Instrumentation for Rapidly Acquiring Pose-Imagery

by

Douglas S. J. De Couto

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Computer Science and Engineering

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1998

Author .... [illegible] ....
Department of Electrical Engineering and Computer Science
May 22, 1998

Certified by ....
Seth Teller
Associate Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .... [illegible] ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Instrumentation for Rapidly Acquiring Pose-Imagery

by

Douglas S. J. De Couto

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 1998, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Computer Science and Engineering
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis describes the design, implementation, and initial evaluation of a device for quickly and easily acquiring large numbers of pose-images with a small amount of human effort. Pose-images are digital images annotated with the 6 degrees-of-freedom (DOF) pose (3 DOF position and 3 DOF orientation) of the camera when the image was acquired, and are useful for performing automatic reconstruction of three-dimensional models of the real world. The device presented here, Argus, is a manually propelled cart equipped with GPS receivers, wheel encoders, and inertial navigation sensors. The outputs of these sensors are combined using a Kalman filter to provide an absolute global pose estimate for each picture. Images are acquired using a digital still camera, mounted on Argus with a motorized pan-tilt head. Images are stored to disk or tape on Argus, annotated with their associated pose; images are then automatically downloaded to mass storage in the laboratory after each acquisition session.

This thesis evaluates Argus by looking at its throughput — the rate at which Argus can acquire pose-images — and compares this throughput with manual pose-image acquisition throughput. This thesis also presents an initial evaluation of the pose estimate quality. Argus allows us acquire pose-images much faster than manual acquisition, but pose estimate quality can be improved.

Thesis Supervisor: Seth Teller
Title: Associate Professor of Computer Science and Engineering

Acknowledgments

This thesis is the culmination of five years at MIT, and of two years working on this project. I would not have written this thesis if not for the friendship and support of many people throughout my time at MIT.

First and foremost, I would like to thank my parents for supporting and encouraging me throughout my entire education, and providing me with the opportunities that have made my education possible. I also thank my brother Jesse for his friendship and support.

I am grateful for the support of all my friends — ace-boys and ace-girls — at MIT, including everyone on 5West and at the Computer Graphics group. I am especially grateful for all my friends on the MIT sailing team, and in the MIT Nautical Association (especially my coach, Fran Charles). Without MIT Sailing, my life at MIT would have been incomplete.

On a more practical note, this work was carried out as part of the City project, of which Argus is just one small part, and has involved many people. It has been a pleasure to work with Professor Seth Teller in designing and executing Argus — this has been my most ambitious project yet! Neel Master was a guru, explaining graphics and Windows programming to me; he also helped with figures. Satyan Coorg provided advice on my thesis, and graphics topics in general. Mike Bosse has explained Kalman filtering and inertial navigation to me more times than I can count, and his energy kept me going for the last few months. He also helped to acquire and process the navigation test data. Sami Shalabi explained f-stops, ISOs, and shutter speeds to me many times, helped me render the model of Argus, and provided many breaks from the thesis grind. Marek Teichmann was a patient office mate and helped me understand figures in Latex. Tara Schenkel provided the amazing models of Argus for many of my figures. My good friends Michael Taylor and Anders Hove gave valuable input on my drafts. Finally, I have enjoyed working with everyone at Korbin Systems and Peace River Studios.

My graduate program was partially supported by the Bank of Bermuda, and my undergraduate program was partially supported by the Bank of N.T. Butterfield and Son, also located in Bermuda.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis motivates and describes Argus, a device for quickly and accurately acquiring pose-images, using a navigation system and a digital camera. Pose-images are images of the real world labeled with the camera pose when the picture was taken. The pose consists of the 3 degrees-of-freedom (DOF) position and 3 DOF orientation of the camera, along with the camera settings and a timestamp. Pose-images are fundamental data objects that are useful for reconstructing metric three-dimensional computer models of the real world [CMT98, CT97b]. Realistic computer models of the world are useful for many tasks, ranging from architectural visualization and urban planning applications, to entertainment and virtual reality applications. However, almost all computer models of the real world are currently created by hand with a laborious modeling process [JLF95], or using semi-automated techniques that require substantial human input [DTM96].

The goal of the City project [Tel97] at the MIT Computer Graphics Group is to demonstrate the feasibility of an end-to-end system for the automatic reconstruction of textured, 3D models from images of real urban environments [CT98]. A key feature of the City project's approach is the use of camera pose estimates to expedite the reconstruction process [CT97b, MLPT97, CT97a] by geo-locating the camera in a globally absolute coordinate system. As part of the project, we need to acquire large pose-image datasets which cover significant urban areas. However, existing image datasets are either not labeled with pose data, or do not cover the kinds of urban regions that the City project is concerned with. Therefore, pose-image acquisition is a significant component of the City project effort.

Although we have manually collected a pose-image dataset (§1.1), our goal is to automate the pose-image acquisition process to the greatest possible extent. Argus, our pose-image acquisition device, provides this automation. We have designed and built Argus (§1.2) to accelerate and add accuracy to the manual pose-image acquisition task. Our experiments so far show that Argus does indeed accelerate the pose-image acquisition process, allowing us to automatically capture about 50 pose-images every 10 minutes (50 pose-images every 6 minutes plus time for moving between locations). Not only is image acquisition with Argus faster than manual acquisition, it is also highly repeatable, eliminating much of the operator error in moving and operating the camera. In terms of accuracy, Argus can automatically determine the pose for each image to within 1–2 degrees of attitude, and 2–7 meters in position; our manual collection effort was able to determine position to within several centimeters using professional surveying techniques, and relative attitude to within a degree [CMT98]. Although the initial pose accuracy delivered by Argus is not as good as the manual pose accuracy, the Argus navigation system, which determines pose, is still under development, and has yet to be fully tuned and calibrated. We expect position accuracies on the order of a few centimeters and rotation accuracies on the order of several milliradians in the future. Also, the Argus navigation system is completely automated, and delivers globally absolute pose estimates, as opposed to the relative pose estimates delivered by manual collection efforts, which require further manual processing to produce global pose estimates.

The following sections in this chapter describe pose-image acquisition (§1.1) and Argus (§1.2, §1.3) in more detail. Chapter 2 provides an overview of other pose-image acquisition systems, as well as a discussion of some systems that reconstruct camera pose data without the need for special pose-measurement instrumentation. Chapter 3 outlines the design guidelines followed in creating Argus, and sketches the major elements of the design of Argus. Chapters 4, 5, and 6 present detailed descriptions of the mechanical, image acquisition, and navigation subsystems used on Argus, respectively, while chapters 8 and 9 evaluate the pose-image acquisition throughput and accuracy of Argus.

1.1 Manual Pose-image Acquisition

We have manually acquired a dataset of about 4,000 pose-images that covers the exterior of our office park, Technology Square in Cambridge. Technology Square is a group of four buildings arranged around a large quadrangle, with trees planted in a grid pattern in the interior (Figure 1-1).

It took two researchers about a week to acquire images of Technology Square using a Kodak DCS420 digital camera, acquiring about 100 images per hour. Images were acquired, tiling a hemisphere (Figure 1-2), at several discrete locations. Each hemisphere of images is referred to as a *node*. The camera was mounted on an indexed pan-tilt head that allowed the relative orientations of each image in a hemisphere to be set according to a predetermined tiling pattern. Images were stored on the camera's local disk, and transferred to computers in our lab after every three or four nodes were acquired.

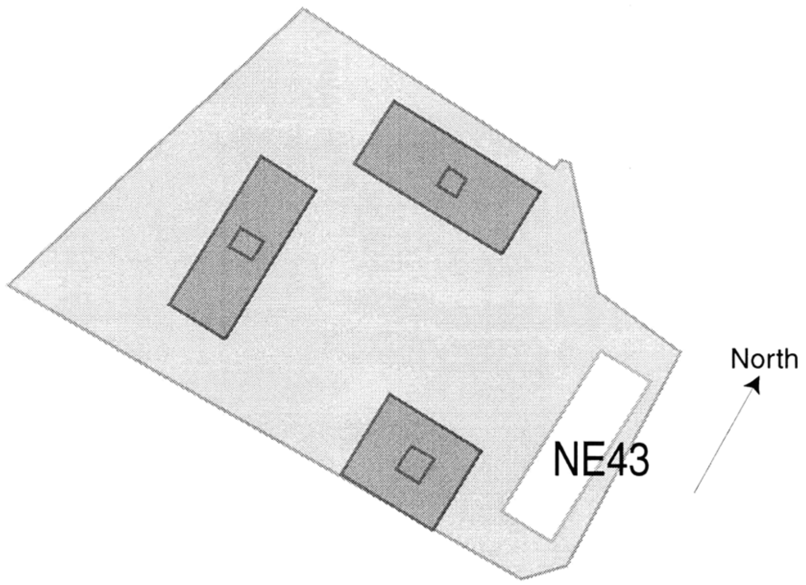

Figure 1-1: Layout of the Technology Square office park. Our laboratory is building NE43.

Acquiring the pose data for the images first involved professionally surveying Technology Square, producing a local coordinate system. Then, individual node positions were manually tied into the survey coordinates, also using professional surveying techniques and equipment. The absolute rotation of each hemisphere in the survey coordinates was determined by post-processing the image data using custom software and fiducial marks in the images [CMT98]; the camera orientation for each image in a hemisphere was then produced by composing the camera's relative pan and tilt for that image with the hemisphere's absolute rotation. The acquisition of our pose-image dataset took about a month of full-time effort by three researchers, and resulted in over 16 gigabytes of raw data, in addition to many more gigabytes of derived data such as mosaiced images.[1]

[1]The image collection and surveying was done by Neel Master and Adam Holt. Post-processing work to recover pose data was done by Neel Master, Satyan Coorg, and Barbara Cutler.

