Three-dimensional imaging with the Argus sensor array

Evan C. Cull, David P. Kowalski, John B. Burchett, Steven D. Feller, David J. Brady

Fitzpatrick Center for Photonics and Communication Systems and Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708

ABSTRACT

The Argus project uses an array of computers and cameras as a means of investigating telepresence and real-time three-dimensional imaging. In this paper we will briefly discuss telepresence from an information flow and visualization perspective. The paper also includes a detailed description of the Argus hardware and a software layer developed to manage the imaging and computational resources. MPEG-2 and feature extraction will be described as parallel compression systems for the Argus camera array.

Keywords: Argus array, three-dimensional imaging, stereoscopic imaging, cone-beam tomography, parallel processing

1. INTRODUCTION

The divide between digital and analog systems is particularly pronounced for multidimensional imaging. While holographic and stereoscopic sensors allow us to record the illusion of three-dimensional scenes, they do not in fact construct three-dimensional models that are needed for fully interactive scene visualization. Computed tomography and other three-dimensional scene analysis schemes create true three-dimensional digital models from sensor array data. In most cases, however, end users cannot process full three-dimensional models and thus do not demand them. However, there are cases in which full three-dimensional models are useful, and, as technology advances, end users will possess the ability to process and render three-dimensional models.

Argus is a testbed for investigating research topics in the area of real-time three-dimensional imaging. The ability to image a space, transmit the information to a different location, and virtually recreate the space is the primary goal of this project. This process is known as telepresence [1]. The Argus array is composed of a Beowulf cluster computer and an array of video cameras [2]. Specific applications developed for use on Argus have involved the creation and transmission of stereo pair as well as volumetric images. This paper will present an overview of the project to date, discussion of the imaging and processing paradigms, and near-term goals.

1.1. Telepresence Modalities

The most basic form of telepresence is simple two-dimensional video conferencing – a dedicated sharing of video and audio information to provide a virtual collocation of the participating parties. Two-dimensional video conferencing has been available for nearly 30 years, initially using phone lines and now using the internet. Traditionally, telepresence provides a two-dimensional view or set of two-dimensional views. In order to provide a more immersive environment for virtual interaction, however, three-dimensional information is required. Other research projects focused on developing immersive telepresence systems include [3], [4], [5].

Three-dimensional video systems can be classified into one of three basic types. The simplest three-dimensional video uses a pair of two-dimensional images to form a stereo image which must be viewed using some type of image separating glasses, with anaglyph (red/blue) or polarizing lenses to achieve an illusion of three-dimensionality. While this stereo-pair video does not directly contain three-dimensional information, it is included in this discussion because the human brain can interpret the pair of images as a three-dimensional scene. However, this type of system introduces depth-of-field and focus errors, and does not allow the user to manipulate the view of the environment. By placing multiple view "stations" within the environment, the user can change the view of the scene, but is still limited to choosing between camera locations. Further, using a system such as anaglyph glasses or polarizing lenses requires the

Three-Dimensional TV, Video, and Display, Bahram Javidi, Fumio Okano, Editors,
Proceedings of SPIE Vol. 4864 (2002) © 2002 SPIE · 0277-786X/02/$15.00

use of grayscale images (as in anaglyph), or require very specialized projection systems (as in the use of polarizing glasses).

In order to provide a user the ability to choose any arbitrary view of a three-dimensional scene on a standard two-dimensional viewing device, such as a television or computer monitor, a three-dimensional model of the scene must be created from which a two-dimensional view can be reprojected. There are two modalities for generating three-dimensional models – surface models or volumetric models, both of which can be created using limited view backprojections [6]. Surface models of an environment use multiple cameras to generate a three-dimensional model of the surfaces in the environment, sometimes referred to as a "mesh model." Such a model visualizes the environment as a set of triangles or similar primitives. Many different research projects exist which try to reconstruct surface models from three-dimensional scenes, such as [3], [4], [5] [7].

The third system is volumetric visualization. In comparison, it utilizes a three-dimensional scalar field to store the environmental geometry [6], [8]. This allows users to visualize the complete environment without feature loss, and is good for complex objects with internal structure. Such an environmental model is also known as a full tomographic model. These models are often used for three-dimensional object recognition and tracking. However, the processing power required for display of volumetric data is higher than that for mesh visualization. Other research projects that deal with volumetric imaging based on visible light cameras include [9], [10], [11]. One of the visualization modalities of the Argus array, that of generation of a "marionette" model of the human body, compresses this volumetric image information down to several regions of interest – in this case joint locations – thus compressing the data greatly. Several groups have done research with related joint or human body tracking systems [12], [13], [14], [15].

