Providing Captured Images and Video to REVEAL

Will Doutre

Brown University
wdoutre@cs.brown.edu

Abstract

The REVEAL project aims to provide Archeologists with advanced Computer Vision/Pattern-Recognition tools for reconstructing recovered archeological fragments into digital representations of their historical objects. Of the many components involved in the REVEAL project, I worked on the video acquisition system called C-REVEAL and the main user interface application, J-REVEAL. This report highlights my work on these two pieces of the REVEAL system.

1. – Introduction

REVEAL is a joint research project of Brown University's Division of Engineering, Laboratory for Man/Machine Systems (LEMS), Brown University's Joukowsky Institute for Archeology and the Ancient World, the Institute for the Visualization of History in Williamstown, MA, and archeologists at Tel Aviv University in Israel. These institutions are working together to develop new contributions to the computer vision/pattern-recognition (CVPR) research field while concurrently advancing the technology and capabilities of archeologists in the exploration and documentation of archeological sites of interest around the world.

The REVEAL project is comprised of four smaller research projects:

1) The development of an Internet accessible archaeological space-time excavation-site database, which contains data about archeological findings, images, video, and extracted 3D representations and geometry

2) The development of tools for 3D object, fragment, and large architectural chunk surface and geometry estimation from moving and stationary still and video cameras.

3) The development of tools for the automatic re-assembly of object fragments from the database into identifiable glass, ceramic, and other archeological objects or architectural structures.

4) The development of interactive and 3D immersion tools for the purpose of manipulating database objects (such as fragments, images, and architectural structures) to enable re-assembly and the taking of precise archeological measurements at the location of an archeological excavation site.

All four projects constitute research opportunities in CVPR and the development of these tools stands to benefit the discipline of archeology by granting "unprecedented access to and analysis of past human activity, with geometry providing the capability to recognize, visualize, and evaluate forms of material culture accurately, rapidly and (above all) non-destructively."[1] The four projects each address other potential applications in secondary areas such as homeland security, remote collaboration, education, and entertainment.

In the remainder of this report: section 2 offers a high-level overview of REVEAL's software components and my contributions to the overall system, sections 3 and 4 detail my contributions to the project, and section 5 outlines future work to be done, before section 6 concludes this report.

2. – REVEAL Overview

The REVEAL system is comprised of four main components, depicted in Figure 1:

1. D-REVEAL: The archeological space-time excavation-site database, which is used for storing all the archeological data.

2. C-REVEAL: Small computers that collect data from and configure digital still and video cameras and then send that data to D-REVEAL.

3. J-REVEAL: The user-interface that allows archeologists to query information, perform re-assemblies, and interact with 3D objects from the D-REVEAL database. Also the user-interface for managing the C-REVEAL computers' acquiring of image and video data.

4. Other data acquisition tools for capturing archeological object information and storing it in D-REVEAL.

My main contribution to the REVEAL project was the development of the C-REVEAL software system. C-REVEAL is tasked with the capturing, cataloging, and uploading of images and videos from the attached cameras to D-REVEAL. C-REVEAL is also responsible for receiving commands from J-REVEAL and the user for changing camera properties and settings, for returning image and video directly to the J-REVEAL user (if that data has not already been uploaded to D-REVEAL), and performing camera calibration. I also contributed to the development of the piece of J-REVEAL responsible for communication with C-REVEAL systems.

