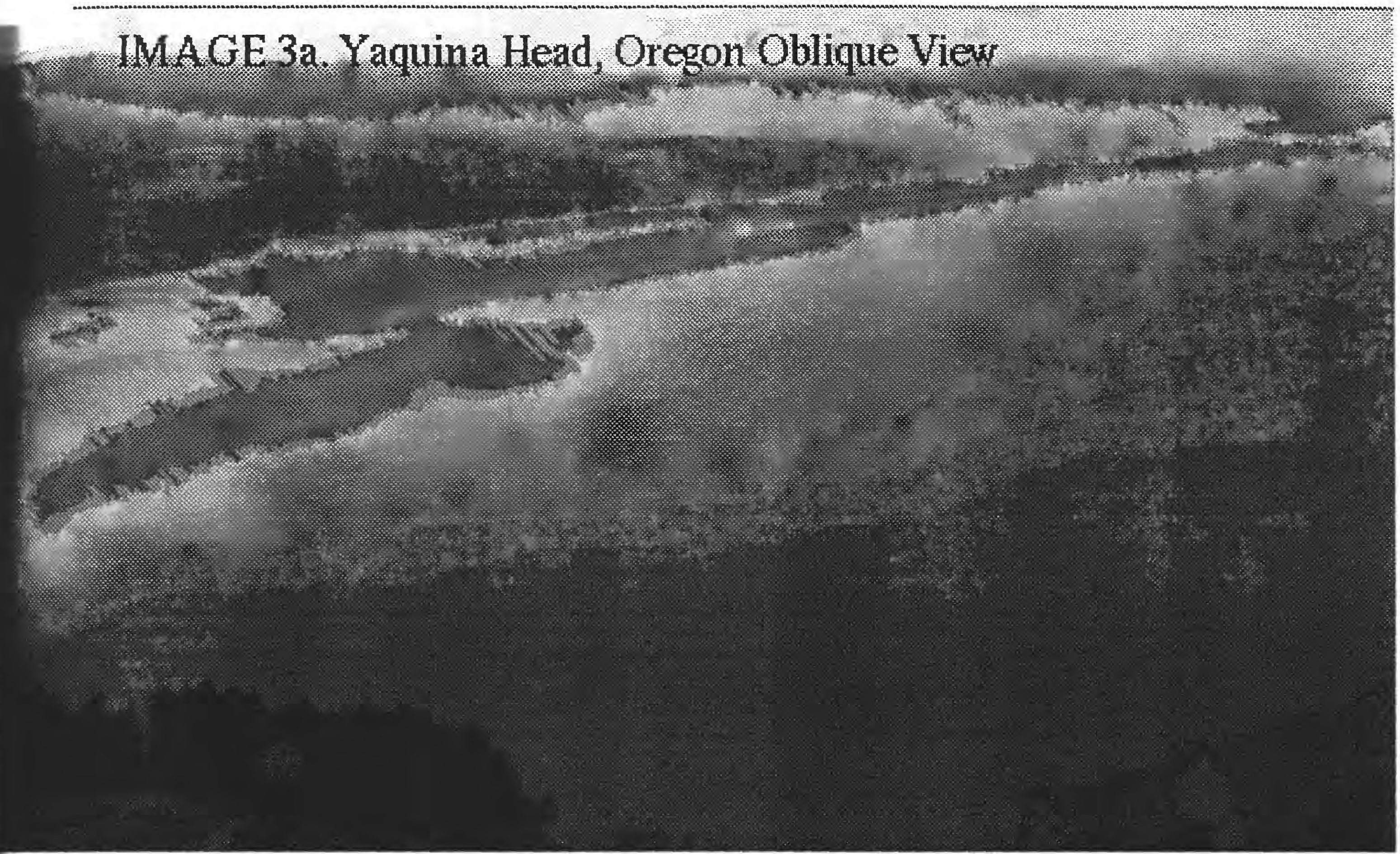


IMAGE 3a. Yaquina