1.2. Telepresence Systems

As with telepresence modalities, telepresence systems can also be separated into three basic categories. The original telepresence systems were integrated capture and display systems. These systems could also be used to capture a stereo view and project that view for the user. Such systems include holographic and autostereoscopic recording and display solutions. These systems may be represented by the block diagram shown in Figure 1. This approach was developed in the age of analog processing. With recent advancements in ubiquitous digital processing power, a better system could be developed [16].

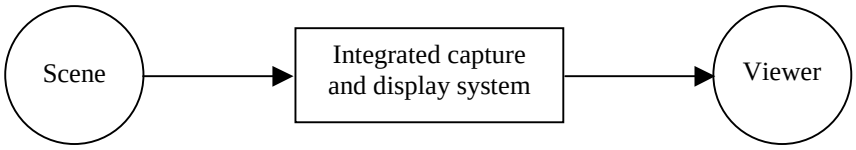


Fig. 1: An integrated stereoscopic capture and display system.

A more advanced telepresence system utilizes discrete capture and display systems. These systems include stereo camera recording/holographic or stereo display pairs. A block diagram for this approach is shown in Figure 2. The fundamental attraction of this approach is that it supports multiple users. It is also more adaptive and allows separate optimization of capture and display system parameters.

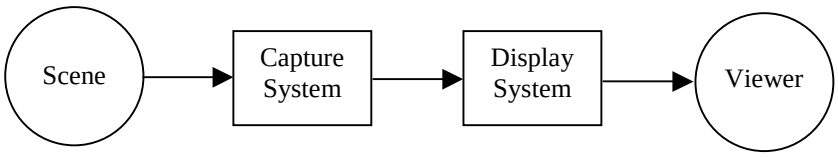
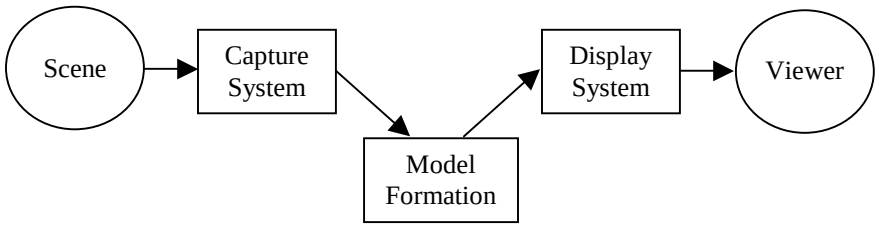


Fig. 2: A discrete capture and display system.

The most advanced model for telepresence systems uses separate capture, inversion and display systems. As exemplified by tomographic imagers, these systems form complete three-dimensional models of the scene and reproject these models for viewers. Figure 3 illustrates this approach.



2. THE ARGUS ARRAY

The Argus system is comprised of two major sections, the sensor space and the processing system. The sensor space is essentially the imaging studio itself – an area surrounded by cameras a; a networked computer cluster provides the processing power. By combining these two parts, we in effect achieve a set of networked sensors with embedded processing capabilities. One of the Argus design goals was to build the system efficiently, quickly, and in a cost-effective manner, prompting the use of less-expensive and readily available components such as off-the-shelf desktop computers and web cameras rather than high end workstations and scientific grade CCD cameras with embedded digital signal processing.

2.1 The Sensor Space

The sensor space consists of a fourteen-foot diameter camera framework. The framework is constructed from three-inch diameter metal pipe in an octagonal shape. The 64 cameras are equally spaced within a wooden circular frame that circumscribes the octagonal pipe construction. Arranging the cameras along the wooden circle instead of along the main octagonal frame simplifies camera alignment issues by placing the cameras along a near perfect circle. In order to keep light sources and objects outside the sensor space from interfering with the pictures taken within the sensor space, a twelve foot tall black curtain was originally hung on the inside of the framework, with holes cut for the lenses of the cameras. For similar reasons, the floor was covered with black carpet. Recently, with the use of color cameras, background subtraction has been possible, allowing the removal of the black curtains and carpeting. Eight 500 W halogen lamps were placed in the ceiling above the space provide even lighting. Figure 4 shows a picture of the Argus sensor space with the original grayscale cameras installed.

