CAVECamX - Developing an Improved CAVECam

Jorge Pacheco, Nathan Hui

June 11, 2015

1 Abstract

The CAVECamX project aims to develop a device for generating immersive 3D spherical images that is a significant improvement over the previous generation CAVECam by providing more flexible algorithms and more intuitive interfaces. This will give scientists and researchers an accurate and immersive way to document environments. The CAVECamX uses two Sony QX-1 cameras mounted on top of a precision gimbal to gather a series of images that, when stitched together, produce a high resolution 3D spherical image. The sensors used in this device will improve the quality of the 3D content over that generated by the previous generation CAVECam by improving the resolution and depth perception, as well as including additional position data. Currently, we have produced an algorithm that dynamically generates orientations for any given set of camera parameters, a web interface for remote connection and control of one Sony QX-1 camera, and software modules to gather the data coming from the CAVECamX. The overall goal for the CAVECamX is to increase access to 3D visualization technology by providing a simple and easy-to-understand interface for generating immersive 3D content.

2 Introduction

Imagine the possibility of experiencing any place in the world by seeing what another person has seen and being where he or she has been. Not long ago, this idea was preposterous, but Dr. Tom DeFanti (UCSD Calit2) envisioned a solution where a set of images, stitched together in a spherical stereographic image, could generate a realistic 3D environment. This was the birth of a system known as the CAVECam, which is a system that generates 3D immersive spherical views that allow users to experience immersive 3D content.

The original CAVECam enabled the generation of immersive virtual reality media by providing stereo images in a spherical pattern. Dr. Tom DeFanti envisioned this system as a way to capture a 3D representation of the temples located at Luxor, Egypt. Today, this system has evolved and is used in many different locations where capturing accurate representations of spaces or the environment is desired. The CAVECam has enabled the documentation of beautiful interior scenes in Florence, Italy, Egyptian temples at Luxor, and caves in El Salvador, among others. However, as revolutionary as this technology is, it does not provide the level of flexibility and usability necessary for a layperson. Configuring the CAVECam consists of setting individual camera parameters and using a dated orientation generator that limits the functionality and potential of this technology. We propose creating a new iteration of the system named the CAVECamX that will allow for total control of the setup while providing an easy-to-use interface.

The CAVECam is a system that, at the time, provided the best resolution (12 MP), and provided a usable LCD interface for controlling the movement of the camera rig, but did not provide simultaneous control over stereo camera configuration. In particular, camera setup required the user to manually adjust exposure, resolution, and focus for each set of pictures, making the process slow and cumbersome. In contrast, the CAVECamX is a low-cost high-resolution 3-D visualization tool that will enable researchers to easily generate immersive 3-D content using high resolution stereo cameras. This system will, from the ground up, be designed to replicate and improve on each of the CAVECam components. First, we will replace the Lumix GF1 cameras with Sony QX-1, which is capable of providing better resolution and real-time feedback. Second, we will develop a precision gimbal that will provide total coverage along both pan and tilt axes. Third, we will provide a better software that will generate camera orientations based on camera parameters.

Fourth, we will incorporate an IMU (Inertial Measurement Unit) and an LRF (Laser RangeFinder) to enable georeferencing of objects within the scan. Finally, we will incorporate programmable interfaces to enable users of the CAVECamX to quickly and effectively configure the device with custom camera and orientation configurations.

3 Technical Material

3.1 System Overview

The CAVECamX, at a high level, is composed of seven key components. The include the precision gimbal, Telemetry Management System. Panorama Orientation Generator, Real-Time Control Code, Camera Control Code, and dual camera network configuration. The first six components were split between the two CSE 145 groups, and the last component was handed off to a Calit2 staff engineer (Chris McFarland).