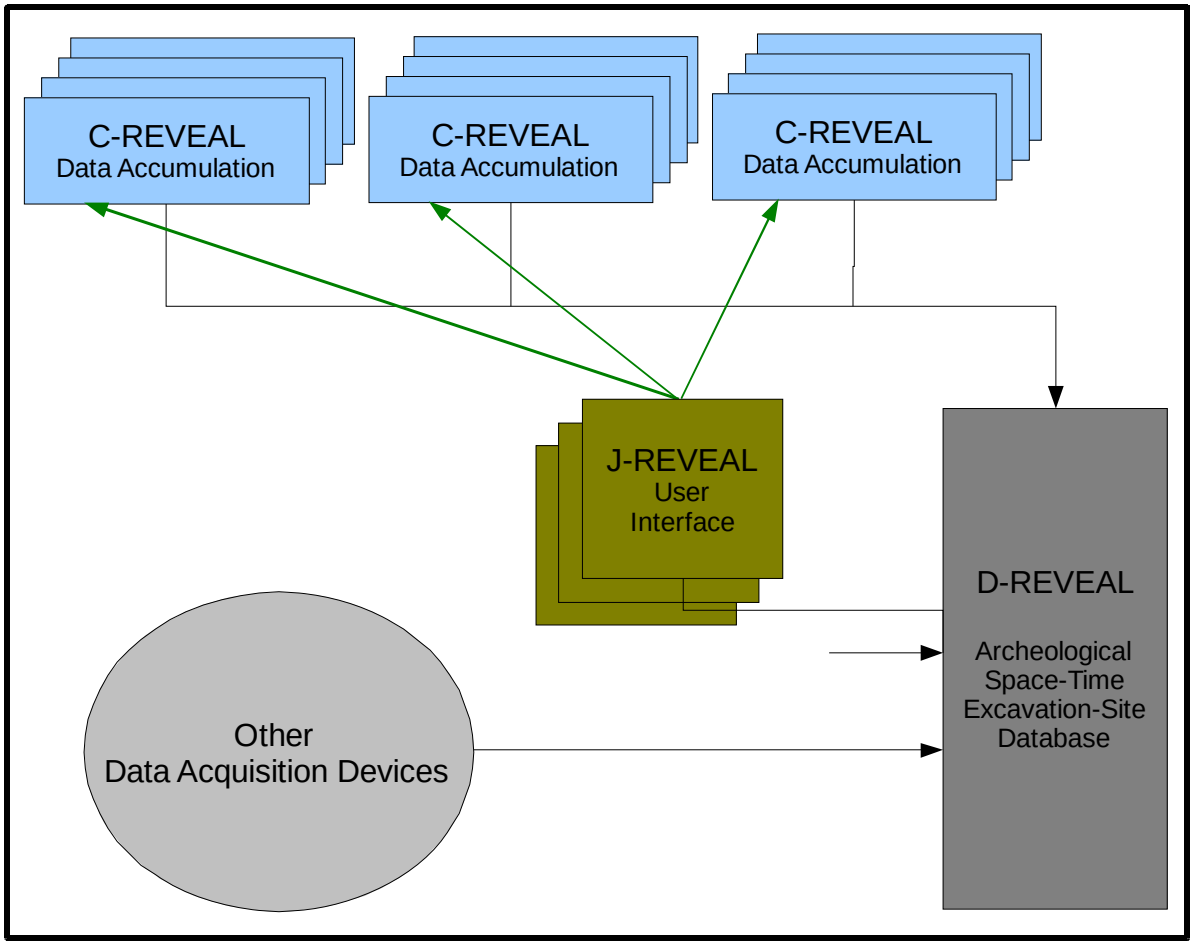

Figure 1 - Overview Diagram of REVEAL system components and interactions.

3. – C-REVEAL

The C-REVEAL unit consists of a small (12 x 11.6 x 5 cm) computer, a small network switch, two D-Link DCS900 IP video cameras, and one Olympus SP-500 Ultra Zoom six Megapixel digital camera for taking stills. The two D-Link IP cameras and one of the computer's Ethernet ports communicate via the network switch, while the computer talks to the still camera via USB 2.0. The computer's other Ethernet port is connected to a general switch granting access to D-REVEAL and the J-REVEAL computer systems. The choice to run the Windows XP operating system instead of an open source alternative was necessitated due to compatibility issues with the Olympus Developer SDK and USB drivers, which would only run on Windows XP. (An attempt was made to use Windows Vista, but the SDK and/or the USB driver was not compatible with that operating system.)

The C-REVEAL server software is written in the C++ programming language due to the memory constraints of the Fit-PC 1.0 computer (AMD Geode LX800 CPU, 256MB RAM, Dual 80 GB Hard Drives, Dual 10/100 Ethernet ports). Communication with the two D-Link IP cameras was via HTTP,

however other REVEAL devices issue commands to C-REVEAL using a TCP protocol, which I designed and implemented for this project.

Inside the C-REVEAL computer, a small server application called CapServer manages several worker threads to enable all the functionality of C-REVEAL, these threads can be grouped into four different categories of listening for client requests, handling those requests, continuous recording of camera video, and recorded video management. These process threads are depicted in Figure 2, and described in sections 3.1 through 3.4 below.