Fig. 4: Picture of the Argus sensor space.

2.2 The Sensors

The cameras initially used in Argus are grayscale CMOS focal planes with 2.5mm lenses with a 320 x 240 resolution. There are sixty-four cameras evenly spaced along the circumference of the sensor space, pointing inward. A custom designed power supply / controller for the cameras is used to provide the capacity for frame synchronization. The cameras are mounted using a simple clamping system which allows manual adjustment and aiming of the cameras. Recently, the grayscale CMOS cameras were replaced with Firewire (IEEE 1394) based CCD cameras offering color pictures with 640 x 480 resolution and 24 bits per pixel. The current mode of camera operation is 320 x 240 4-2-2 YUV, which results in 37 Mbps of data per camera at 30 frames per second capture speed.

2.3 Camera Alignment

Camera alignment is important for all imaging algorithms currently running on Argus. When simply viewing the video captured across different cameras as stereo pairs, cameras aimed slightly off center or rotated a few degrees off vertical can be quite distracting. These visual distractions are even more serious for the three-dimensional modeling algorithm used in Argus because they can seriously degrade the quality of the models. Due to the imprecise method of mounting the cameras, alignment is a significant challenge. Initially, the height of the cameras and spacing within the ring were set using a laser leveling system. From there, the cameras were aimed by hand until they reported a point source at the center of the imaging volume to be near the center of the captured image. The imprecise camera mounting hardware caused this method of alignment to be accurate within two pixels. A digital transformation of the captured image is required to align the cameras further. By imaging a leveled straight edge, camera rotation angels were calculated. The measured pixel location of the point source and the rotation angle of the camera was then be used to translate and rotate the image to improve alignment.

2.4 The Computer Array

The imaging and compression algorithms that run on Argus require the ability to process data from all cameras simultaneously. At the very least, a computer system for Argus would need the capacity to input data from 64 sources and perform rather intensive numerical transforms on each source in real time. A Beowulf-enabled Linux cluster computer provides the input bandwidth as well as the required computational power. Our cluster is built using 32 Pentium II 400MHz dual-processor computers connected with 100Mbit Ethernet. These machines are networked using two 24-port switches with a 2 Gbps backplane. A single dual processor computer is used as a master node which is used to control the cluster and provide a gateway to the outside world. The master node is connected to the cluster with gigabit Ethernet. Figure 5 shows the configuration of the Argus computer network.

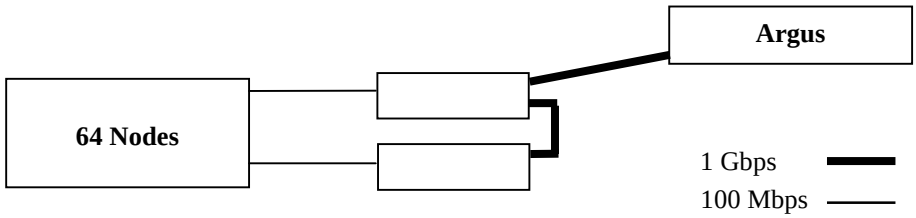


Fig. 5: Argus network connectivity.

3. THE ARGUS RESOURCE MANAGER

A software layer is required to control the large amounts of data generated by the Argus cameras [17]. Managing the operation of the Argus array from the client display program is the Argus resource manager. This software allows the end user to specify the tasks which individual nodes complete, whether data capture, processing, or relaying. The end goal of the Argus resource manager is to implement a distributed, heterogeneous sensing and analysis network using a flexible framework. To this end, we have implemented a messaging system on a distributed framework that allows

clients to connect to the Argus array computing system and request services from a predefined set of available resources, in both the data capture and data processing realms. We have also included the ability for the framework to simultaneously support multiple devices as well as allow for the quick insertion of new devices into the array. Routines for data analysis, computation, and distribution are predefined.

The framework itself was designed with both the client and server in mind. The client model is designed to allow for a flexible connection request interface that allows the user to specify particular node operations. The server, which runs on the individual nodes in the system, allows any node to have exterior connectivity (i.e. connect to the networked world outside the Argus array while maintaining internal network security), to have attached sensors (although nodes may be attached to the system for the sole purpose of providing computation power), and to collect and/or process data. This framework results in a connection-request client-server model with an easy-to-use programming interface. It also allows for a variable data source/sink model as well as non-persistent connections. Figure 6 shows a simplified model of the Argus resource manager layout. Use of the resource manager will be discussed with respect to the different imaging modalities below.