Head, Oregon Oblique View

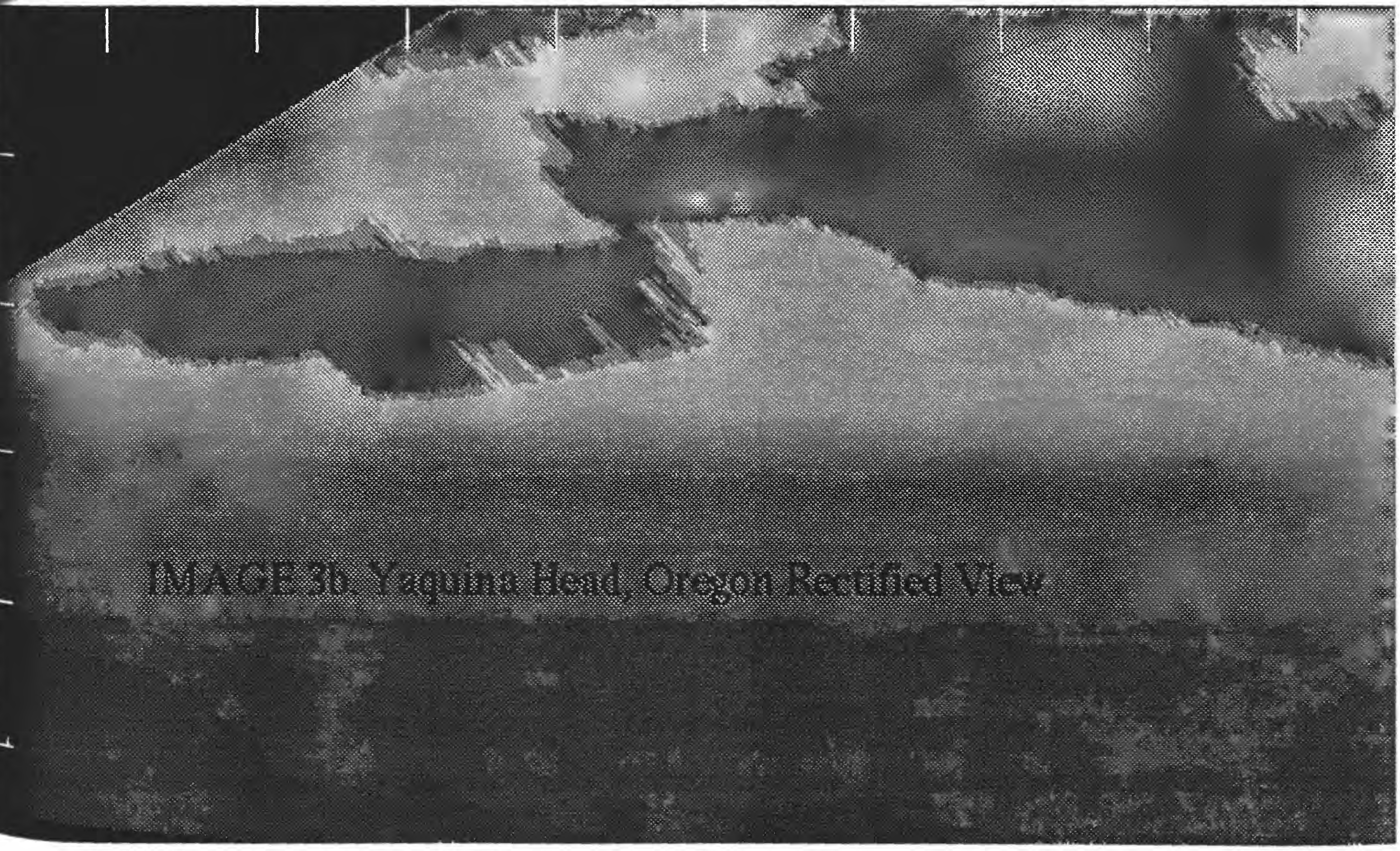