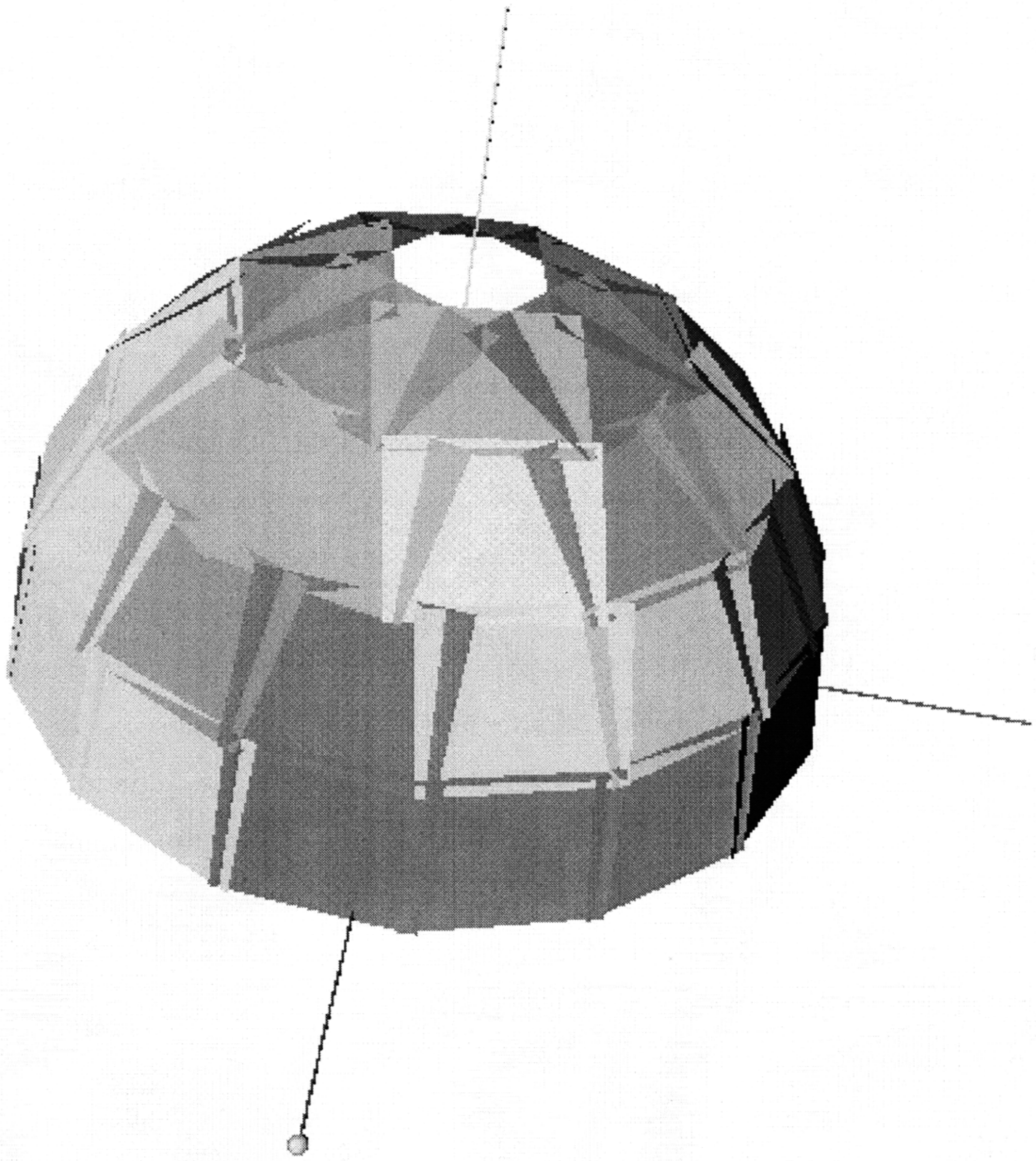

Figure 1-2: Tiling a hemisphere with images. Each shaded panel represents a picture covering part of the hemisphere.

1.2 Argus: Automatic Pose-image Acquisition

The purpose of this project was to automate and greatly accelerate the acquisition of pose-image datasets, so that we can acquire in a day or two a pose-image dataset that would otherwise take several months to acquire manually. We have created a device, Argus, that allows us to speedily and semi-automatically capture pose-image datasets like the one described above (Figure 1-3).

We have automated pose-image acquisition by building a custom designed, manually propelled cart. We have equipped this cart with an image acquisition subsystem, which uses a high resolution still digital camera (Chapter 5), a multi-sensor integrated navigation system (Chapter 6), and mass storage in the form of disks and digital tapes. The navigation system integrates a Global Positioning System (GPS) sensor (Appendix A), inertial sensor, and wheel encoder data via a Kalman filter [Kal60] to produce an estimate of the camera's position and orientation in absolute earth coordinates. A ruggedized PC controls the camera and the navigation system. The control software captures images with the camera, tags them with the navigation system's current pose estimate, and writes the resulting pose image to a mass storage device. The camera is mounted on a motorized pan-tilt head which allows us to automatically capture a full hemisphere of images at each acquisition location. The pan-tilt head can be raised and lowered to allow acquisition of imagery at different heights. Finally, the cart has a leveling system which stabilizes it on uneven ground, producing more reliable pose estimates, and increasing safety. The PC automatically controls all motor movements.

We call our device Argus, after the mythical all-seeing Greek herdsman with 100 eyes (also called Panoptēs) [How89].

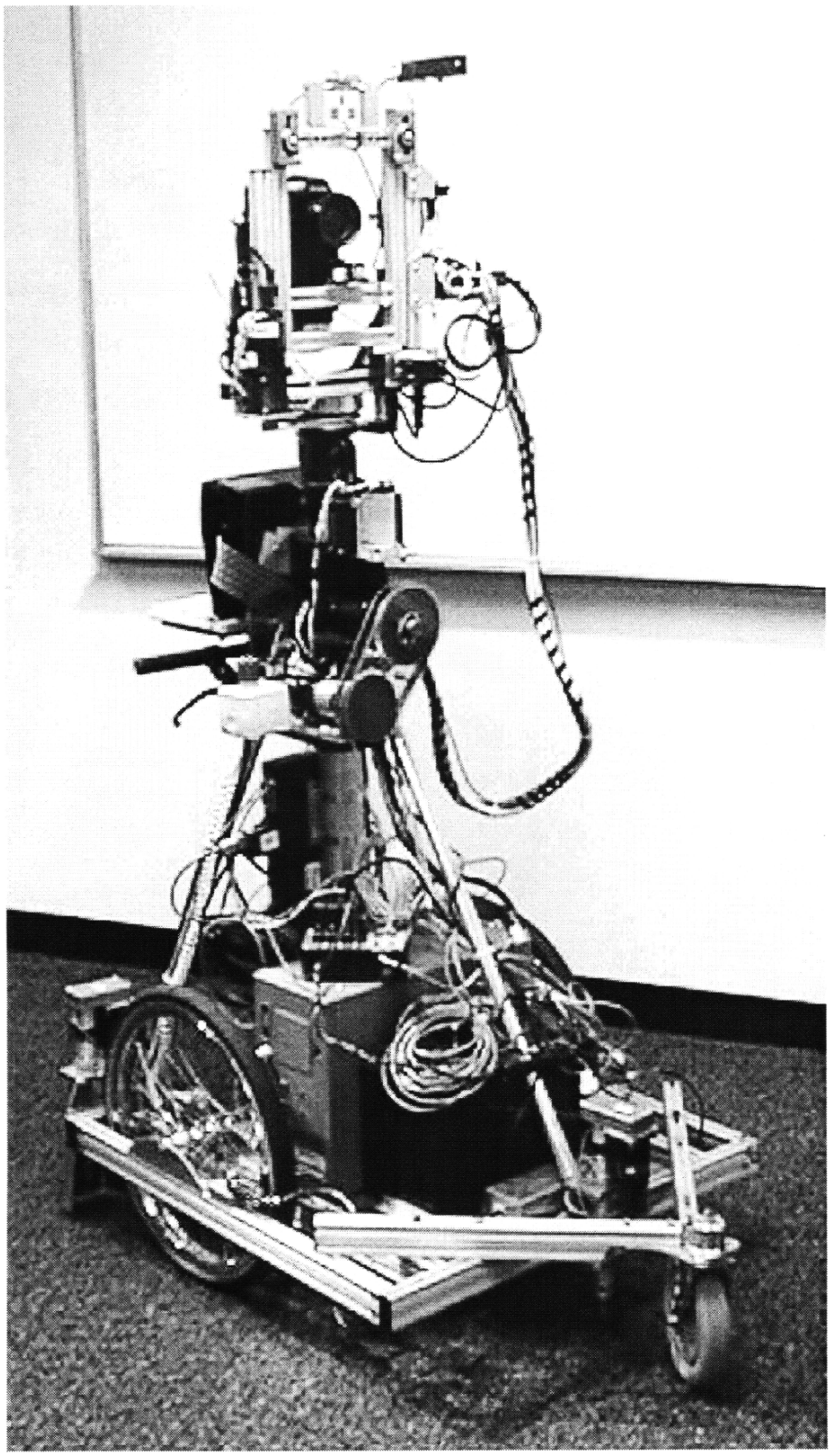

Figure 1-3: Argus: a pose-image acquisition device.

1.3 Using Argus

We have tried to make the operation of Argus as simple and efficient as possible. Argus is first taken outside, and the navigation system is activated and begins logging navigation data. Then, after a period of calibration and alignment, Argus is moved to node locations in and around the area of interest, guided by the operator.

By examining feedback from the navigation system the operator can determine the quality of the current pose estimate. If the pose estimate quality is not sufficient for reconstruction, perhaps because of bad GPS reception, the operator can move the cart or otherwise take steps to improve the navigation system's pose estimate. The operator can also place the cart strategically to maximize the quality of the dataset. For example, there should be a good distribution of pose-images across the faces and corners of each building that the operator wants to capture for reconstruction.

Once the cart has been positioned, the operator can initiate a pose-image acquisition sequence by pressing a key on the console keyboard. From this point on, the pose-image acquisition is completely automatic: after Argus levels itself, the camera pans and tilts under computer control, acquiring a sequence of images which tiles the upper hemisphere around the camera's nodal point. As each picture is taken, it is time-stamped and written with its accompanying pose estimate to the disk or tape drive. Multiple hemispheres of images may be acquired at a single latitude and longitude location, by setting the camera tower at different heights for each hemisphere.

Once data has been acquired at a suitable number of nodes, the cart is returned to the lab. At the lab, the navigation data is post-processed to obtain better accuracy, and images are uploaded from Argus' disk or tape onto centralized mass storage. Then, each image is correlated with its corrected pose data using the timestamp, and the resulting pose-image is entered into a large database, where it is accessible to reconstruction algorithms or further processing.

Chapter 2

Related Work

This chapter describes work which is related to the Argus pose-image acquisition device. Three main pose-image acquisition projects are covered: The Aspen Project [Lip80]; the work of Michitaka Hirose and others aimed at providing data for "synthetic sensations" [HTMW96, HWE98]; and the GPSVan [GB96, Tot95, TB96]. This chapter will also briefly survey some systems that *recover* relative pose from large numbers of images, either as an end or intermediate product, instead of directly measuring global absolute pose when the images are acquired.

This chapter does not address topics such as GPS assisted aerial photography [ASP96] and satellite imagery [SEM97]. Although these topics are related because they involve collecting large numbers of images of the earth and automatically registering the images in a global coordinate system, they are beyond the scope of this thesis. Such systems operate under an extremely different paradigm. First, these systems aim to collect images from remote distances, while we are concerned with relatively close-range images, which provide much more detail than aerial or satellite imagery. Second, aerial or satellite imagery does not typically cover the environment from multiple points of view; it captures the environment looking straight down from overhead, while the pose-image data we acquire with Argus covers the environment from many diverse points of view, primarily looking up and across at buildings from locations on the ground. Third, images acquired with Argus overlap and cover the same parts of the environment from multiple viewpoints. Aerial or satellite imagery usually captures each feature of the environment only once, in a single image; multiple images often only overlap slightly at the boundaries between images. Finally, surveyed ground fiducials are often used as control points in aerial photography.

2.1 Other Pose-image Acquisition Projects

This section discusses three projects which use various forms of instrumentation to collect pose-images. However, each project differs from the Argus acquisition device in one or more substantial ways.

2.1.1 The Aspen Project

The Aspen project [Lip80] collected images of the streets of Aspen, Colorado. These images were then stored on optical videodiscs and used in a system where users could take a virtual drive through the streets of Aspen. Each image corresponds to a particular street location in Aspen, and one of four viewing directions at that street location.

Four cameras were mounted at 90 degree increments around a circle on a car or dolly, pointing forward and backward along the street direction, and to the two sides. As the cameras were moved down the street, a trailing wheel simultaneously triggered the four camera shutters after every ten feet of travel. Street intersections were handled specially.