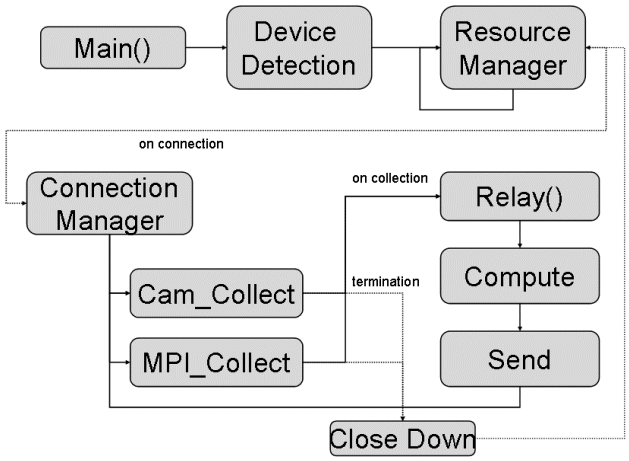


Figure 6: Argus Node Layout

4. THREE-DIMENSIONAL IMAGING WITH ARGUS

Three different imaging modalities have been developed on Argus. The first modality implements stereo-pair view generation. This is the simplest imaging modality, as it requires very little computational power from the cluster as well as gross camera alignment. We chose not to implement the second telepresence modality as described in the introduction; rather, we implemented full tomographic modeling, the third telepresence modality, which we present as a volumetric, rather than surface, model. Recently, we have been working on feature recognition for data compression, resulting in our third imaging modality – joint-tracking on human subjects, which we refer to as the marionette model. The goal of all three modalities is to provide a remote user with a real-time telepresence in the imaging space, meaning that the user should be able to view the contents of the imaging space from the perspective of his choosing.

4.1 Stereo View Generation

This approach used for viewing the sensor space relies on positioning the cameras in the array to produce a close approximation to any desired view of the space which the cameras themselves are capable of generating. In this approach the user informs Argus of his position and orientation in the virtual space, specifying a particular view of the imaging environment. Argus then uses the position of the sensors in the environment to produce a close approximation of the requested view. To make the experience more realistic, Argus generates a stereo pair of images based on the location of the user. With anaglyph or shutter based stereo glasses, this gives the user a three-dimensional perspective of the environment. Figure 7 shows a stereo pair of images taken by Argus cameras.

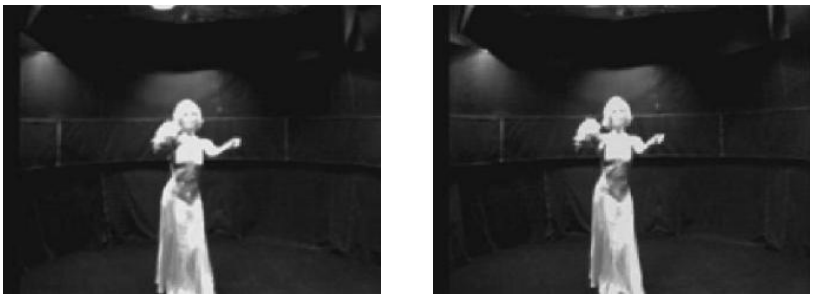


Fig. 7: Images from adjacent Argus cameras.

This modality is relatively easily implemented using the Argus resource manager. The client will simply connect to two nodes simultaneously, requesting the camera data. The nodes will then package in the resource manager data structure and send the data over sockets to the client machine. The client (knowing *a priori* the organization of the array) will request a pair of adjacent images. The current stereo-pair client also requests camera data from nodes adjacent to the nodes being viewed, so that when the user rotates the view to the right or left, the data from the camera is already cached at the client, allowing for smoother rotation of the viewpoint. Stereo-pair viewing allows the client the most basic form of telepresence. Although the user can view the enclosed environment from any angle, the viewpoint is limited to the camera locations.