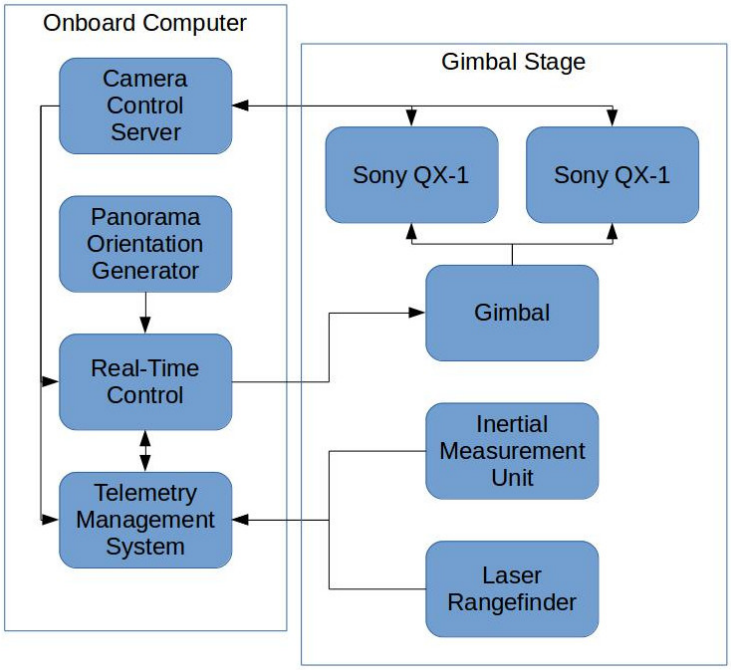

Figure 1: CAVECamX System Diagram

Figure 1 shows the full system diagram of the CAVECamX. The CAVECamX is physically separated into two components - the onboard computer, and the gimbal. The gimbal serves as the main structural platform for all the sensors, while the onboard computer serves as the high-level controller and interface for the entire system.

3.2 Panorama Orientation Generator

The panorama generator is the heart and soul of the CAVECamX project. It is an algorithm that provides full control over the creation of images that generate the immersive 3D environment. The code starts by asking the user some information about the cameras being used. By inputting the sensor sizes along with the focal length of a camera and its lens, it is possible to mathematically predict the field of view that such a camera provides. In mathematical terms, field of view is often expressed as angular size of the view cone, or angle of view. The formal definition of field of view is given in Equation 1 where FOV stands for field of view, being a function of the sensor size and focal length.

$$FOV\,(s,f) = 2 \times arctan\left(\frac{s}{2f}\right) \tag{1}$$

After the field of view of each picture is calculated, the algorithm will attempt to create a data structure that will hold the points in space corresponding to locations where a picture will be taken. To do this, the user must specify the overlap desired between adjacent pictures. In addition, the user must be aware of the stitching algorithms used to create the panoramas, as well as the saliency of the environment being imaged. Some stitching software recommends a particular amount of overlap needed in order to ensure enough common features amongst images. If there are not enough features, pictures will not match correctly. We have establish overlap amounts of 30% on vertical, and 30% horizontal relative to each picture edge.

3.3 Real Time Control Code

The Real Time Control (RTC) module functions as the communications module between all the software and hardware components. Its main function is to provide a smooth transition from picture to picture by managing the timing of the individual components.

One component that the RTC communicates to is the Camera Control code. The RTC receives notifications from the Camera Control code that the cameras have taken a picture. This allows the RTC to ensure that it does not progress to the next picture before the current picture is complete.

The second component that the RTC manages is the Telemetry Management System TMS). The communication between these modules provides a feedback system that measures the motion on the gimbal by means of an Inertial Measurement Unit (IMU). By relying on hardware to measure settling times, the CAVECamX is able to minimize the amount of time between picture transitions.

The RTC will maximize the efficiency of the system by minimizing the time needed for the CAVECamX to settle while providing just enough time for the cameras to take the picture. This allows the CAVECamX to optimize the time it takes to complete any scan.

3.4 Telemetry Management System

The Telemetry Management System (TMS) functions as a data czar for all of the sensors on board the CAVECamX. The previous CAVECam did not include any measurement devices on board, and as a result, users had a difficult time matching objects in the scans from the CAVECam to physical locations in the world. By using an IMU and an LRF, the CAVECamX will be able to georeference objects in it's view by performing specific translations and rotations of its known scan locations, and from that information, be able to concretely identify a particular object in view as being at a specific location on the globe.