IMAGE 3b. Yaquina Head, Oregon Rectified View

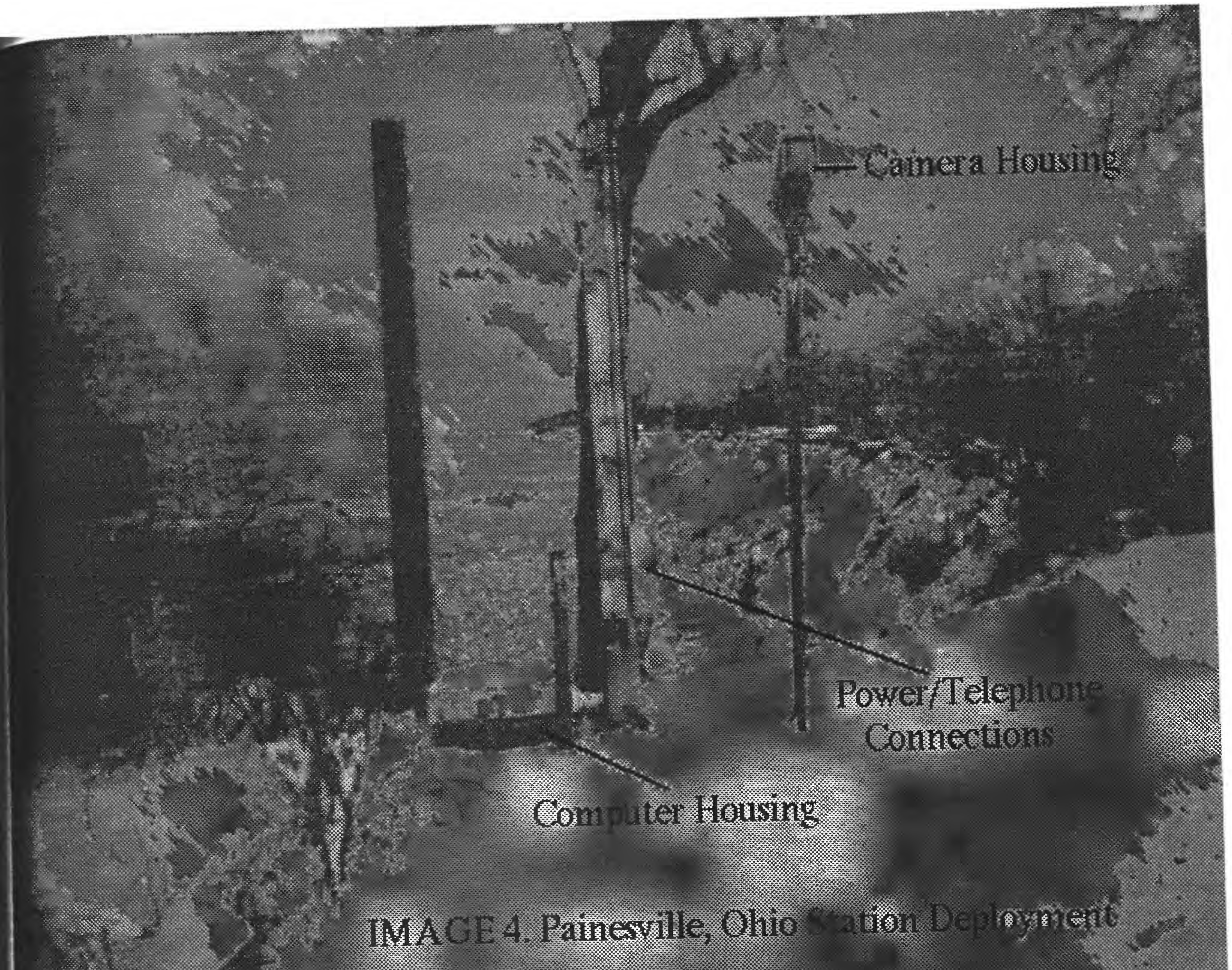
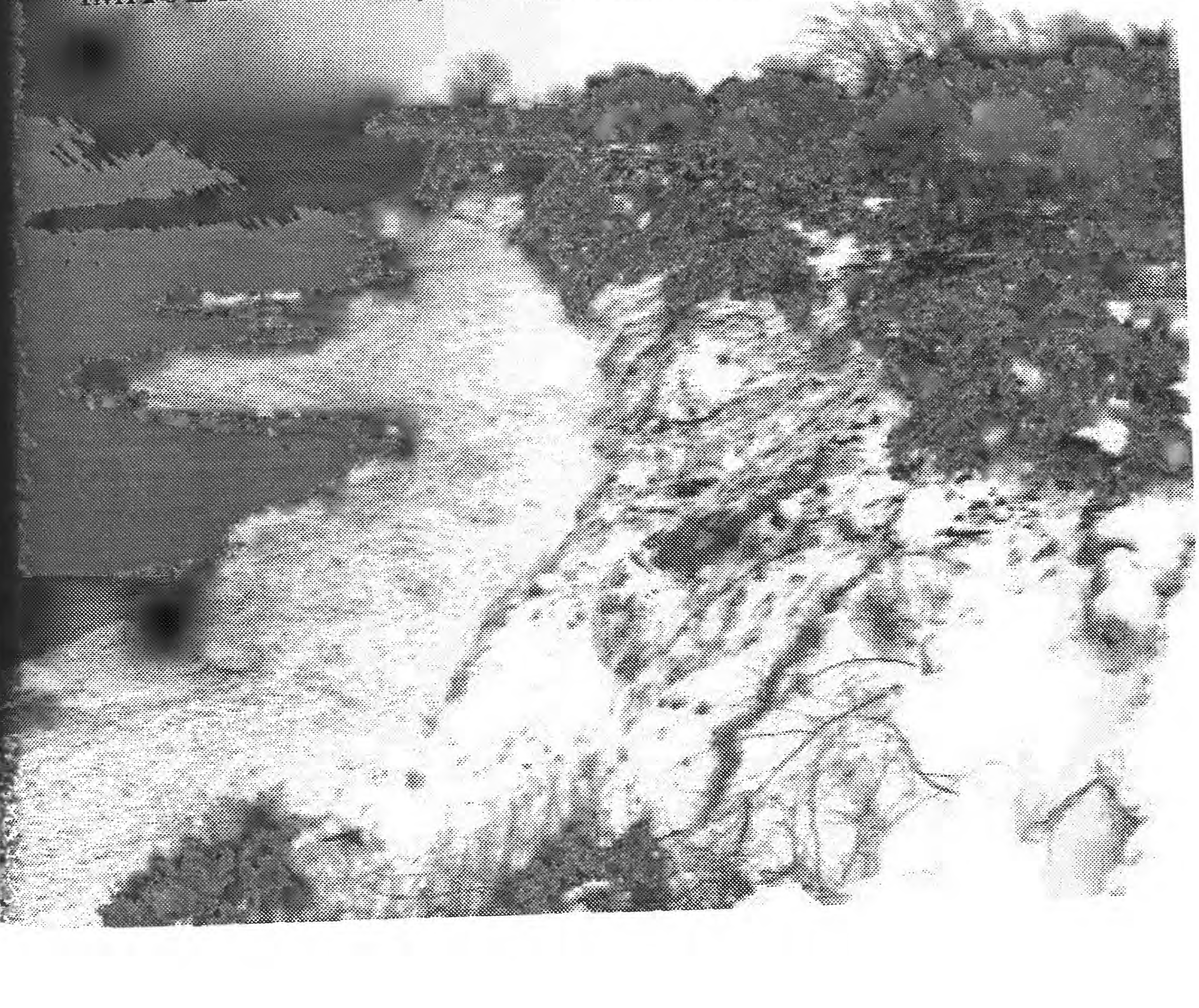