4.2 Cone-beam Tomography

The second imaging modality of the Argus array generates volumetric models of the sensor space similar to tomographic algorithms used by medical CAT x-ray machines. The tomographic algorithm used by Argus is based on Feldcamp's cone-beam algorithm [18]. The adaptation of Feldcamp's algorithm to visible light cameras treats the individual pixels of a camera image as rays which project from the focal plane of the camera out through the volume. The set of rays from a camera approximate a cone, hence cone-beam tomography. A three-dimensional model is formed by additively backprojecting the rays from the set of Argus cameras into a voxel space. With intensity thresholding and backgrond removal applied to the images prior to the back-projection, the intensity of each voxel represents the probability of an object being at the corresponding position in physical space. The specifics on the adaptation of Feldcamp's cone-beam algorithm for use with visible-light cameras are available in [19] .

Feldkamp's algorithm consists of two major steps. The first step is a filtering step that reduces the background components of the images while enhancing the objects of interest. This uses a convolution filtering process and can be efficiently performed with a fast Fourier transform. Compared with the filtering step, back-projection is computationally expensive for reasonable sized voxel arrays due to the number of rays that must be projected into the voxel space. In order to approach real-time performance, we developed a parallel implementation of Feldkamp's cone-beam algorithm, using the computational capabilities of the Beowulf cluster. The parallel coded tomography algorithm uses the MPICH implementation of the message passing interface (MPI) standard [20]. For each voxel image generated, the Argus nodes capture images from each camera and perform the convolution / backprojection process. This leaves a separate voxel set for each image. These voxel sets are passed from node to node for summation until a final voxel array with information from all Argus cameras is created. Figure 8 shows a ray projection of a reconstructed three-dimensional volume (left) in Argus beside a single-camera view (right) of the same scene.

Fig. 8: The projection of the tomographic model (left), 2D image of scene (right)

4.3 Human Marionette Generation

The third imaging modality currently available on the Argus array is the "marionette model" visualization. This system allows the client to view a stick-figure model of a human target within the imaging space. This modality implements the data capture and processing as a multi-step algorithm across multiple machines, fully demonstrating the power of the Argus resource manager. Development of this modality began with black-and-white cameras and has recently been converted to a full-color system. This system uses multiple cameras to track regions-of-interest within the array. It also takes advantage of the voxel-space mapping of the Argus environment as described above.

The initial step in the process is to detect the region of interest on a single camera. On the original array, this was done by threshold detecting point sources. An intelligent algorithm was used to detect multiple centers of high-intensity light on a single image. After parsing the image, "active" locations were condensed to a few centers by proximity and density. These levels were determined experimentally. If a line connecting two active points with half its length touching other active points, or if the points were within two pixels of each other, they were left active, if not, they were assumed to be outliers and deactivated. Each of the center points of these lines was then subjected to the same test. The resulting centers (after several iterations) were then assumed to be the centers of distinct light "blobs" within the image. In Figure 9, a lamp and a mini-flashlight are used to test the correct identification of two distinct, but closely spaced light sources. The algorithm was able to correctly separate the centers to a distance of two inches separation between the sources, or approximately two voxels. The centers are denoted by black squares

Fig. 9: Two point sources.

Figure 10 is the result of two point sources being placed within the array, one at the center and one at a random location, which was changed to test sensing in different locations within the array. For this picture, four adjacent cameras were used to test near-camera interference. For both points, a single center was detected in three-dimensional space with four overlapping rays. The black squares in the figure represent the individual voxels that are turned "on." The large spheres represent regions of voxel overlap.

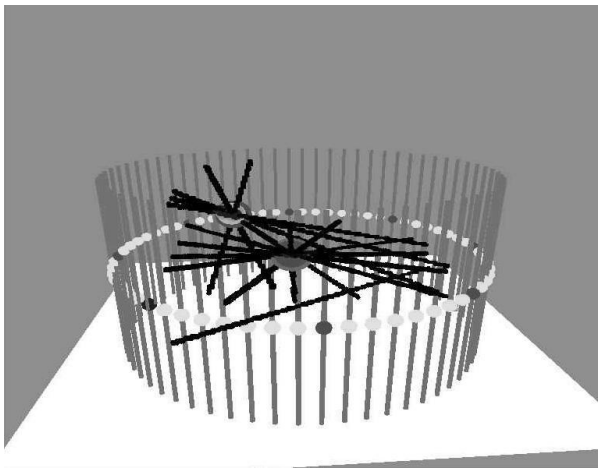


Fig. 10: Tracking two point sources

This algorithm of looking for overlapping rays is the reason that all points detected in three-dimensional space are assigned a probability of containing the source. As the number of overlaps increases, the probability of the source being in that location increases. When the algorithm is complete, the location with the highest probability is assigned as the location of that source. If multiple points are detected that are within 3 voxels of each other, the centroid of those points is assigned as the center of that source.