All the images were acquired during the same time interval on each day, to provide uniform lighting for the image sequences. Also, each sequence of images was taken in a single collection session, by traveling along a predetermined route through the streets.

The Aspen project is similar to the City project in that both projects aim to make real scenes available in virtual environments. However, acquiring data with Argus is different than the Aspen project for four main reasons. First, the end use of pose-images acquired with Argus is to produce realistic 3D models, while the Aspen project was only concerned with displaying discrete images of scenes. This is partially attributable to the current technologies and state of computer graphics when the Aspen project was carried out, which influenced that project's goals.

Second, data acquired with Argus is less constrained than the Aspen project data. Because the goal of Argus is to collect data for 3D reconstruction, Argus collects data while positioned on streets, sidewalks, wheelchair ramps, and many other pedestrian accessible areas. Also, the Argus camera positions are not constrained to a discrete set of positions, while the Aspen project cameras were constrained to street locations, pointing in four fixed directions. Because of this, Argus can acquire multiple images of a particular structure from many different camera locations, while the Aspen project only acquired one image of each location.

Third, the Argus data collection process is completely digital and automatic. There is no need to manually correlate frame counts with camera locations to locate images in space.

Finally, Argus collects pose-images that are located in a single global coordinate system, while the Aspen project data was located in the relatively local coordinate system of the streets of Aspen.

2.1.2 Data for "Synthetic Sensations"

Michitaka Hirose and others have been working on systems that produce virtual environments ("synthetic sensations") using images of the real world. They produce virtual views from recorded images using image interpolation [HTKW94, HTMW96] and other image-based techniques [HWE97, HWE98]. Pose-images are a fundamental data element for their systems, and they have developed two pose-image acquisition systems. Acquired pose-images are then stored and retrieved according to their 6 DOF pose.

The first acquisition device by Hirose et al. [HTKW94, HTMW96, HWE97] was essentially a small rolling cart with a video camera. Rotary encoders measured attitude and position, a GPS receiver provided position data, and a geomagnetic sensor measured absolute direction using the earth's magnetic field. This pose data was integrated by a Z-80 microprocessor and written to the video recorder's soundtrack using a 2400bps modem. Pose-images were loaded into a database on a workstation by using a framegrabber and a modem to read the image and pose data from the video tape.

With this system they collected color images with a maximum resolution of 280 by 210 pixels, and their largest database, was 200 images, about 65 Megabytes of data. This database covered a virtual area of about 20 meters. They cite GPS position resolutions of within 9 meters (1 meter using two receivers); this poor resolution prompted them to use the rotary encoders to provide high quality position data around a small area for their pose data and image interpolation calculations.

The second acquisition system [HWE98] is more sophisticated, with 8 video cameras mounted in a circle on top of a small van. Video data from each camera is recorded at 30 frames-per-second, and indexed with time codes generated by that camera. Pose is measured using a GPS receiver, a 3-axis angle sensor, and a geomagnetic sensor. A PC controls the GPS and video cameras, and logs the pose, video timecodes, and GPS timecodes.

Video data is digitized in color at 720 by 486 pixels (about 1 Megabyte per frame), but only at 10Hz to reduce storage requirements. Pose data is sampled at 5 Hz, and then interpolated to 10Hz using Bezier approximation. Image and pose data are correlated using the recorded timecodes. With the van moving at 20 kilometers per hour, a new set of images is acquired at approximately every 0.6 meters. The images taken by the 8 cameras at each approximate location are merged into a single 4,320 by 486 pixel panorama. The resulting panoramas are used in a virtual walkthrough environment, possibly with warping applied to generate intermediate panoramas.

In [HWE98], Hirose et al. cite GPS resolution of better than 3 centimeters, and attitude resolution of 0.5 degrees for pitch and roll, and 0.9 degrees for yaw. The geomagnetic sensor provides absolute direction in the surface plane accurate to 2 degrees. However, their experiments show attitude errors of up to 1.25 degrees. Their system also has problems correlating pose data to image frames, and correlating image frames between cameras for the panoramas. These problems arise from latencies recording the attitude data (3.7 frames on average) and an inability to externally adjust the timecodes of each of the cameras.

This pose-image acquisition is similar to the Argus pose-image acquisition in that they are recording color images, indexing them with global pose acquired from GPS and other sensors, and storing the whole dataset onto a computer for later virtual walkthroughs. However, the work is different in many significant respects.

First, the pose-image acquisition devices built by Hirose et al. collect relatively low resolution image data, as measured both by pixel count and by the collection process. The Argus acquisition device collects more than four times as many pixels per image frame, and avoids the various problems associated with acquiring video images from tape with a frame grabber. Also, the data acquired with Argus is used to produce an extremely high resolution hemispherical mosaic, compared with the relatively low resolution panorama obtained by the van-mounted system. There is a tradeoff between slowly acquiring lots of high resolution image data at each location (Argus), and quickly acquiring some low resolution data while driving or pushing a cart with a video camera (Hirose et al.).

Second, data can be acquired with Argus in any pedestrian-accessible location, while data acquired with the van system is constrained to vehicle accessible areas such as roads and parking lots. However, this is also a tradeoff: by mounting cameras onto a van, Hirose et al. can collect data over much wider ranging areas much faster than Argus, which is human-propelled.

Third, the Argus acquisition device has much better synchronization between the navigation system, which acquires pose data, and the camera, which is acquiring images. This means that we can be more confident that the pose component of each pose-image actually belongs to that pose-image.

Finally, Argus will provide better pose estimates for each pose-image. Hirose et al. quote an extremely good GPS resolution, but provide no data about the actual GPS accuracy. Argus will eventually provide better positioning accuracy than the van and cart discussed above, and currently provides similar angle accuracies.

The first system developed by Hirose et al. collects much less data than Argus, but their second van-mounted system collects much more data, up to 80 Megabytes per second.

2.1.3 The GPSVan

The GPSVan [Tot95, TB96, GB96] is the product of research into "mobile mapping systems" at the Ohio State University Center for Mapping. Their goal is to produce a system to acquire large amounts of digital data suitable for input into geographical information systems (GIS). The GPSVan is a van with a pair of CCD cameras mounted on the roof for acquiring stereo images of interesting items, a video camera to continuously log the acquisition session, and a navigation system that uses dead reckoning with inertial and GPS sensors.

The primary use of the GPSVan has been for "corridor mapping": locating features on roads and railway tracks. During pose-image acquisition, features of interest (e.g. a utility pole) are imaged with the stereo cameras, and the van's global position is continuously recorded. During post-processing the position of features (i.e. depth) in the stereo camera reference frame is determined, and the features are located in the global coordinate system by linking the camera coordinate system to the van's position.

The stereo image pairs are obtained from the two calibrated CCD cameras which provide digital outputs and allow precise shutter control. The digital image data is sent to a dedicated digital signal processing subsystem, which calculates histograms of each camera's image for shutter control and optionally compresses the images with JPEG compression. This dedicated subsystem then writes the the image data directly to two redundant 8mm digital tape drives. A dedicated real-time clock board, whose time is coordinated with the rest of the system, is used to control camera triggering and the shutter controls, and timetag images. The digital image processing subsystem communicates with a host PC to receive control signals and other data.

The cameras produce 8 bit greyscale images at 768 by 484 pixels, for a total of

743 kilobytes of data per stereo image pair. The image processing subsystem can handle up to 14 Megabytes of data per second per camera, for up to 8 cameras. However, the total data acquisition rate is limited by the tape drives, which can store up to 15 Gigabytes of data at a rate of 0.5 Megabytes per second. This limits image acquisition to one stereo pair about every 1.4 seconds (15,000 stereo pairs or 6 hours of data per tape). Using JPEG compression, [Tot95] claims that storage capacity can be increased by a factor of five. In an actual application of the GPSVan, 80,000 stereo image pairs were acquired over a period of 10 days, which covered features of interest along 750 miles of linear railroad track.

The navigation system measures pose using base-station and mobile GPS receivers that deliver kinematic GPS data every 3 seconds. The dead reckoning (DR) system uses a three-axis reference system with a vertical and directional gyroscope to provide heading and attitude data, along with wheel encoders for measuring heading changes. The DR and GPS data are integrated using a Kalman filter and a nonlinear smoother, which produces pose data at 10 Hz.

In [GB96] the performance of the GPSVan is evaluated for the railroad track survey mentioned above. The integrated pose data from the GPSVan is evaluated by comparing it against the kinematic GPS data only. The kinematic GPS position solutions are cited as accurate to within 10 centimeters for north and east directions, and within 30 centimeters for the vertical direction. The integrated pose solution for the GPSVan has a similar accuracy, but degrades over time with the loss of GPS data to within one meter after two minutes without GPS data. Finally, the north and east accuracy of features located via the stereo image pairs is within 50 centimeters for 95% of the survey's control points; most of this feature location error is attributed to poor attitude data in the GPSVan's DR subsystem.

As with the van-based system by Hirose et al., the GPSVan cannot acquire data as flexibly or for as many locations where Argus can acquire data. Again, however, the GPSVan can acquire more images over a larger area in less time. There exists once again a tradeoff between the density and resolution of pose-image data that is acquired, the area covered by the pose-image data, and the speed with which it is covered. The goal of Argus, providing data for automated reconstruction, means that it is important to have dense and diverse image coverage of areas which are to be reconstructed. That is why Argus sacrifices speed and area of acquisition to gain flexibility in image acquisition.

The image processing system in the GPSVan is very flexible, and its ability to send image data directly the digital tape drives is a major factor in the image subsystem's performance. However, the GPSVan only acquires lower resolution greyscale images while Argus acquires high resolution full-color images, thus the GPSVan data is not suitable for reconstructing realistic textured models of the real world; color is essential. Unfortunately, it is difficult to find a suitable high-resolution, fully digital, color camera. Most color solutions rely on analog video cameras and framegrabbers, which are problematic for many reasons, including lack of resolution and timing problems [TB96].

In terms of pose, the Argus acquisition device can obtain GPS data with the same accuracy as the GPSVan, and can provide superior attitude data. Argus also uses wheel encoder information for short term positioning. More recent versions of the GPSVan, produced commercially, have more sophisticated inertial navigation systems than the simple DR system described in [GB96].

2.2 Pose Recovery: An Alternative to Pose Measurement

The previous section discussed systems that acquired pose-images by directly measuring the pose when each image is acquired. However, there are many systems which use the camera pose of a sequence of images without requiring it as *a priori* input. These systems *recover* the camera pose of each image in the sequence through various techniques, including fiducial marks, human input, and feature correlation. This section briefly discusses three such systems.

2.2.1 Move-X

The Move-X system [EMST91] is designed to overlay video sequences with computer renderings of buildings, to provide a sense of how a proposed building fits into a particular site. The system takes interlaced NTSC video from a handheld or tripod mounted video camera as input, and digitizes it. Move-X assumes that the camera parameters (including pose) can be modeled with a perspective transform, and attempts to recover the parameters of this transform for each frame in the video sequence.

Move-X relies on having a number of fiducial points in each video frame, referred to as "passpoints." The world location of each passpoint is known *a priori*, and they are specially marked in the world by white circles on dark backgrounds. Passpoints must be manually identified in the first frame of the video sequence, after which they are automatically tracked throughout the sequence by the Move-X system. Using the image location of several passpoints (whose world location is known), the Move-X system recovers the camera's perspective transform by solving a system of non-linear photogrammetric equations using Newton's method.