IMAGE 4. Painesville, Ohio Station Deployment

IMAGE 5. Painesville, Ohio Area of Interest



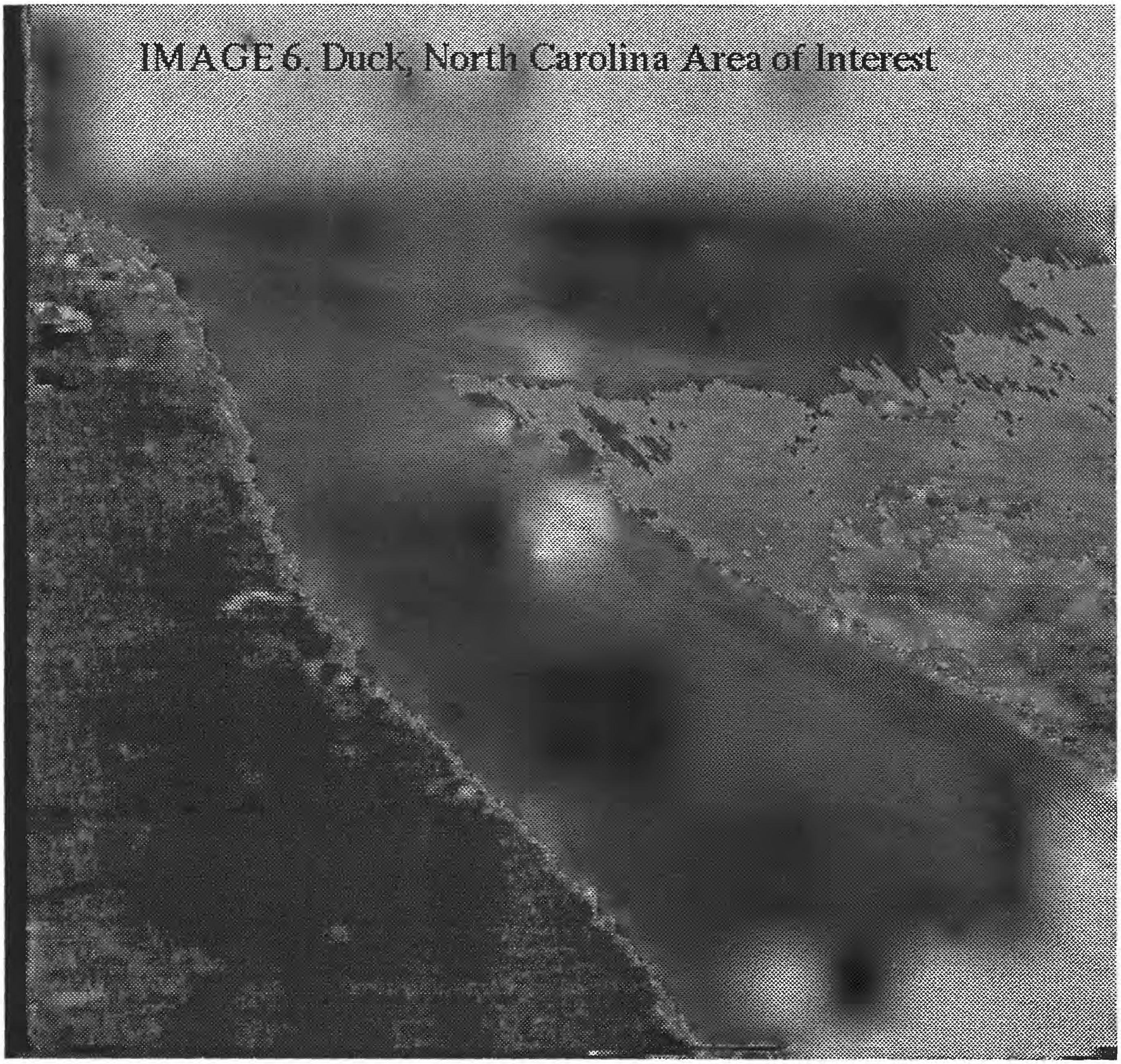


IMAGE 6. Duck, North Carolina Area of Interest

pps picture taken: Sun Jul 2 10:02:56 PDT 1995 Average: -f 2400 -c 0
ipps picture taken: Sun Jul 2 10:02:56 PDT 1995 Average: -f 2400 -c 0