The four pictures in Figure 11 represent the stages in the detection algorithm. The first picture is the original image. Color matching is used to select the blue fabric target in the center of the picture. As one can see, there is a significant amount of background noise in the image data. In order to remove the noise, a cell structure is artificially imposed on the image. In the case of the third picture, 3x3 pixel cells were used to segment the image. Within each cell, the number of active pixels was totaled. If that number exceeded half the cell area, that cell center was added to a list of active points in the image. These active points are displayed in the third image. The target region is well defined with few outliers. These active pixels can then be further clumped if desired. In the final picture, two cameras generated an active points list as per the above algorithm. These points were then back projected as in the black-and-white case discussed above. The resulting active voxels are shown in the picture in black. An intersection point was located at the placement of the fabric in the array. Moving the fabric resulted in a corresponding movement in the intersection point. We have recently re-implemented the intersection algorithm using a one-pass function to determine the closest point of intersection of two rays traveling through the imaged space using their end points. This saves time by reducing the amount of data transmitted between nodes as well as reducing the computational complexity of the data reduction stages.

Fig. 11: Color source detection

Figure 12 outlines how the resource manager is used to implement the marionette model generation. The RM blocks denote a resource manager process running on one of the Argus nodes. The Worker blocks are processes spawned by the node resource managers to perform a task. The client connects to the Argus resource manager through the Argus Server machine. It then requests access to different cameras within the array, specifying the target colors for each camera. The socket manager then spawns processes on the individual nodes. In the diagram below node 0 is a computing node while nodes 1 through 4 are camera nodes. The camera node worker threads capture the image data and process it looking for the colors specified by the client. When they have processed the image, as shown in Figure 10, they send determine the ray paths through the array specified by the centers of the regions-of-interest they have found. They then pass these paths to the compute node, which intersects the rays to determine the locations of the regions-of-interest in three-dimensional space. When it has determined these coordinates, the compute node passes the data to the client, which is able to display a model of the target being imaged by the array.

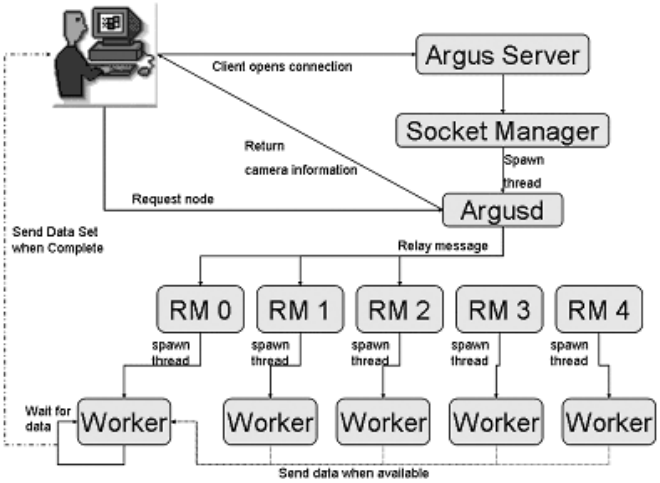


Fig. 12: Resource Manager usage for marionette generation.

5. DATA COMPRESSION

With the grayscale CMOS cameras, each camera within the Argus array is capable of producing a 600 kilobits per second video stream, yielding a 37.4 megabits per second bandwidth for the entire array. The upgrade to color IEEE 1394 cameras pushes the array's bandwidth to 6.6 gigabits per second (Gbps). In practice, the full bandwidth is limited by the Ethernet switch backplane bandwidth of 2 Gbps. Transmitting gigabits of information per second between the sensor space and a remote user currently infeasible except on private networks. Storage also becomes a problem with data generation rates as large as these. While a standard video compression technique may be applied to each Argus camera individually, the Argus camera array has the additional advantage that any particular Argus camera contains quite a lot of redundant information when compared to neighboring cameras. The views of two adjacent cameras are shown in Figure 7.