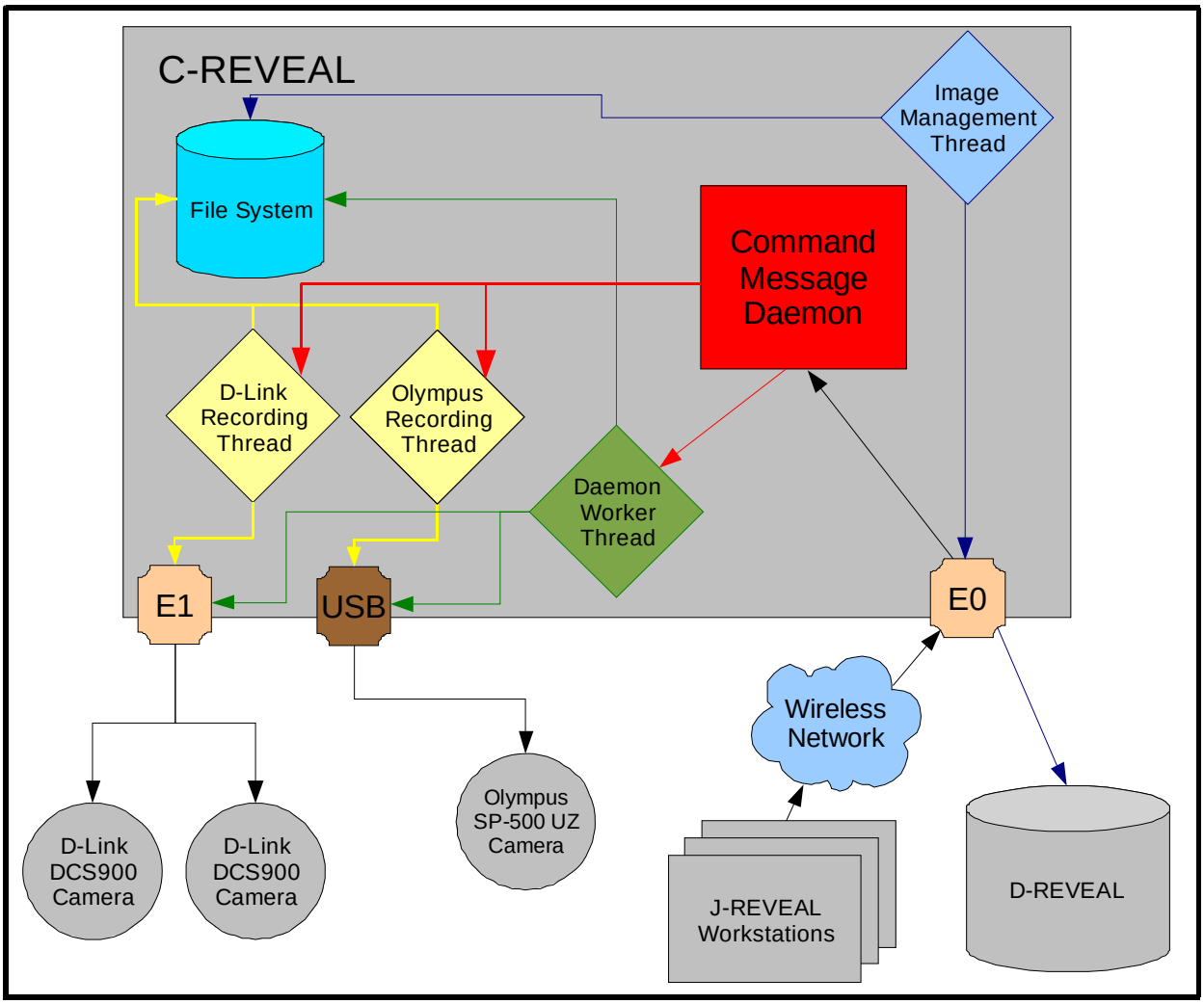

Figure 2 - Depiction of various process threads inside the C-REVEAL computer and their interactions with each other and with the external components of the REVEAL system via Ethernet (E0, E1) and USB connections.

3.1 – C-REVEAL Message Command Daemon and Protocol

Inside C-REVEAL, the CapServer software runs a TCP daemon that listens on port 5555 for incoming connections. Once a client has connected, a client handling thread is spun off that will handle all communication with that client over that TCP socket for the duration of the client's connection.

The client initiates the connection by sending a 62 character command message to the server, this command message details the command to be executed, along with the camera to be accessed, and up to three parameters to carry out the command. All command messages are 62 bytes in length (with the exception of the SETCONF command, which can have a length greater than 62 bytes because the total length of the command is sent inside the first 62 bytes) and all three parameters must be specified either with actual or dummy values. I decided to restrict the size of the command message in order to simplify and streamline the socket receiving code for the CapServer daemon.