This system has accuracy and stability problems when solving the photogrammetric equations. These problems can be resolved by using more passpoints than necessary, fixing camera parameters such as the camera's focal length and focal point, and smoothing the resulting camera pose across successive frames.

The main drawback of the Move-X system is that the real world must contain artificial and highly visible fiducial marks, whose locations must be determined. This is impractical for a system such as Argus that must function unobtrusively and non-intrusively in a wide range of locations. The other disadvantage of the Move-X system is that it requires manual intervention to locate the fiducial marks in the initial frame of the sequence. It is also not clear how fiducial marks are located in images which do not overlap with the first frame; either the user must manually locate the newly appearing fiducials, or the system must automatically find fiducial marks as they appear in successive images in the sequence.

2.2.2 The Bridges of Paris

In [BCS96], Berger et al. describe a system which automatically computes the camera viewpoint throughout a video sequence. This computed viewpoint is used to overlay lighting simulations onto a video of the "Ile de la Cité" bridges in Paris. Unlike the Move-X system, this system does not require any fiducial marks in the real world. Instead, the user manually marks areas and curves in the initial image that are used to track the camera position throughout the sequence. As the image sequence progresses, the user must identify new features for tracking as old features disappear from the images. This system requires a calibrated camera.

This system is attractive because it does not require fiducal marks. However, the user must manually identify appropriate feature (regions and curves) for tracking, and must continuously identify new features as the image sequence progresses. This system is clearly not suitable for recovering pose information from a very large set of images, as the manual effort required is too high.

2.2.3 *Realise*

The *Realise* project [FLR95] was undertaken to partially automate acquiring 3D models of urban scenes from sequences of images. These image sequences can be either video sequences or sets of discrete snapshots, and can be acquired with uncalibrated cameras.

The system leverages the characteristics of urban scenery (parallel and orthogonal lines) to help it reconstruct the geometry of the scene. The system computes the perspective projection for each camera location in the sequence using epipolar geometry [Fau96], refined by bundle block adjustment [MMC$^+$97] or epipolar line adjustment; this projective projection is determined up to an unknown projective transform. The system then uses parallel lines identified in the scene by the user to reproduce the affine geometry of the scene. Finally, the system uses manually determined ratios of lengths and angles in the scene to produce the Euclidean geometry of the scene, to within a global scale factor.

The advantages of this system are that it requires no fiducial points or camera calibration, and it produces camera pose during the process of reconstructing the geometry of the scene. However, the system (in keeping with its design goal of interactivity) requires extensive human input to determine key pieces of information about the scene. As with the Paris bridges work, this is unsatisfactory for very large sets of images.

Also, neither this system nor the two previous systems which recover pose can produce globally accurate poses. All pose measurements recovered from an image sequence by these systems are only relative to other measurements in the same image sequence. This makes it tedious to tie together pose-image data acquired at different times or locations into a single global dataset.

However, these systems (except for Move-X) only require very simple data acquisition equipment: just a video camera. While the Argus acquisition device and other pose-image acquisition devices try to do a lot of work at the front end of a system by collecting pose data and tagging images with complex sensors and subsystems, the pose recovery systems surveyed above push the work to the back end of the system. Of course, the data acquired by Argus is not perfect, and also requires processing at the back end of the system. The Argus data must be refined through similar techniques to those used in the pose recovery systems [CMT98].

Future advances in navigation technology, such as miniaturized GPS receivers and inertial sensors, could enable instrumentation like that on Argus to be built into a hand-held video camera, making pose-image data collection as simple as videotaping a scene.

Chapter 3

Design Overview

We have built the Argus pose-image acquisition device with the main goal of making the pose-image data acquisition process more efficient and accurate, through automation and sophisticated sensors. This allows us to capture pose-image datasets that are larger and more extensive in scope than ever before. This chapter presents the guiding principles of the design of Argus (§3.1), which were adopted to help us meet our goal of streamlining pose-image acquisition. We then give an overview of the main components and interconnections in the Argus design (§3.2), and conclude with a discussion of the software used for Argus (§3.3).

3.1 Design Requirements

Argus was designed to meet several explicit requirements, which are fundamental in meeting our goal of more efficient and accurate pose-image acquisition. These crucial requirements are discussed below.

3.1.1 Digital Data Collection

All the images must be collected in a digital form, without intermediate processing steps. If we did not collect digital images, we would have to scan pictures into our system. It is infeasible to scan thousands of images into a database and expect to be able to repeatedly and accurately acquire data this way. Scanning is too time consuming and error prone, in terms of the manual effort required. For example, our manual data collection effort acquired approximately 4,000 images. If it takes five minutes to scan each image, it would have taken continuous scanning for five to six days simply to get the images into digital form, not including the time needed to correct mistakes and other problems. Since the City project will eventually acquire a dataset that is at least two orders of magnitude larger, scanning images is clearly not an option.

3.1.2 Automatic Pose Measurement

Pose should be measured completely automatically by a navigation system. That is, acquiring the pose data should not require any manual measuring or calculations during the acquisition process. As we learned from our manual dataset collection effort, this manual effort consumes the largest amount of time spent acquiring the dataset. Reducing the pose acquisition time will drastically reduce the total dataset acquisition time. Also, measuring pose data is the part of pose-image acquisition most prone to error; automatic pose measurement reduces errors, increases accuracy, and enables more repeatable measurements.

3.1.3 Automatic Image Acquisition

The image acquisition and storage, including camera motion and configuration, should also be fully automatic. Just as in collecting pose data, automation eliminates most chances for operator error, and ensures that images are acquired under repeatable conditions. Automating the image collection also enables us to automatically record all the camera configuration data, such as aperture and shutter time. Ideally, we would also like to be able to automatically adjust image acquisition parameters, such as camera shutter time and focus. This would allow us to acquire images adaptively throughout an acquisition session. If, for example, images are too dark, we can increase shutter time or change the camera aperture to make images lighter (unfortunately, our current digital camera does not support this capability, as discussed in §5.1.1).

3.1.4 Human Scale Acquisition

The goal of the City project is to reconstruct urban environments. This introduces the design requirement that any pose-image acquisition device must be able to collect data in areas built on the human scale. That is, we need to be able to acquire data in areas designed for people to occupy and move around. Although many urban areas are barren and forbidding to people, many interesting urban areas also include people-oriented areas like plazas, fountains, parks, courtyards, and sidewalks.

Since these areas are designed for access by people, we must design Argus on the human scale so that it is able to fit and acquire data in these areas. Practically, this means that the physical configuration of Argus must allow it to pass through gates and arches. Also, Argus must be able to navigate wheelchair ramps, curb cuts, and other paths designed for people. Argus must also be maneuverable by a single person, to make data acquisition more practical. Finally, we need to be able to store Argus in our laboratory; this means that Argus must be able to pass through doors and fit into elevators.

These are sufficient constraints on the physical configuration of Argus, as most urban surroundings, including the MIT campus and the area inhabited by our laboratory, provide access via facilities such as wheelchair ramps, sidewalks, and curb cuts. Argus can use these facilities, providing us with maximum access to our surroundings for data acquisition.

3.1.5 Commodity Parts

Our final design goal has been to use commercial, off-the-shelf (COTS) parts whenever possible. We have tried to avoid building custom hardware and investing resources and effort in inflexible components. We have also tried to buy and use standard components in our system wherever possible. This reduces the work necessary to build Argus and makes it easier to replace parts, or reconfigure parts of the design as Argus evolves.

3.1.6 Summary

Our aim, with the above requirements, has been to "remove the operator from the loop." We feel that if there is less work for the operator of the Argus platform, then the data acquisition process will be faster and smoother. Also, if the operator can concentrate less on the mechanics of acquiring the data, then she can concentrate more on the quality of the data being acquired: e.g. what parts of structures or the environment are being imaged by the camera.

By using commercially available parts and systems, and by building Argus to a human scale, we have aimed to make Argus a practical device for pose-image

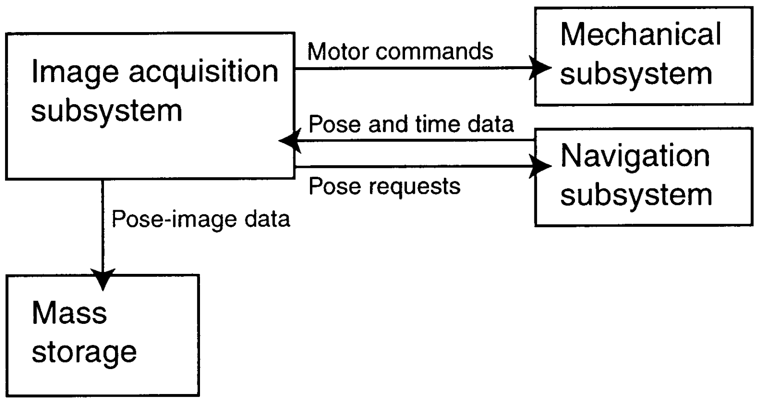

Figure 3-1: Argus subsystems. The power and control infrastructure links the image, navigation, and mechanical subsystems.

acquisition. It is not necessary to build a device that uses extremely exotic hardware, or a device that requires an armada of assistants and operators to guide it through its functions. Instead, we have striven to design Argus so that one person can acquire a pose-image dataset with a mild amount of effort.

3.2 Subsystems

The requirements discussed above naturally partition our design into four main areas or subsystems: the image acquisition subsystem, the navigation subsystem, which acquires pose data; the mechanical subsystems and infrastructure of Argus; and the rest of the power and data interconnect infrastructure. Figure 3-1 depicts these subsystems, and shows how they are connected.

The image acquisition subsystem (Chapter 5) is responsible for taking pictures with a digital camera, and sending those pictures to mass storage on Argus. The image acquisition subsystem also serves as the control subsystem for Argus, as it directs the mechanical subsystems so that the camera moves to the right locations for each picture, and queries the navigation system for pose data to tag each digital image.

The navigation subsystem (Chapter 6) is an integrated navigation system that uses a Kalman filter to combine inputs from multiple sensors and produce pose estimates. The Argus navigation system inputs include GPS, inertial, and wheel counter sensors. The navigation system's duties are to maintain the global system time, record time-tagged data from the navigation sensors, and optionally process the navigation data in real time. The navigation hardware also includes a custom circuit board for interfacing to the inertial sensor and performing other miscellaneous functions, such as power and signal level conversion for the sensors.

In addition to simply providing pose data, the navigation system provides a synchronization service to the image acquisition system. The GPS receiver in the navigation system provides our system with a global clock. The navigation system uses this global clock to synchronize image acquisition, i.e. the camera shutter trigger, with the navigation data. Although the current system only takes pictures while the camera is stationary, and can therefore easily synchronize images and pose data, a more dynamic system must be careful about synchronizing pose data and image acquisition times, as mentioned in [HWE98].

Argus also includes a significant amount of mechanical infrastructure. In addition to the actual frame and wheels of Argus, there is a mechanized pan-tilt head, mechanized adjustable camera tower, and a three-point motorized leveling system.

Finally, there is an array of data and power wiring on Argus which must be managed. For example, the pan-tilt head contains the inertial and GPS navigation sensors, and the camera. This means that bulky cables run from the moving pan-tilt head to the rest of Argus, but the pan-tilt head must still move freely.

3.3 The Software Environment

One of our major design decisions was the choice of CPU and software environment. Although the main components of our system, the navigation and image acquisition subsystems, involve their own specialized hardware, the nexus of our system is the CPU and the various pieces of software running on it.