When viewed as a whole, the set of 64 images that Argus captures is a set with incremental differences between adjacent images. A normal video sequence is also a set of images with very small differences between images. The close similarity between video data and Argus data allows standard video compression algorithms to effectively compress data from Argus. The MPEG-2 video coding standard was chosen for Argus image compression because of its versatility and compression performance. The basis of MPEG-2 compression is inter-frame motion estimation that is applied to an image separated into discrete cosine transformed blocks. Individual images within a video sequence compressed via motion estimation can depend on prior images, or both prior and following images. Normally, a video sequence in time is compressed as the images are recorded, requiring only a small cache of images on which to base the motion

estimation. The major challenge in adapting the MPEG-2 standard to work on the Argus array was parallelizing the MPEG-2 algorithm [21].

To satisfy data dependencies between the inter-compressed frames, an encoding sequence as shown in Figure 13 must be used. This chart only shows the compression sequence for a quarter of the Argus processor / camera pairs—the other three sections are compressed in the same manner. The frames are labeled I, P, and B. The I frames are not compressed with inter-frame compression. P frames are only backward dependant to previous P or I frames for compression. B frames are compressed based on data from both previous and later I or P frames. The blank spaces on the chart show when a processor is idle.

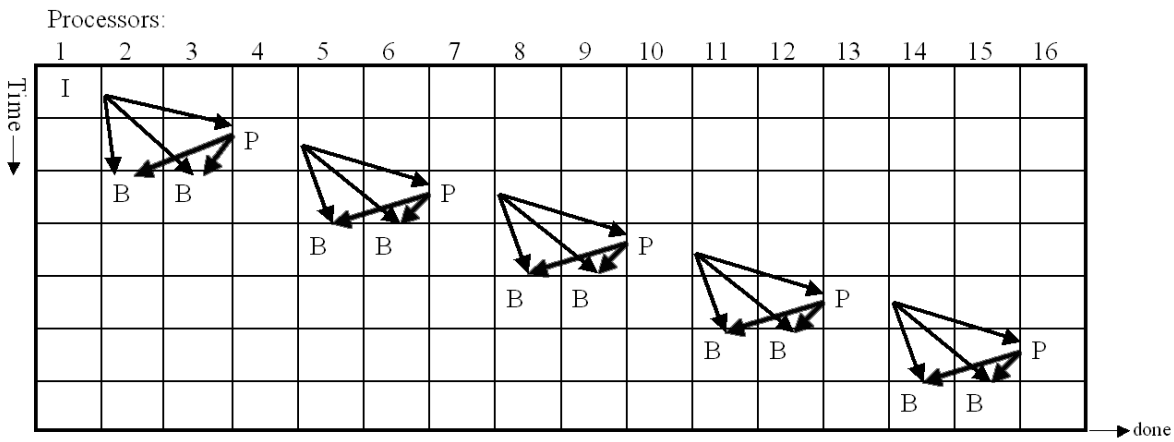


Figure 13. MPEG-2 encoding sequence.
Each horizontal segment is approximately two seconds. Arrows denote data dependencies.

To improve the performance of the MPEG-2 implementation on Argus, a pipelined version was developed. This version minimizes processor idle time and provides a constant compressed image set after a short start-up latency. A chart depicting the flow of the pipelined sequence is shown in Figure 14.

Processors:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
I1	X	X	X	X	X	I1	X	X	X	X	X	I1	X	X	X	
I2	X	X	P1	X	X	I2	X	X	P1	X	X	I2	X	X	P1	
I3	B1	B1	P2	B1	B1	I3	B1	B1	P2	B1	B1	I3	B1	B1	P2	
I4	B2	B2	P3	B2	B2	I4	B2	B2	P3	B2	B2	I4	B2	B2	P3	1 done
I5	B3	B3	P4	B3	B3	I5	B3	B3	P4	B3	B3	I5	B3	B3	P4	2 done
	B4	B4	P5	B4	B4	P5	B4	B4	P5	B4	B4	P5	B4	B4	P5	3 done
	B5	B5		B5	B5		B5	B5		B5	B5		B5	B5		4 done
																5 done

Figure 14. Pipelined MPEG-2 encoding on Argus, each horizontal segment is approximately two seconds.

6. CONCLUSION

The Argus project was developed as a testbed for imaging and telepresence research, emphasizing real-time performance and the use of off-the-shelf technology. With Argus, we have explored different telepresence imaging modalities. By implementing both the stereoscopic and tomographic algorithms, we are able to compare the human response to different imaging processes as well as the resource requirements of each of the methods. With stereoscopic imaging we present a discrete display and capture system with low computational complexity. The full tomographic backprojection presents a much greater computational load which stresses the parallelism of the computer and camera array to create a capture, inversion and display system. We have also examined two different compression schemes. The MPEG-2 scheme compresses the complete dataset by exploiting the spatially redundant nature of the camera data. The marionette application also represents a compression methodology by reducing a complicated data set to a small set of interesting features.