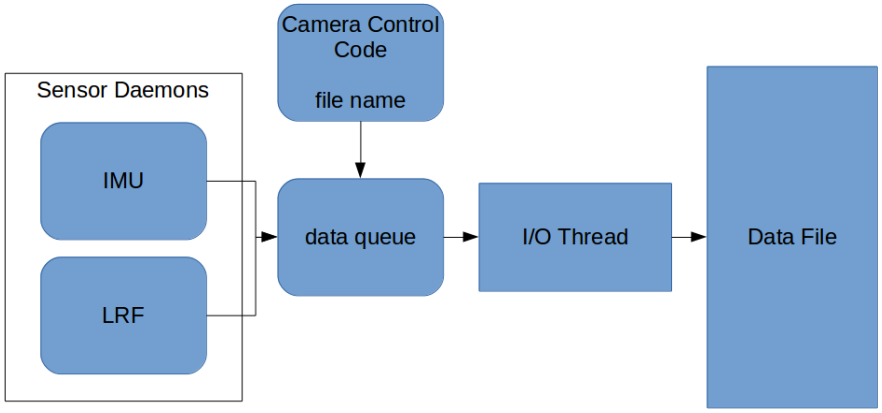

Figure 2: Telemetry Management System Topology

The TMS consists of a software daemon for each sensor, a data queue, and a I/O thread, as shown in Figure 2. Each sensor is managed by a daemon, so that updated values are constantly available for polling.

The RTC module has a software handle which enqueues a snapshot of sensor data, a timestamp, and the current image filename for writing. The I/O thread then asynchronously writes data frames from the queue onto the disk. This software topology ensures that the disk I/O speed never directly impacts the ability of the CAVECamX to gather data.

4 Milestones

4.1 Milestone 1

4.1.1 Panorama Orientation Generator

For Phase 1, Jorge Pacheco concentrated on developing an algorithm that tested the functionality of the gimbal. During this stage, the gimbal was still in development. We implemented a relatively naive algorithm that generated a large number of orientations, which we then used to test the gimbal's movement and response.

We used the locations in space to measure the settling times needed in order to provide for smooth transitions from picture to picture. The limiting factor at this stage were the focus and exposure times for individual images, since settling times for the gimbal were negligible.

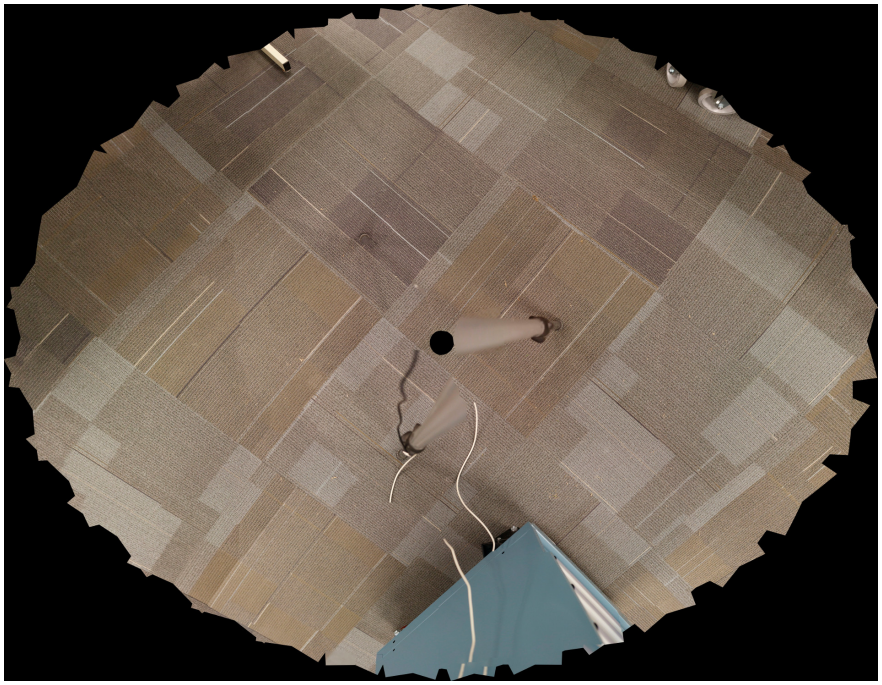

Figure 3: Phase 1 Stitch Result

To test the code, we began to take a 2D scan of the lab. The resulting stitch of the first 200 pictures is shown in Figure 3. The full scan would have required approximately 800 pictures in total.

4.1.2 Real Time Control Code

For first milestone, Jorge Pacheco focused on the hardware and software components as well as reimplementing previous CAVECam code in Python. We decided to use Python because is one of the easiest programming languages that provided a simple serial communications interface. When first communication with gimbal was successful, some settling time values were hard coded in the code as to provide smooth transitions from different gimbal configurations.

4.1.3 Telemetry Management System

The goal for the TMS at this stage was providing a basic pipeline that captured data from the appropriate sensors and wrote them to a file. Nathan Hui was able to to achieve this using a Python thread with a thread-safe deque. The data from the queue was asynchronously written into a hard-coded file on disk. We did a basic functionality test, demonstrated at https://youtu.be/E2BtxC87Fm8. This shows Hui running the telemetry manager using iPython and displaying the last few lines of the output file each second.