In response to the command message, the CapServer returns a 128 byte acknowledgement message to the client. This acknowledgement message either details an explanation of why the command could not be processed (unknown command, improperly formatted parameters, etc.) or a WAIT command instructing the client to await the command response. After issuing this acknowledgement, the daemon spins off a worker thread to handle the command and returns to listening for more commands on the socket. Following a WAIT command the server will send one of four types of response messages:

1. A 128 byte message containing the information the client requested. This type of message is frequent in response to Get/Set Camera Property commands, Get/Set Camera Configuration commands with short parameter sets, and server information queries like requests for connected camera information.

2. A 128 byte OVERSIZE message telling the client that the command resulted in a response too large for the 128 byte standard buffer, and also informing the client of the size of the actual response message to follow. After the OVERSIZE message, the actual response message is sent over the socket.

3. A 128 byte STREAM message telling the client that it should prepare to receive a stream of a specified number of JPEG images encoded in an MJPEG stream. After sending the STREAM message the actual stream is sent one image frame at a time as quickly as the video should be displayed. For example: if there is 1 second between the capture time of each image, then 1 image will be sent each second. CapServer will send the frames until each queued frame has been sent or the client has closed the socket, as such, it is suggested that the SENDVID command responsible for a STREAM response be sent over a dedicated socket connection.

4. A 128 byte BINARY message telling the client that it should prepare to receive a binary message of specified length. After the BINARY message is sent, the actual binary data follows. This type of message is sent in response to the SENDIMG command.

This time, the fixed length response size was chosen to make coding of the thin, command line test client easier to use and debug. Altering the response protocol to indicate a single message type and following message size could streamline the communication channel and improve performance, but the 128 byte

opening window is so small, the impact of such a change to the overall performance is unlikely to be significant.

3.2 – C-REVEAL Daemon Worker Threads

As mentioned above, after each request is received a worker thread is spawned to service that request before the daemon returns the appropriate response to the client. These worker threads query the file system to retrieve recorded images or image streams, configure camera properties, and requests for capturing single images on demand for the client.

For file system and DCS900 camera interaction these worker threads really don't have to worry too much about running into the persistent recording threads mentioned in section 3.3. However, due to device driver restrictions in accessing the Olympus SP-500 Ultra Zoom camera, only one thread can be capturing and downloading an image from the camera at a time. If more than one thread tries to access the camera at once, the camera locks up and must be shut down and restarted manually by means of the on-off button.

The Olympus SDK did not account for the possibility of two threads trying to access the same camera at the same time, so I was forced to implement a locking mechanism to protect the SP-500 from multiple simultaneous access attempts. So far my manual lock seems to be performing fairly well, but once the recording threads are set up to be started and stopped by a client, it'd be advisable to disallow a SP-500 Ultra Zoom capture when the SP-500 Ultra Zoom recording thread is actively recording video frames.

3.3 – C-REVEAL Video Acquisition Threads

Each C-REVEAL system is responsible for capturing video from its connected cameras. This activity is also managed by CapServer, as it spawns a worker thread for each camera. Because video is just a series of still frames concatenated together and displayed at a specific frame rate, the recording threads are really only saving individual still images. The Olympus and D-Link recording threads saves the stills differently due to their respective protocols, but they both make use of the Windows file system for cataloging and storing the frames until C-REVEAL needs to upload them to the database.

The threads keep track of the time the images were recorded by naming the frame by the date and time the image was recorded. The thread associates each recorded image with its corresponding camera by saving the frame in that camera's image directory. Borrowing a frame management system I observed in the Digital Video Recorder industry, each camera's directory is further subdivided into a directory tree made up of year, month, date, hour, and minute directories, and the frame is saved inside the appropriate minute directory. I don't know if this saves any real execution time, but when it comes to searching for files in Windows Explorer, this structure performs significantly better than having every frame stored in the same directory.

Currently, the CapServer daemon spins off a video recording thread for each connected camera as the daemon begins execution. The plan is to change this behavior so that video recording starts and stops only when a command message to start/stop recording so is received from a client.

The Olympus recording thread utilizes the Olympus Developer's SDK and accesses the SP-500 Ultra Zoom camera via a USB cable. Because the SP-500 is not designed for video capturing, and because the

camera's images are much larger than those captured by the DCS900 IP cameras, the Olympus recording thread is limited to recording at most around 1 frame every 5 seconds. In contrast, the DCS900 is a digital video camera that sends JPEG frames in MJPEG format over an HTTP PUSH resulting in tremendous throughput that allows the D-Link Recording thread to save frames at rates of 20 frames per second per camera.