We chose to use a ruggedized PC, for the straightforward reasons of compatibility, easily obtainable parts and software, low cost, and high flexibility. We then decided to base our software around the Windows NT operating system for several reasons, despite an initial bias against and lack of knowledge about Windows NT. The first reason is that the camera which we chose to use requires software to process image data before images are available as color RGB image data; this software is only available for Windows NT or MacOS based computers. The software also includes device drivers for the camera, so by using Windows NT we were also able to avoid having to write a device driver. The second reason for choosing Windows NT is that the vendor who supplied us with the GPS and inertial sensor parts of our navigation system was also implementing software for their hardware in a Windows NT environment. Although we could have worked with the vendor to produce navigation system software that would run in any environment that we chose, by choosing the same environment as the vendor we have been able to stay involved in the vendor's development process, and have been able to update our software as the vendor improves their software.

Most of the control software exists in a multithreaded Win32 dynamically linked library (DLL). This library contains code to initialize and communicate with the motor and camera subsystems. The library also contains the code which executes the hemispherical acquisition sequences and logs image and pose data.

The navigation software is a separate program that runs in Hyperkernel, a

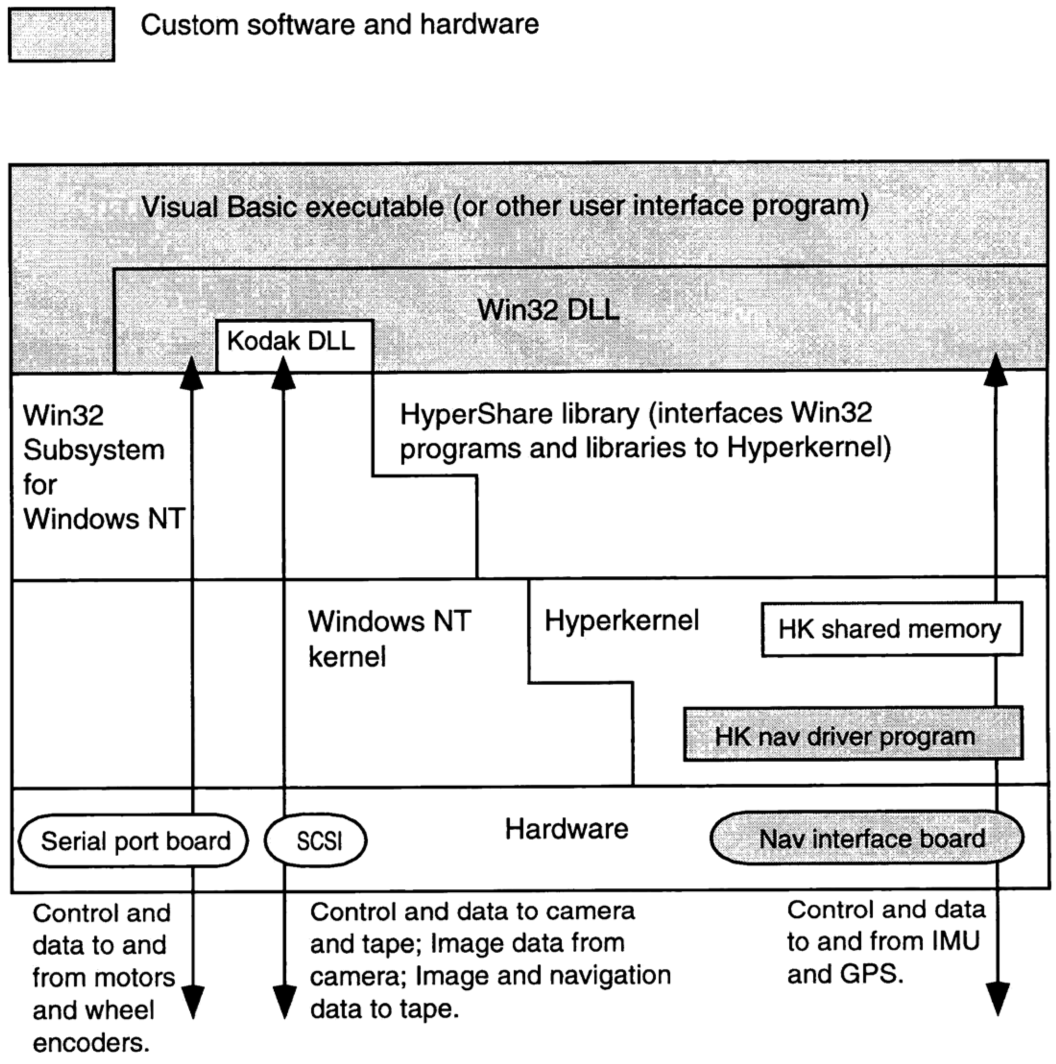
Custom software and hardware
Visual Basic executable (or other user interface program)
Win32 DLL
Kodak DLL
Win32 Subsystem for Windows NT
HyperShare library (interfaces Win32 programs and libraries to Hyperkernel)
Windows NT kernel
Hyperkernel
HK shared memory
HK nav driver program
Hardware
Serial port board
SCSI
Nav interface board
Control and data to and from motors and wheel encoders.
Control and data to camera and tape; Image data from camera; Image and navigation data to tape.
Control and data to and from IMU and GPS.

Figure 3-2: Data flow through the Argus software and hardware.

third-party pseudo-realtime subsystem for Windows NT. This program runs the Kalman filter, reads GPS data, and services the navigation interface board (§6.3) to collect inertial sensor data. The Win32 library logs data from the navigation program and sends camera trigger commands to the navigation program via the Hyperkernel shared memory mechanism.

Finally, a Visual Basic program provides a user interface for the whole system, calling the Win32 DLL to perform any requisite functions.

Figure 3-2 shows how data flows through the various software components.

Chapter 4

Mechanical Subsystems and Design

4.1 The Anatomy of Argus

A schematic model of Argus is shown in Figure 4-1. The physical design of Argus was developed in conjunction with Peace River Studios (PRS), who performed most of the physical construction described here. Argus is a cart with 3 wheels and a centrally mounted tower. The cart consists of a flat base, about 3 feet square. There are two pneumatic tires, 18 inches in diameter, mounted just inside the left and right edges of the base, offset towards the back of the base, and there is a single rotating caster wheel extended from the front of the base via two diagonal struts. This three-wheel arrangement allows the cart to be easily turned and maneuvered.

An extendable camera tower rises from the middle of the back edge of the base, and is supported by vertical struts which extend to the two back corners of the base, and by a vertical strut to the middle of the front edge of the base. The camera and its mechanized pan-tilt head are mounted on top of the extendable camera tower. The platform has handlebars attached to the camera tower, and a positive action braking system on the wheels (one must pull the brake lever to release the

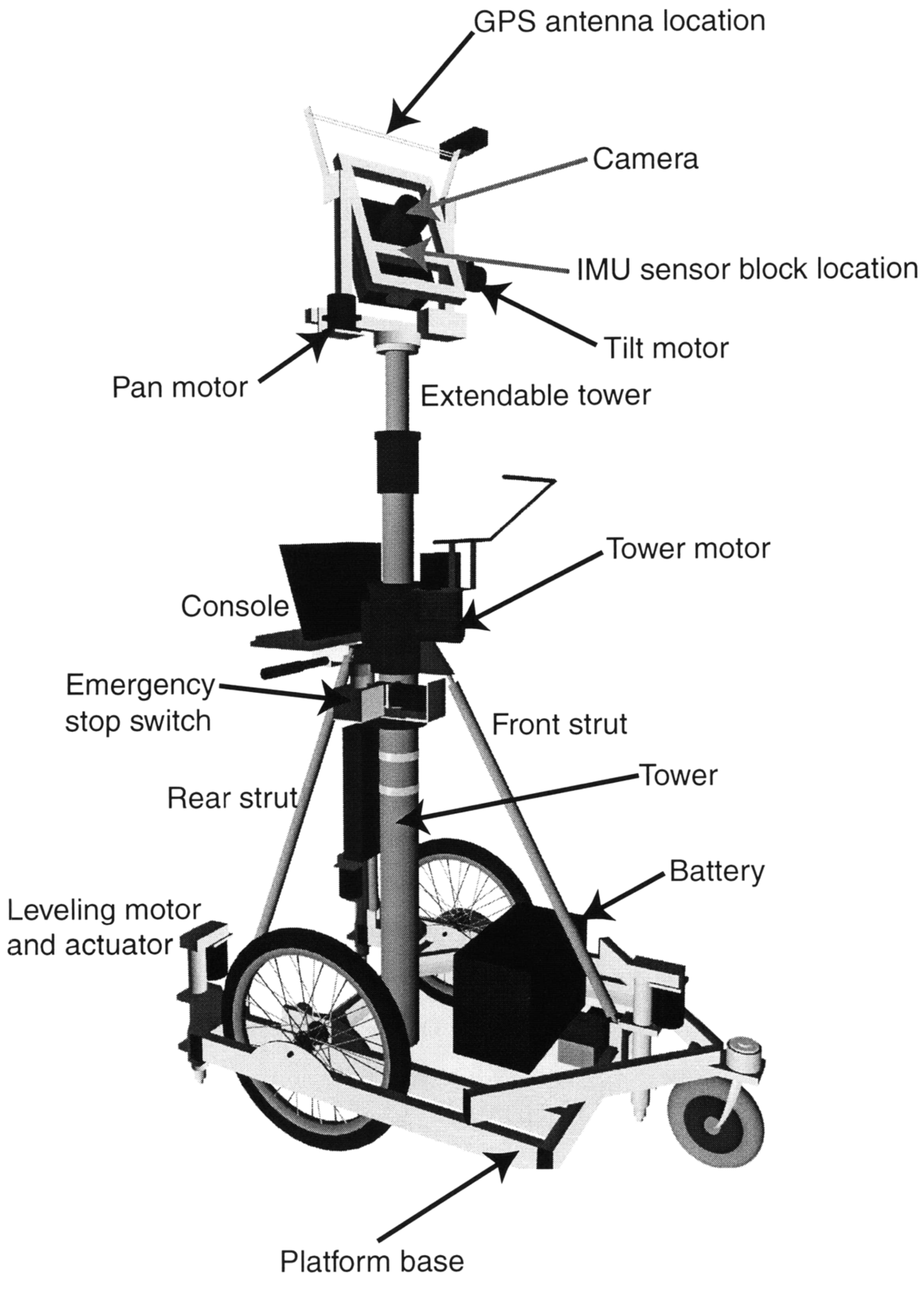

Figure 4-1: A model of Argus, with the tower slightly raised.

brakes from the wheels). The brake lever is mounted on the handlebars.

The CPU, battery, tape drive, and other power supply and data interconnection boxes are located on the base of the platform, in front of the camera tower. The weight of these components balances the large mass of the camera tower at the back of the cart's base. A small, light-weight LCD display is mounted just above the handlebars, and a compact keyboard is mounted on the handlebars.

Most of the platform is built out of prefabricated aluminum beams, joined with stock connectors. The camera tower is a small portable lighting tower, modified to fit onto the platform. There are a few custom machined parts, such as those that connect the camera tower to the base, and those that connect the front wheel to the platform, but otherwise all mechanical parts are stock parts available via a catalog or other means. The prefabricated construction gives us flexibility in mounting equipment to the platform, and allows us to reconfigure the cart's construction without too much expensive and time-consuming custom machining.