IMAGE 7. La Jolla Shores, California Area of Interest

Jul 03 11:14:30 1995 HST10HDT F: 804802465.dip C: 2 S: 2400 G: 0 O: 42

IMAGE 8. Waimea Bay, Hawaii Area of Interest

APPENDIX A. Frame Averaging C Code

```
/* Average.c is a frame averaging program to average X number of frames
and store the resulting file to disk. Version 1.1 was produced by John
Stanley of OSU. Version 1.2 is a variation produced by Bill Townsley
of USGS to allow for multiple cameras. */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "stddefs.h"
#include "ptype.h"

#define TAGFILE "C:\\P360F\\A.TAG"
#define FN_LEN 80
#define HORIZ 640
#define VERT 480
#define CAM_PORTS 4
#define VERSION "Frame Averaging Program Version 1.2 by Bill Townsley 2/25/93.\n"

/* Initialize struct with default gain, offset and framecount. */
/* No filename and bad camera number given for error checking */
/* since they must be specified by the caller anyway. */
struct arg_struct {
        char outfile[FN_LEN];
        int gain;
        int offset;
        int framecount;
        int camera;
} args = {"",38,48,2400,9};

time_t now;

/*******************************************************
usage is called from parse_args and check_args and gives
information on how to use the program.
*******************************************************/
void usage()
{
        fprintf(stdout, "\nUsage: average [-g gain] [-o offset] ");
        fprintf(stdout, "[-f framecount] [-c 1..2] filename\n");
        fprintf(stdout, "\n -default switches are '-g 38 -o 48 -f 2400'\n");
        fprintf(stdout, " -camera and filename must be specified\n");
        exit(1);
}

/************************************************************
parse_args is called from main to parse the command line
argument list and acquire images accordingly.
************************************************************/
void parse_args(argc, argv)
        int  argc;
        char  *argv[];
{
        int  i = 1;

        while (i < argc) {
                if (*argv[i] == '-') {
                        switch(*(argv[i] + 1)) {
                                case 'o':
                                        i++;
                                        args.offset = atoi(argv[i++]);
                                        break;
                                case 'g':
                                        i++;
                                        args.gain = atoi(argv[i++]);
                                        break;
                                case 'f':
                                        i++;
                                        args.framecount = atoi(argv[i++]);
                                        break;
                                case 'c':
                                        i++;
                                        args.camera = atoi(argv[i++]);
                                        break;
                                default:
                                        usage();
                        }
                } else
                                strncpy(args.outfile, argv[i++], FN_LEN);
        }
        return;
}

/************************************************************
```

```
check_args is called from main and checks to see that the
command line arguments are acceptable.
**********************************************************/
void check_args()
{
	int i, name_length;

	if (fopen(args.outfile, "r") != NULL) {
		fprintf(stdout, "\nFile(s) already exists: %s\n", args.outfile);
		fprintf(stdout, "Filenames must be unique.\n");
		usage();
	}
	if ((args.camera != 1) && (args.camera != 2)) {
		fprintf(stdout, "\nBad or missing camera value.\n");
		usage();
	}
	if (strlen(args.outfile) < 1) {
		fprintf(stdout, "\nMust specify output filename\n");
		usage();
	}
	if (args.offset < 10 || args.offset > 63) {
		fprintf(stdout, "\nBad offset value: %d\n", args.offset);
		usage();
	}
	if (args.gain < 10 || args.gain > 63) {
		fprintf(stdout, "\nBad gain value: %d\n", args.gain);
		usage();
	}
	if (args.framecount <= 0 || args.framecount > 24000) {
		fprintf(stdout, "\nBad framecount value: %d\n", args.framecount);
		usage();
	}
	return;
}

/**********************************************************
load_init_dsp is called from acquire_images to load digital signal
processing (DSP) code and initialize DSP memory.
**********************************************************/
void load_init_dsp()
{
	init_dram();
	load_tag_file(TAGFILE);
```

```
        run_dsp();
        init_cbinding();
        sleep(1);
        return;
}

/**********************************************************
acquire_images is called from main and acquires images
according to the command line arguments.
**********************************************************/
void acquire_images()
{
        char    start[80], end[80];

        load_init_dsp();
        fprintf(stdout, "\n%s\n", VERSION);
        fprintf(stdout, "Output: %s\n", args.outfile);
        fprintf(stdout, "Gain: %d, Offset: %d, Frame: %d, Camera: %d\n", \
        args.gain, args.offset, args.framecount, args.camera);
        fprintf(stdout, "Average will take approx %d seconds\n", \
        (int)(args.framecount/3.3));

        set_gain_and_offset(args.gain, args.offset, TRUE);
        set_grab_size(HORIZ, VERT, TRUE);
        set_video_input(args.camera, TRUE);
        frame_grab(0, TRUE);

        time(&now);
        strcpy(start, ctime(&now));
        fprintf(stdout, "Starting average NOW!\n");
        faverage(args.framecount, 0, TRUE);
        time(&now);
        strcpy(end, ctime(&now));
        ras_image_buffer_to_disk(0, args.outfile);
        fprintf(stdout, "Started at: %s\n", start);
        fprintf(stdout, "Ended at:   %s\n", end);

        return;
}

void main(argc, argv)
        int         argc;
        char        *argv[];
```

```
{
        if (argc < 4)
                usage();
        parse_args(argc, argv);
        check_args();
        acquire_images();
        exit(0);

 }
```