3.4 – C-REVEAL Image Management Thread

To compliment the image recording threads, CapServer processes many commands for interaction with the recorded images, which include: retrieving single images (SENDIMG) or videos (SENDVID), lists of available images (IMGLIST), and deleting images (DELIMGS).

The SENDVID command requires that CapServer compile a list of images to meet the request parameters' camera id, start date-time, and end date-time, or a number of frames from the start date-time. To actually build this list CapServer employs the method used by the IMGLIST command to traverse through the camera subdirectories and find images that fall in between the date-time values (or keep track of how many images it has found already, in the case of a frame count request). To increase efficiency during this lookup process CapServer evaluates the year, month, date, hour, and minute subdirectories to determine whether it is necessary to traverse into each directory down the tree.

At this time, the image deletion is only executed via the client-initiated DELIMGS command, but incorporating that command's deletion methods into an image maintenance thread is underway.

4 – J-REVEAL Contributions

J-REVEAL is a Java application developed by Dr. Gabriel Taubin to host the various computer vision image manipulations required by parts 2, 3, and 4 of the REVEAL project. J-REVEAL is also responsible for configuring each C-REVEAL unit, so it required additional development to speak C-REVEAL's TCP command protocol as well and GUI components for configuring C-REVEAL components.

4.1 – J-REVEAL Configuration Panels

Dr. Taubin had already created GUI panels for retrieving and setting the configuration details of the D-Link DCS900 cameras, so I only had to take the existing GUI interface and change the communication channels so that J-REVEAL spoke with C-REVEAL instead of directly to the DCS900s.

J-REVEAL had not previously communicated with the Olympus SP-500 Ultra Zoom cameras though, and so I had to decide which of the available parameters were worth making available to the user, and then develop GUI panels for setting those Olympus parameters.

These tasks were not difficult, but they did require a good amount of time and resulted in the creation of several additional java classes.

4.2 – J-REVEAL Retrieval of C-REVEAL Images and Video

From the begging, it was planned that J-REVEAL would perform on-demand image captures via the C-REVEAL cameras, in addition to configuring them. This was so that users could make sure that the C-REVEAL cameras were pointed in the right direction, and to capture extra images of noteworthy archeological artifacts. Issuing the Image Capture command was straightforward, retrieving the correct image from C-REVEAL after that took a little more doing. The largest obstacle in doing this was resolving the encoding differences between the way C-REVEAL's C++ socket writer was writing out binary data, and the way J-REVEAL's Java socket reader was reading it. For quite some time the data would all get transferred, but the image could not be viewed, but employing Base64 encoding/decoding resolved this problem.

In addition to displaying single images from C-REVEAL, it also became necessary to facilitate the streaming of recorded video from C-REVEAL to the J-REVEAL application so that the overall project could continue progressing while the project team awaited delivery and integration of D-REVEAL.

Dr. Taubin already had J-REVEAL successfully streaming MJPEG video from DCS900 cameras directly, so it was a matter of unplugging the HTTP back end of his video viewer and connecting it into the TCP Command Protocol developed for C-REVEAL. This was more difficult than the changes required for the configuration panels, but by instantiating a dedicated socket to the server for each MJPEG stream, J-REVEAL was able to watch several streams from each C-REVEAL system at a time.

5. – Future Work

While I have completed a great deal of work, the scope of this project leaves a few unfinished items of business to tend to. These items include:

- Finish the D-REVEAL database image uploading and C-REVEAL image deletion thread that is responsible for loading the database and preventing the C-REVEAL computer from running out of disk space.
- Enable Start/Stop recording commands for stopping/starting the Camera Recording threads that currently only start on CapServer initialization.
- Define additional meta-data for recorded images, and implement an efficient way to store that information until database upload. The current plan is to embed the information in the JPEG files.

6. – Conclusion

The REVEAL project aims to propel archeological surveying into the cutting edge of CVPR technology by providing field archeologists with tools to allow them to reconstruct recovered fragments into the objects they used to be without having to disturb the fragments themselves.

For the REVEAL project I developed the C-REVEAL software system for configuring cameras, capturing images, and recording video. This work has provided a means for acquiring and managing imaging data in order to load the D-REVEAL database full of information the archeologists can use to reconstruct history.

Acknowledgements:

I would like to thank Gabriel Taubin for allowing me to work on this project and for providing sound counsel and advice in my attempts to resolve issues encountered while working on it. I would also like to thank my lovely wife Sara for her understanding and support as I put in long hours and late nights trying to solve the various issues involved in developing this project.

References:

[1] – Taubin, Gabriel – Grant Proposal for the REVEAL project.