4.2 Leveling System

Argus is able to automatically level itself, using a two DOF level sensor mounted on the cart's base, and three linearly actuated ground supports. The ground supports are mounted in the three corner locations where the tower support struts are connected to the platform base, and when extended provide a rigid connection between the camera tower and the ground through the tower support struts (the cart's wheels leave the ground). This connection reduces the effect of mechanical play in the tower's construction on the pose-image acquisition process. Without the ground supports extended, the platform can sway significantly: the extendable camera tower and the pan-tilt head produce a high center of mass, and there is mechanical play between the cart base and wheels, and between the wheels and the ground (via the pneumatic tires). This swaying is more severe when the camera tower is raised.

The ground supports are also used to level the platform to help navigation system determine the camera pose. There is one less degree of freedom that has to be determined by the navigation system if the platform is level with respect to gravity.

When leveling, the linear actuators extend down from the base of the platform and attempt to level the platform according to the readings from the level sensor. Each support is actuated by a "Smart Motor," which is a servo motor with a built-in encoder and microprocessor; we use these flexible motors for many tasks on the platform, and they are discussed in detail in Appendix B. The front motor, which is directly connected to the level sensor, is the "master" motor. This master motor executes a program which implements the leveling algorithm described in Appendix C; the program reads tilt values from the level sensor and sends motion commands as necessary to the two "slave" motors mounted at the back corners of the cart. The program also commands the master motor.

The ground supports are extended and the platform is leveled before a sequence of images is captured at each node. When Argus is finished acquiring images at a node, the supports are retracted and Argus again rests on its wheels.

4.3 Pan-tilt Camera Mount

The camera's pan-tilt mount (Figure 4-2) is constructed from a smaller gauge of the same pre-fabricated aluminum bars that comprise the platform base. A rectangular frame supporting the camera and inertial sensor is supported by two horizontal rods at either side, supplying the pivot for the tilt axis. These rods in turn connect to a vertical enclosing U-shaped frame which connects to the camera tower via a large, heavy duty bearing, mounted horizontally. This bearing allows the U-shaped frame to rotate around the center of the camera tower, supplying the pan axis pivot.

The camera is mounted with its optical center roughly aligned with the intersection of the pan and tilt axes, so that the camera will pan and tilt without excessively translating the optical center.

The pan and tilt axes are independently driven by toothed belts from two Smart Motors via 30:1 reduction gearheads. The gearheads allow the motors to supply more torque to the pan-tilt mount, and prevent oscillations caused by the servo-motor control algorithms interacting with the slightly elastic belts.

The pan and tilt motors are each wired with separate limit switches for each direction of travel. These limit switches prevent the motors from driving the pan or tilt mechanisms past a full rotation. Any pan or tilt rotation that moves through more than a full circle could break or otherwise damage the cables and equipment attached to the pan-tilt head, such as the delicate custom manufactured cable that connects to our inertial navigation sensor. The limit switches also provide a repeatable reference point for calibrating the motion of the pan-tilt mount. Besides the pan and tilt limit switches, there is also one switch each for the pan and tilt axes which is activated when the pan-tilt mount is aligned at its "origin." We define this origin to be where the camera is pointing straight ahead at the front of the cart, and denote it as 0 degrees of pan and 0 degrees of tilt.

The motor encoders have a resolution of 2,000 counts per rotation. With the gear reduction, we can hypothetically position the pan-tilt head with a precision of 0.006 degrees. However, this precision is not available to us, due to mechanical

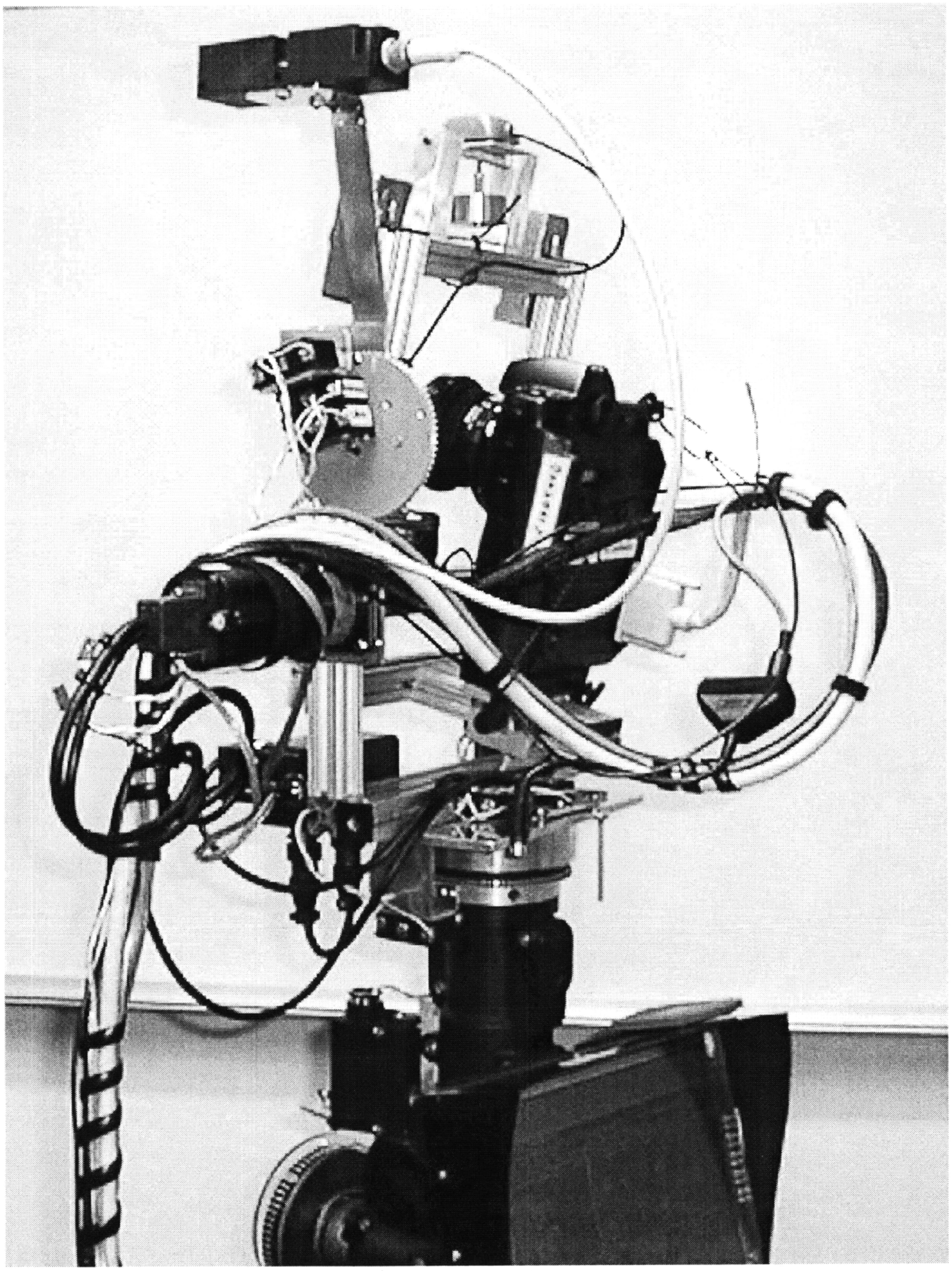

Figure 4-2: Argus pan-tilt head.

slippage, slack and backlash in the drive belts, and errors in the motor controller.

Finally, there is a significant amount of cabling that connects the moving pan-tilt head and the rest of the platform. This cabling includes the power and control lines for the camera, a substantial data cable connecting the camera to the CPU, a bundle of cables for the navigation sensors, as well as power and control wiring for the pan-tilt mount's motors. Cable management has been and continues to be a problem with the design of the pan-tilt head.

4.4 Motorized Camera Tower

Besides panning and tilting, the camera can also move vertically. The camera tower is a portable, extendable lighting tower which has been modified and attached to the base of Argus. The tower was originally designed to be raised and lowered using a hand crank and an internal cable mechanism, but we have replaced the hand crank with a toothed gear. Like the pan-tilt mount, the tower is also driven by a Smart Motor via a toothed belt. The motor driving the tower uses a 150:1 gear reduction because of the friction in the cable mechanism.

The tower motor also has two limit switches, one for each direction of travel. As with the pan-tilt mount, these switches prevent the motor from trying to drive the tower past its travel limits, and provide repeatable reference points for automatic motor calibration. As another safety precaution, the tower is equipped with an ultrasonic range sensor, which is mounted on the top of the pan-tilt head. This range sensor is wired in parallel with the upper limit switch, and prevents the tower motor from raising the tower into ceilings or other overhead objects (e.g. concrete building overhangs).

The moving camera tower makes it possible for Argus to automatically acquire images of objects at different heights, providing vertical stereo pairs. The moving tower also performs a very important practical function, as it allows us to capture images over the tops of cars, bushes, people, and other obstructions that might block the camera's field of view.

Chapter 5

Image Acquisition Subsystem

The Argus image acquisition subsystem comprises one sensor, the digital still camera (discussed in §5.1), and software. The software transfers image data to storage, interfaces with the navigation subsystem (to trigger the camera and tag images with pose data), and interfaces with the mechanical subsystems (to level the cart and pan, tilt, and raise the camera), as discussed in §5.2.

5.1 Camera

The camera is the most important part of Argus; a pose-image dataset is not very useful without good images. There are many options for acquiring images. However, given the requirements of §3.1, there are not many good options. We can either use an analog video camera, with a framegrabber to digitize frames, or we can use some sort of digital still camera.

As [TB96] discusses in detail, using an analog video camera with a framegrabber is problematic. There are problems with image quality, synchronization, and resolution. Analog video data is low resolution, yielding 640 by 480 pixels from a framegrabber. Synchronization is an even greater problem; because there is no control over exactly when a video camera acquires and transmits each frame, there is no way to exactly correlate image data with its associated pose data from the navigation system. Finally, there are problems with the way that most analog video cameras deliver their images as two interlaced fields, acquired at different times; for scenes with motion, these two fields may not be images of the same scene.

The other option is to use a completely digital camera. The problem with most digital cameras is that they do not actually capture true color. Video cameras typically use three separate charge coupled device arrays (CCDs) to capture the three components of color, as a CCD can only capture monochrome intensity images. However, using multiple CCDs introduces more problems with pixel alignment and registration, which is why many high resolution digital cameras only use a single CCD and produce monochrome images. Some high resolution digital still cameras use three separate color filters and a single CCD. They acquire three separate images of the same scene using a different filter for each picture; this requires that scenes are completely static, which is infeasible for Argus.

We use a Kodak DCS 420 professional digital camera, which uses a single CCD with 1,536 by 1,024 pixels. However, as discussed below (§5.1.1), each pixel in the CCD has an individual color filter, and full color RGB images are produced using an interpolation algorithm.