APPENDIX B. Image Acquisition Perl Script

```
# snap a picture, UUCP it to the central lab computer

$now = time;
system( "average -c 1 -f 1 -g 34 -o 50 $now.snp" );
system( "cjpeg4 -quality 80 $now.snp $now.jpg" );
system( "uucp -c $now.jpg wayback!~/receive/townsley/voyeur02/cam_2/$now.snap.2.jpg" );
unlink "$now.snp";
unlink "$now.jpg";

$now = time;
system( "average -c 1 -f 600 -g 34 -o 50 $now.tmx" );
system( "cjpeg4 -quality 80 $now.tmx $now.jpg" );
system( "uucp -c $now.jpg wayback!~/receive/townsley/voyeur02/cam_2/$now.timex.2.jpg" );
unlink "$now.tmx";
unlink "$now.jpg";

$now = time;
system( "average -c 2 -f 1 -g 35 -o 48 $now.snp" );
system( "cjpeg4 -quality 80 $now.snp $now.jpg" );
system( "uucp -c $now.jpg wayback!~/receive/townsley/voyeur02/cam_1/$now.snap.1.jpg" );
unlink "$now.snp";
unlink "$now.jpg";

$now = time;
system( "average -c 2 -f 600 -g 35 -o 48 $now.tmx" );
system( "cjpeg4 -quality 80 $now.tmx $now.jpg" );
system( "uucp -c $now.jpg wayback!~/receive/townsley/voyeur02/cam_1/$now.timex.1.jpg" );
unlink "$now.tmx";
unlink "$now.jpg";
```