The resource manager allowed us to simultaneously develop these different imaging and compression algorithms by providing a layer of separation between the algorithms and the hardware / networking configuration. The stereoscopic imaging application provides a baseline test for the networking and Argus-to-client interactions. Marionette is a higher-level algorithm that allows us to test the resource manager's ability to simultaneously manage heterogeneous computational tasks.

7. ACKNOWLEDGEMENTS

Argus is an Air Force Office of Scientific Research project originally supported by the Distributed Optoelectronic Processing for Multidimensional Digital Imaging grant, number F49620-99-1-0229.

REFERENCES

1. Buxton, W. "Telepresence: integrating shared task and person spaces." in *Proceedings of Graphics Interface*. 1991.
2. Ridge, D., D. Becker, P. Merkey, and T. Sterling, "Beowulf: Harnessing the power of parallelism in a pile-of-pcs." in *Proceedings, IEEE Aerospace*, 1997.
3. Raskar, R., et al. "The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays." in *SIGGRAPH*. 1998.
4. Mulligan, J., V. Isler, and K. Daniilidis. "Performance evaluation of stereo for tele-presence." in *International Conference on Computer Vision*. 2001.
5. Schreer, O., et al. "A Virtual 3D Video-Conference System Providing Semi-Immersive Telepresence: A Real-Time Solution in Hardware and Software." in *eBusiness and eWork*. 2001.
6. Girod, B., G. Greiner, and H. Niemann, *Principles of 3D image analysis and synthesis*. Boston: Kluwer Academic Publishers, 2000.
7. Narayanan, P., P. Rander, and T. Kanade. "Constructing Virtual Worlds Using Dense Stereo." in *Proceedings of the Sixth IEEE International Conference on Computer Vision*. 1998.
8. Dyer, C., *Volumetric Scene Reconstruction from Multiple Views*, in *Foundations of Image Analysis*, L. Davis, Editor. Kluwer: Boston, 2001.
9. Vedula, S., S. Baker, and T. Kanade. "Spatio-Temporal View Interpolation." in *ACM Eurographics Workshop on Rendering*. 2002.
10. Marks, D.L., et al., "Visible cone-beam tomography with a lensless interferometric camera." *Science*. **284**(5423): p. 2164-2166, 1999.
11. Dachille, F., K. Mueller, and A. Kaufman. "Volumetric Global Illumination and Reconstruction Via Energy Backprojection." in *Symposium on Volume Rendering*. 2000.
12. Aggarwal, J. and Q. Cai. "Human Motion Analysis: A Review." in *IEEE Nonrigid and Articulated Motion Workshop*. 1997.
13. Liu, X., Y. YZhuang, and Y. Pan, "Video Based Human Animation Technique." *ACM Multimedia*, p. 353-362, 1999.

14. Cheung, K., et al. “A real time system for robust 3D voxel reconstruction of human motions.” in *IEEE Conference on Computer Vision and Patern Recognition*. 2000.
15. Kakadiaris, I. and D. Metaxas. “Model-Based Estimation of 3D Human Motion with Occlusion Based on Active Multi-Viewpoint Selection.” in *IEEE Computer Vision and Pattern Recognition*. 1996.
16. Brady, D.J., et al., “Information flow in streaming 3D video.” in *SPIE Proceedings*, p. 306-321, 2001.
17. Helbig, T. “Development and Control of Distributed Multimedia Applications.” in *4th Open Workshop on High-Speed Networks*. 1994.
18. Feldkamp, L., L. Davis, and J. Kress, “Practical cone-beam algorithm.” *Journal of the Optical Society of America A*, **1**: p. 612-620, 1984.
19. Marks, D.L., et al., “Cone-beam tomography with a digital camera.” *Applied Optics*. **40**(11): p. 1795-1805, 2001.
20. Gropp, W., E. Lusk, and A. Skjellum, *Using MPI*. Boston, MA: MIT Press, 1994.
21. Haskell, B., A. Puri, and A. Netravali, *Digital Video: An Introduction to MPEG-2*. Chapman & Hall, 1997.