4.2 Milestone 2

4.2.1 Panorama Orientation Generator

For Phase 2, Jorge Pacheco refined the implementation of the algorithm in order to optimize the number of pictures needed to create nice panoramas. Originally, the Panorama Generator code was written by Sergei I. Radutnuy on a previous iteration of CAVECam, but code had some bugs[1]. As a result, some of the panoramas generated using this algorithm were not successful. In these cases, the code did not provide enough overlap. This undesired functionality arose from the fact that the old algorithm focused on generating different yaw increments depending on pitch angles. Even though the implementation seemed correct, the code did not generate enough features between pictures and some panoramas failed to stitch properly. We revised the algorithm, and then decided to create a new algorithm from scratch that concentrated on generating the correct overlap for each camera.

We tested this code again, scanning the entire lab. The resulting stitch is shown in Figure 4.

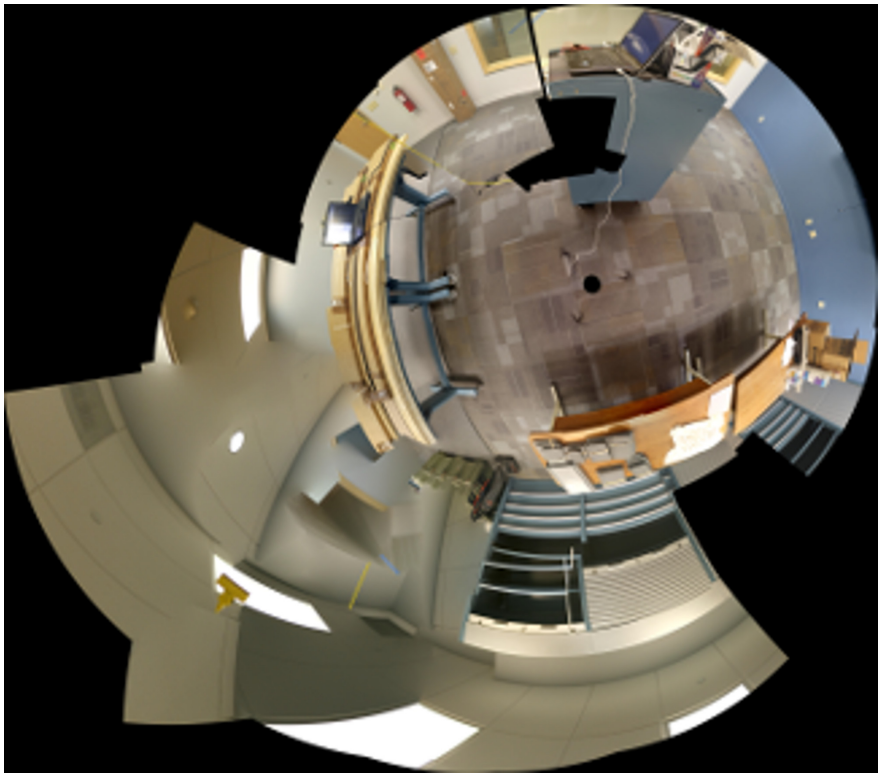

Figure 4: Phase 2 Stitch Result

Figure 5 shows an example of the incorrect overlap. Here, the camera positions were generated with a targeted vertical overlap of 30%.

[1] https://github.com/UCSD-E4E/sacp/tree/master/panorama

Figure 5: Overlap Error due to Incorrect Field of View

4.2.2 Real Time Control Code

For this phase, Pacheco was able to specify fixed bounds of movement for the system. The gimbal by design allows 360°movement on pan axis, and 180°on tilt axis. For convenience, we decided to define the vertical range to be 0°to 180 °, with 0°pointed up. We also decided to define the horizontal range to be 0°to 360°. The initial position of the gimbal was set to 0°pan, 90°tilt (pointed forward). This convention was taken in order to avoid working with negative degree values and making the Panorama Generator Code easier to work with.

4.2.3 Telemetry Management System

The goal for the TMS during Phase 2 was to provide an interface for near real-time processing of sensor data to provide feedback for the RTC module. This was implemented by creating the concept of sensor daemons for each sensor, and allowing each sensor daemon to update sensor data independently of the main thread execution. This allowed the main thread to asynchronously check sensors for a particular state. Unfortunately, no demonstration video was made of this capability due to uncertainties in sensor selection.