APPENDIX C. Data Storage Perl Script

```
#!/net/rocky/roc1/local/bin/perl
###############################################################################
# This script moves RVM video data from staging directories on wayback to    #
# the video archive directory structure on rocky.                             #
#                                                                             #
#    USAGE: rvm_mv_data.pl [station_name]                                     #
#                                                                             #
# The default is for all data from all cameras from all stations to be       #
# moved.  There is no harm if no data for any station is in the staging      #
# area.  Results of the creation of new archived sub-directories and the     #
# movement of files is mailed to the SYSOP.                                   #
#                                                                             #
# The text file rvm_data is REQUIRED for this script to work.  Its           #
# location is defined below.                                                  #
###############################################################################

# required to decode time
require "ctime.pl";

# some boolean values
$TRUE = 0;
$FALSE = 1;

# a boolean variable
$FOUND = $FALSE;

# location of mail utility
$MAIL = "/usr/ucb/mail";

# mail alias for the sysop
$SYSOP = "townsley";

# arrays of daily monthly data
@DoW = ('Sun','Mon','Tue','Wed','Thu','Fri','Sat');
@MoY = ('Jan','Feb','Mar','Apr','May','Jun',
      'Jul','Aug','Sep','Oct','Nov','Dec');

# location of the rvm data file
$RVM_FILE = "/net/rocky/roc1/home/townsley/rvm_scripts/rvm_data";

# check for argument
```

```
if($#ARGV > 0) {
 print "\nUSAGE: rvm_mv_data.pl [station_name]\n";
 print "  Function: moves data from staging area to storage area\n";
 print "  Default: move data from ALL field stations\n\n";
 exit(0);}

# open the rvm data base file
open( RVM_FILE, $RVM_FILE ) || die "Can't open/find $RVM_FILE\n";

############################################################
# for each station described in the rvm_data file OR          #
# for the single station listed on the command line.          #
############################################################
LOOP1: while(($rvm_data = <RVM_FILE>) && ($FOUND == $FALSE)) {

        # get rid of the newline
        chop($rvm_data);

 # a counter to specify camera
 $counter=1;

 # break out station name, location, login name, password
 # telephone number, number of cameras, source directory, and
 # destination directory from current rvm data file line.
 # skip any comments
 if($rvm_data =~ /^#/) {
  next LOOP1;}
 else {
  ($Name,$Loc,$Login,$Passwd,$Number,$Cameras,$Src,$Dst) =
  split(/,/,$rvm_data);}

 # if a single station was indicated check to see
 # if we've found it, if not move to next line
 if($#ARGV == 0) {
  if($Name eq $ARGV[$#ARGV]) {
   $FOUND = $TRUE;}
  else {
   next LOOP1;}
 }

 # open a mail for for logging to the SYSOP
 open(MAIL,"| $MAIL -s $Name.data.files $SYSOP");
```

```
######################################
# for each camera the station has..  #
######################################
LOOP2: while($counter <= $Cameras) {


 # create the source camera pathname
 $src_cam_dir=sprintf("%s/cam_%01d",$Src,$counter);

 # open and read the source camera directory
 opendir(SRC_CAM_DIR,$src_cam_dir);
 @camfiles = grep(!/^\.\.?$/,readdir(SRC_CAM_DIR));
 closedir(SRC_CAM_DIR);

 # indicate which station/camera we're on
 print MAIL "\n--------$Name Camera #$counter file transfer log--------\n\n";

 ###############################################
 # for every source file in the camera directory             #
 ###############################################
 LOOP3: while($_ = shift @camfiles) {

  # reset the time variable
  $time=0;

  # if the filename begins with 9 digits, extract the 9
  # digit time variable and the image type from the filename
  # else move on to next file
  if(/^(\d\d\d\d\d\d\d\d\d)/) {
   ($time, $type)=split(/\./,$_,2);}
  else {
   next LOOP3;}

  # break out time components from the 9 digit time code
  ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)=gmtime($time);

  # convert yday into Julian day
  $yday++;

  # create/open the base destination directory
  $dst_base_dir=sprintf("%s/%02d/%03d_%s.%02d",
     $Dst,$year,$yday,$MoY[$mon],$mday);
  if(!opendir(DST_BASE_DIR, $dst_base_dir)) {
      system("mkdir $dst_base_dir");
      system("chmod 755 $dst_base_dir"); }
    close(DST_BASE_DIR);

    # create/open sub directories for the current camera
    # and log results
    $dst_cam_dir = sprintf("%s/cam_%01d",$dst_base_dir,$counter);
    if(!opendir(DST_CAM_DIR, $dst_cam_dir)) {
      system("mkdir $dst_cam_dir");
      system("chmod 755 $dst_cam_dir");
      print MAIL "\ncreated $dst_cam_dir\n"; }
    close(DST_CAM_DIR);

    # create source and destination filenames,
    # copy the file to the destination directory,
    # test that it actually was relocated
    # then delete the original file
    $src_cam=sprintf("%s/%s",$src_cam_dir,$_);
    $dst_cam=sprintf("%s/%s.%s.%s.%02d_%02d:%02d:%02d.GMT.19%02d.%s",
        $dst_cam_dir,$time,$DoW[$wday],$MoY[$mon],$mday,$hour,$min,
        $sec,$year,$type);
    system("cp $src_cam $dst_cam");
    system("chmod 644 $dst_cam");
    if(-e $dst_cam) {
      unlink $src_cam;
      print MAIL "\n$time.$type to $dst_cam\n";}
    else {
      print MAIL "\nERROR $src_cam not moved\n";}

   } # END OF LOOP3

   # increment the camera counter variable
   $counter++;

  } #END OF LOOP2

  # close things down
  close $RVM_FILE;
  close MAIL;

} # END OF LOOP1

# if a single station was indicated and
# never found give an error message.
if(($#ARGV == 0) && ($FOUND != $TRUE)) {
print "\n$ARGV[$#ARGV] not valid selection\n";
print "  run rvm_stations.pl or check rvm_data file\n\n";}
```