5.1.1 Camera Characteristics

The Kodak camera is not actually a complete camera; instead, it is a replacement camera back that is available fitted to either a Nikon N-90 camera body or a Canon EOS-1N body. We use the Nikon body version. The camera has a built in battery and uses any F-mount lens. The camera is essentially a regular SLR camera

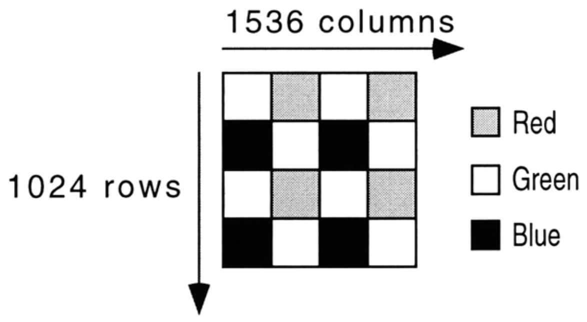

Figure 5-1: Details of the camera's CCD sensor pixel filter pattern.

with a digital CCD imager; all camera settings, most notably focus, aperture, and shutter time, must be set manually.

Sensor Details

Kodak marketing literature and product information claims that the camera produces images of 1,524 by 1,012 pixels with up to 36 bits of color information per pixel (12 bits per RGB color). However, the actual CCD sensor uses an array of pixels as shown in Figure 5-1. There are not actually 1,524 by 1,012 = 1,542,288 red, 1,542,288 green, and 1,542,288 blue pixels. Instead, there are only 1,536 by 1,024 pixels in total; one half of these pixels are green, one quarter are red, and one quarter are blue, as shown in Figure 5-1. Each 2 by 2 square of pixels has 2 green pixels, one blue pixel, and one red pixel. Only 8 bits of raw CCD data are read from the camera for each pixel, and software provided by Kodak is used to interpolate the raw CCD pixel data and produce a 1,524 by 1,012 pixel RGB image. The interpolation process (briefly sketched in [Eas95]) is non-trivial, and requires individual sensor response curves for each color.

Taking Pictures

The camera has a SCSI interface, and can be used either as a stand-alone camera or connected to a host computer. On Argus, the camera is permanently connected to the controlling CPU. The camera takes pictures when its shutter release button is depressed, or when commanded to take a picture over the SCSI interface. In addition, there is an external electrical interface to the shutter release button, the "remote release" line, which commands the camera to take a picture when grounded. We use the remote release line to trigger the camera on Argus, so we can precisely synchronize the camera's shutter action with the navigation system's pose data. The navigation system produces a 1 Hz clock signal generated by the GPS, which upon software command is used to trigger the camera shutter via the remote release line (§6.2.1).

The DCS 420 camera has an 8 Megabyte internal memory, and is equipped with a PCMCIA hard disk. In "burst mode" the camera can shoot up to 5 pictures at about 2 frames per second, storing each raw CCD image into its internal memory. The raw CCD images are each about 1.5 Megabytes of data. This "burst" frame rate is determined by the time required to read data from the CCD, as well as the mechanical shutter time of the camera body itself. However, in the steady state (acquiring up to 50 images in a row, as we do with Argus), the frame rate is determined by the speed at which the camera can shoot images and save them onto secondary storage. Either the camera must save each new image on its internal hard disk, or we must offload each new image over the SCSI bus, as the camera has to make room in memory for each new image. Image acquisition throughput is covered in Chapter 8.

Software Interface

The camera communicates with the host computer via low level commands sent over the SCSI bus. These commands, described in [Eas95], allow the host computer to take pictures with the camera, configure various parameters in the digital camera back (but not any settings for the camera body itself), and download data from the camera's memory. The host computer can also directly read and write data to a PCMCIA hard disk installed in the camera using the SCSI commands.

However, Kodak supplies a library [Eas97] for using the camera under computer control, in Windows NT or MacOS, without having to write SCSI level software. This library allows us to acquire images with the camera, download images to the computer over the SCSI bus, and perform the image interpolation needed to produce RGB image data from the camera's raw CCD data. The library also allows us to get the camera's raw CCD data, and the TIFF file format [Ald92] image data used internally by the camera. Acquiring the TIFF format data allows us to use the Kodak library to produce RGB image data from the raw TIFF data in an off-line post-processing phase. The camera also provides information about the camera back settings with each picture, as listed in Table 5.1. This settings information is available for each picture through the Kodak library. Note that this information does not include the camera's focus setting, although it does include the approximate focus distance as measured by the camera.

When the camera is under computer control, it can be used without a PCMCIA hard disk installed. In this configuration, each image acquired by the camera overwrites the last acquired image in the camera's memory; the host computer must explicitly download each image from the camera to avoid losing image data.

Camera Information
ISO setting
Shutter speed
Lens type
Exposure program
Focus mode
Focus distance
Aperture setting

Table 5.1: Camera settings information available from the camera for each picture.

Problems with the Camera

With respect to our design goals for Argus, there are some deficiencies with the camera. The most obvious deficiency is that the camera does not acquire true RGB data for each pixel, lowering the effective resolution of the camera. However, this lowered resolution has not been a large problem for us, and the camera data has been more than adequate for reconstruction [CT98].

The main problem with the camera is that although it provides information about key camera parameters, e.g. aperture, it does not support programmatic control of these camera parameters. Parameters such as aperture and focus must be set by manually operating the camera, contrary to our goal of being able to programmatically control all subsystems of the Argus platform. We currently manually adjust the camera's aperture to an appropriate average setting before acquiring each node, for better image quality. We hope to be able to modify the camera to allow setting the aperture automatically. The camera's focus is fixed at infinity for all pictures.

Finally, because the camera was designed to be used as a hand-held SLR camera, and not in a tethered application, it has been a challenge to obtain good performance (in terms of the total time required to acquire a node's worth of images) from the camera (Chapter 8).

5.2 Sequence Control Software

We use the term *sequence* to refer to all the actions taken by Argus in the course of acquiring a single node of pose-image data. This includes activating the automated leveling subsystem, moving the camera tower, panning and tilting the camera, and acquiring a single image at each pan and tilt. A sequence also includes any image storage tasks and optional online image interpolation or processing.

The set of tasks executed to perform a sequence is specified in a user-defined script file, which dictates the actions of the sequence control software. This script file is reread from disk before acquiring each node, so that Argus can be configured to carry out different sequence actions at different nodes. The sequence control software then sends commands to the various subsystems on Argus to perform the tasks described in the script.

A typical sequence script used on Argus to acquire a node of pose-image data for the City project performs the following actions:

1. Level.

2. Acquire a hemisphere of pose-images, logging each picture with its associated pose as it is acquired.

    - Acquire a ring of 12 pose-images, one at every 30 degrees of pan, at zero degrees of tilt.

    - Acquire a similar ring of images at 20 and 40 degrees of tilt.

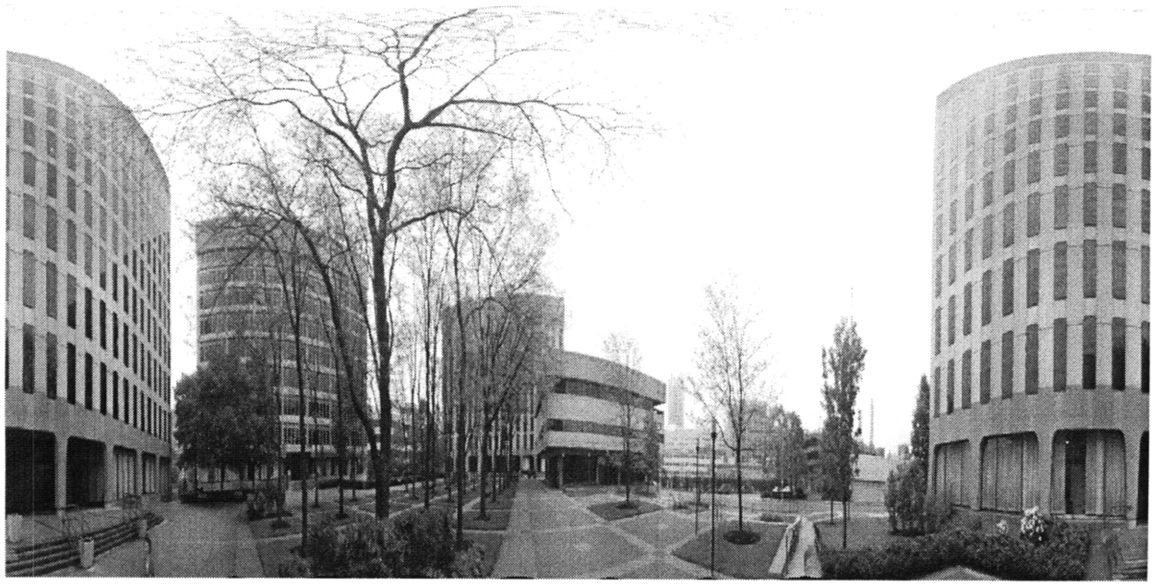

Figure 5-2: A spherical mosaic, using images acquired with Argus. The rectangular picture was produced by environment mapping the hemisphere onto a cylinder, and unrolling the cylinder. Mosaic courtesy of Neel Master and Satyan Coorg.

- Acquire a ring of 9 pose-images, one at every 40 degrees of pan, at 60 degrees of tilt.
- Acquire 2 pose-images at 80 degrees of tilt: one at 0 and one at 180 degrees of pan.

3. Raise the camera tower.
4. Acquire a hemisphere of pose-images as in step 2.
5. Lower the camera tower.
6. Retract the leveling actuators.

Figure 5-2 shows a spherical mosaic produced using images acquired by Argus, following the sequence described above.

One of the most crucial functions of the sequence control software is synchronizing image and navigation data to produce a coherent pose-image. As stated earlier, the camera's shutter is actually triggered by the navigation system, synchronized with a 1 Hz clock signal. The navigation system software runs as a completely separate program from the sequence control software, but provides a data and control channel via an interprocess communications (IPC) facility. The interface supplied by the navigation software over this channel allows the sequence software to request the navigation system to trigger the camera shutter; the navigation software replies to this request with the pose data (including timestamp) for the instant in time when the camera shutter was triggered. Thus to acquire a pose-image, the sequence control software sends a camera trigger request to the navigation system, reads the resulting pose data back from the navigation system, retrieves the new image from the camera, and records the pose and image data as a new pose-image on disk or tape.

Chapter 6

Navigation Subsystem

Argus obtains its pose data from an integrated navigation system, which uses GPS [Log92, HW94], inertial [Law93], and wheel counter data as inputs to a Kalman filter [Kal60, WB97] to obtain an estimate of the camera's pose throughout time. The navigation system was designed in collaboration with Korbin Systems Incorporated, which implemented and interfaced the navigation hardware and portions of the navigation software.

Figure 6-1 shows how the navigation system is organized. The navigation sensors interface to the Argus CPU via a custom interface board (§6.2). The sensor data is logged and processed by software running on the Argus CPU (§6.3), which also makes pose data available to the Argus sequence control software.