4.3 Milestone 3

4.3.1 Panorama Orientation Generator

During Phase 3, we improved the panorama orientation generator to fix issues relating to the improper calculation of the camera's field of view. Slight errors in calculating the camera's field of view rippled down into the orientation generation code, which affected the actual amount of overlap between to images. After manually measuring the cameras field of view, we determined that original values used for sensor sizes and focal length were incorrect. We updated these values into the code resulting in correct panoramas and correct points in space. This problems can be minimized if field of view of camera is tested before being used.

After fixing the overlap issue, we tested the code in the CSE courtyard. The resulting stitch is shown in Figure 6.

Figure 6: Final CSE Courtyard Stitch

4.3.2 Real Time Control Code

For phase 3 of the Real Time Control code, Pacheco specified the communications protocol on the data link layer between the RTC and the gimbal controller. The program will open a serial port using python. It will then pass a string in the format "$Pxxxxx,yyyyy", with P as a 0 or 1 indicating whether or not to take a picture, and xxxxx and yyyyy as 5 digit numbers corresponding to gimbal steps.

$$Step(\theta) = \frac{\theta \times 1600}{360} \tag{2}$$

Equation 2 gives the conversion of degrees to microsteps.

This string is sent at time intervals determined by hardcoded values in the Panorama Orientation Generator. For each image, two strings are sent with the same orientation parameters. The first is used as an indicator for the gimbal to move. The following string uses the flag 1 to specify that no movement is necessary but that the gimbal controller should command the camera to take a picture.

For example, system will send the string"$000691,00094" which tells the gimbal to move to location 155.5°from forward, and 21.3°from vertical. It will then send another sting "$100691,00094" with the first character set to 1 instead of 0 to tell the gimbal to trigger the camera at the current location.

4.3.3 Telemetry Management System

The goal for the TMS during Phase 3 was to provide an integrated software package that could be loaded onto the CAVECamX platform to provide data capture. This would include integration with the RTC module and the Camera Control module. Due to unmet software development dependencies, the TMS was not integrated with the other software modules, however, the functionality required to integrate the module was added to the TMS. This functionality is demonstrated in https://youtu.be/j54ONzDRrYY, which shows the TMS running in a simulation harness and the last few lines of the output file printed to the terminal. This demonstrates that the TMS is capable of compiling and saving the data, provided that all pertinent software handles are integrated properly.

5 Overall Deliverables

The Guatemala expedition deliverable has been eliminated due to the project being behind schedule. At current, the hardware for the system has not been properly integrated, so the system cannot be thoroughly tested prior to the deployment. Additionally, the user interface and multi-camera technology are not mature enough to deploy with confidence that they will work in the field. In addition to this, we have determined that there is not an immediate and critical use case for the CAVECamX on the Guatemala expedition, especially in its current state.

Future deployments include trips to the Duomo in Florence, Italy, as well as cave systems in New Mexico. These deployments are contingent upon the completion and integration of the CAVECamX.

6 Technical Difficulties

The gimbal is being developed by Dimitri Schreiber. He encountered some problems with the mechanical components of the gimbal, but he is already making arrangements in order to complete it as soon as possible. Due to this inconvenience, testing of the real time communication between the Real Time Control Code and gimbal has not been possible. We will be postponing the real time processing of data from sensors and gimbal firmware by a few days until gimbal implementation is finished. In the mean time, next iterations of software components are being developed in order to minimize delays of project deliverables.

7 Conclusion

The CAVECamX project resulted in several key developments in the effort to build a new CAVECam system. These key development include developing new gimbal hardware and control software to properly orient the cameras, dynamic orientation generation algorithms, and minimalist data acquisition software packages. Because of the scope of this project, the development accomplished by this team does not comprise the entire development push needed to complete the CAVECamX. However, the developments in this project provided the some of the key components to building the new CAVECamX.

Future development on this system will consist of polishing the existing software modules and improving the current hardware components. From a top level perspective, certain key technologies still need to be fully developed to enable the project to be fully integrated.

8 References

1. http://www.ccs.msstate.edu/conferences/NSFcyberbridges2014/presentations/keynotes/DeFanti-Great%20Data%20Cyberbridges%206-1-14.pdf

2. http://ucsdnews.ucsd.edu/archive/newsrel/general/2011_06defanti_luxor.asp