6.1 Navigation Sensors

6.1.1 GPS Sensor

We use the ONCORE miniature GPS [Mot96] manufactured by Motorola. This is a credit-card sized board that connects to an external GPS antenna and communi-

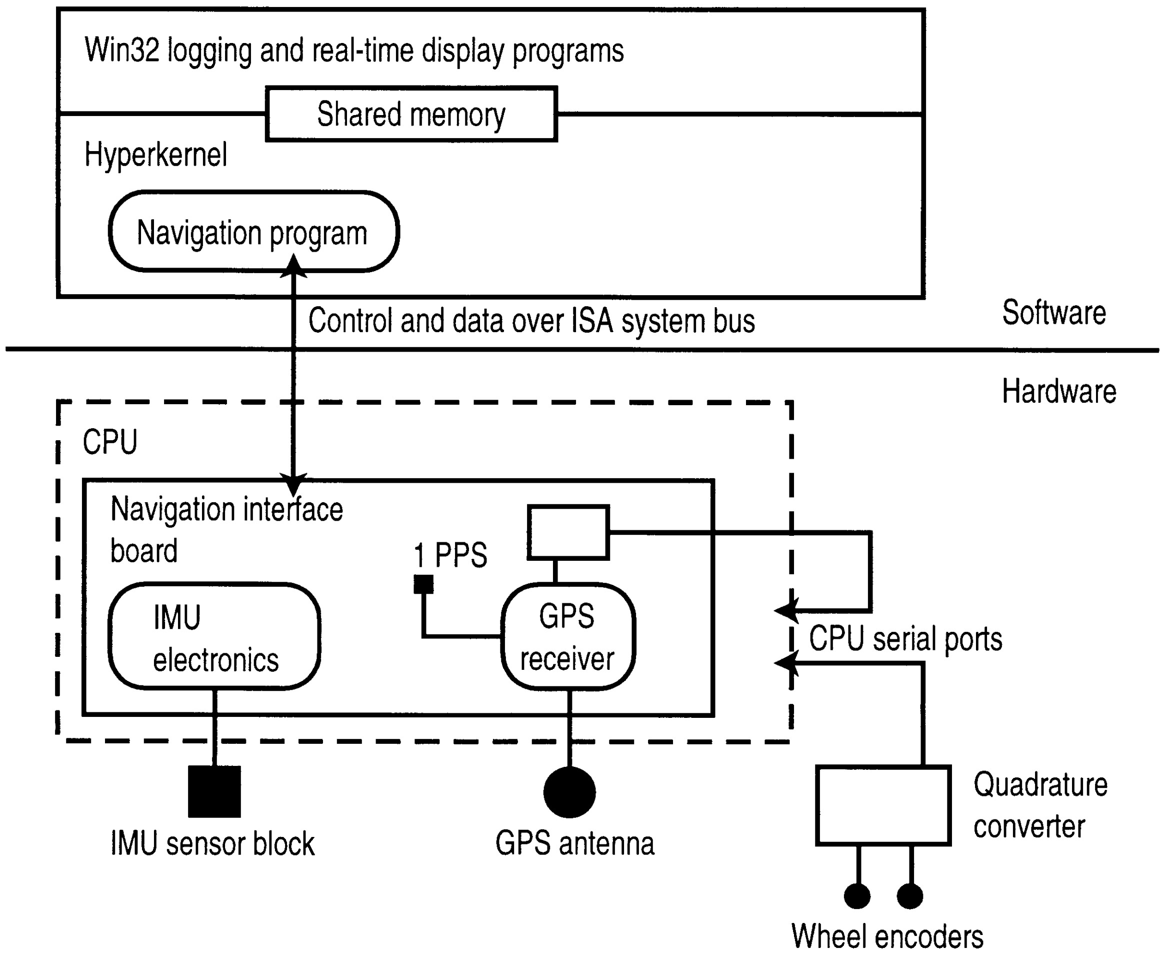

Figure 6-1: Navigation system organization.

cates control and data information over a TTL level serial data link. We use the VP model ONCORE receiver, which has differential capability, 1 Pulse Per Second (PPS) output, and carrier phase capability; these are explained further in Appendix A. The GPS receiver is mounted on the navigation interface board, which also supplies power to the receiver. The receiver provides data in the form of ASCII encoded "sentences," delivered at three Hz.

We use a GPS "base station" and post-processing techniques [Kai97] to improve the GPS data. The base station is a GPS receiver identical to the mobile receiver, located in a fixed position with good GPS reception, which receives and logs detailed GPS data. Post-processing software combines base station and mobile receiver data to improve the accuracy and precision of the final position data. There are techniques, such as kinematic GPS, that allow you to perform similar data processing in real time using a data link between the base station and the mobile receiver. We plan to eventually use kinematic GPS on Argus by incorporating radio modems to provide the necessary data link.

Appendix A includes a discussion of some of the issues involved in using GPS as a sensor on Argus, including accuracy and data quality.

6.1.2 Inertial Sensor

The inertial sensor is the second main input to the navigation system, and provides pointing and dead reckoning data. We use a Longbow model Inertial Measurement Unit (IMU) manufactured by GEC-Marconi. This unit is a strapdown device — the IMU remains fixed to its moving host platform, and does not rely on complicated gimbals to maintain a fixed reference frame. The Longbow IMU consists of an electronics package and a miniature sensor block, weighing about 8 oz. and measuring 2 inches by 2 inches by 1 inch (Figure 6-2).

The sensor reports rotation rate and acceleration along its 3 orthogonal axes; these measurements are used by the Kalman filter to provide angular position and to perform dead-reckoning calculations. The Longbow IMU was originally designed for use in missiles, and is built to military specifications (mil-spec). One characteristic of the mil-spec electronics package is that it nominally requires a +28 VDC power supply, along with the more common +5 VDC and +/−15 VDC power supply requirements. This +28 VDC supply is required to quickly "spin up" the sensor's gyroscopes to meet military requirements. However, we do not require a fast spin-up time, and omit the +28 VDC power supply.

Like the GPS receiver, the IMU electronics are mounted onto our custom ISA board. Because the IMU electronics package is mil-spec hardware, it is a large, robust board that uses many separate integrated circuits, and consumes the most space and thickness on our ISA interface board. This package provides data via a unique digital interface that requires an external clock signal; a "strobe" signal is sent to the IMU electronics to request data. The IMU provides new data at 110 Hz.

The IMU sensor block is rigidly mounted with the camera, immediately underneath the lens. There is a short cable with a hard to obtain mil-spec connector attached to the sensor block; we have a custom manufactured extension cable that connects the sensor block to the electronics package mounted in the Argus CPU. This extension cable also has a steel wire strain relief to prevent damage to the many small conductors inside the cable.

6.1.3 Wheel Encoders

The navigation system also uses two rotary encoders, one linked to each of the two large wheels at the back of the platform via toothed belts. These encoders produce a quadrature encoded output, which is fed into a microprocessor based quadrature to RS-232 serial interface. This interface tracks the encoder counts from the quadrature signal; these counts are read over one of the CPU's serial ports, recorded, and provided to the Kalman filter. The primary use of the encoders is to provide reliable short-term velocity data to the Kalman filter.

The encoder resolution, combined with the wheel diameter and the gear ratio between the wheels and the encoder, provides a resolution of about 3,200 encoder counts per meter of travel over the ground. This number varies with tire pressure, but the encoders can be calibrated by moving Argus over a known distance and recording the encoder counts. The actual accuracy of the wheel encoder data is degraded by changes in the wheel diameter over time (due to changes in tire pressure), wheel slip through turns or during straight motion, and discontinuities resulting from bumps or other irregularities in the ground.

6.2 Navigation Interface Board

The navigation system relies upon a custom interface board to provide power to the GPS and IMU sensors, and to provide various glue logic and circuitry. The board, shown in Figure 6-2 with the GPS and IMU sensors, is an ISA format PC expansion card that plugs into the Argus CPU's system bus. The miniature GPS receiver, the IMU electronics, and various buffers, timers, and power supplies are mounted on the board, which occupies two full length ISA slots due to the thickness of the IMU electronics package.

The interface board supplies power to the GPS receiver, and converts the receiver's TTL level serial data signal to an RS-232 level serial data signal. After signal level conversion, the serial signal is connected to one of the CPU's RS-232 serial ports, where it is logged and processed by the navigation software. The in-

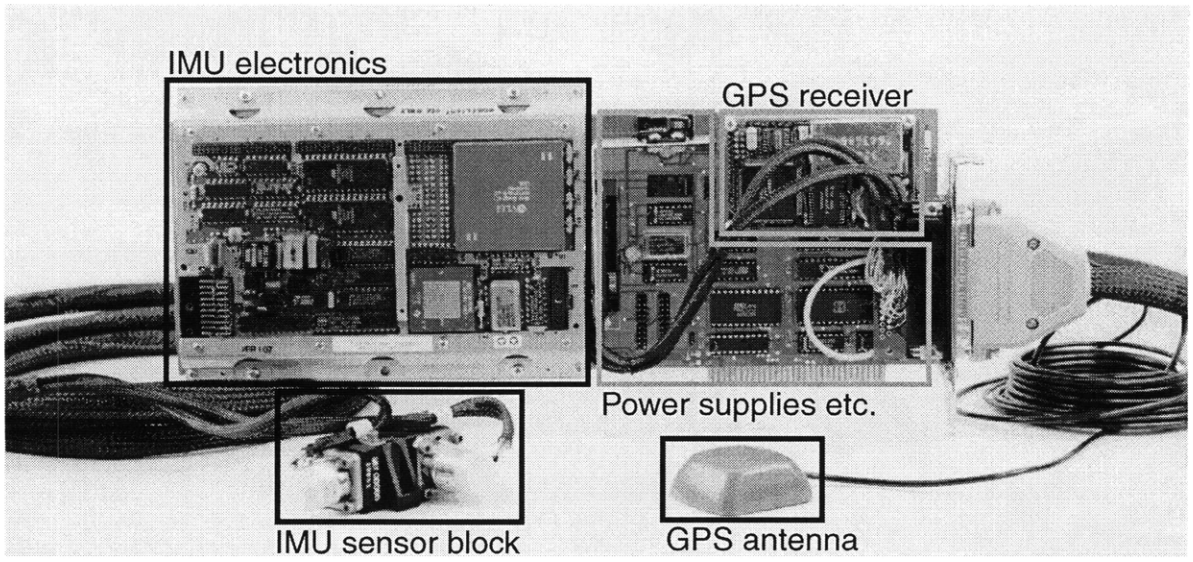

Figure 6-2: Navigation interface board, manufactured by Korbin Systems, GPS antenna, and IMU sensor block. Image courtesy of Korbin Systems.

terface board also includes counters, that are connected to the GPS receiver's 1 PPS output; these counters allow the navigation software to exactly record the current GPS epoch (the current second, according to the GPS).

The navigation system interface board also buffers the IMU data. Without buffering, the navigation software would have to reliably respond to interrupts at 110 Hz to avoid losing IMU data. Lost IMU data causes cumulative errors in the navigation solution, so it is crucial that no data is lost. The board triggers an interrupt when the amount of data in its buffers reaches a predetermined high-water mark, prompting the navigation software to read and reset the contents of the board's buffers.

6.2.1 Synchronizing Images to the Navigation System

The navigation interface board provides the interface which allows us to synchronize the camera shutter with the navigation data. We can trigger the camera shutter by electrically signalling the camera; this signal is activated when the GPS 1 PPS signal is received, while a latch on the interface board is also activated. That is, the navigation software sets up a latch on the board; when the next 1 PPS signal is received, the latch's output triggers the camera shutter. The GPS receiver returns a sentence containing the current position and time immediately after each 1 PPS signal, which was valid at the time of the 1 PPS signal. This GPS position and time is used to help determine the complete pose estimate at the time the camera was triggered, and the pose estimate is returned to the image acquisition subsystem.

6.3 Navigation Software

The navigation software is responsible for servicing and collecting data from the navigation sensors, running the Kalman filter (described in [Kai97]), and logging the navigation data. We use Hyperkernel [Ima97] as the software environment for our navigation system. Hyperkernel, a proprietary product from Imagination Systems Inc., is described and marketed as a real-time kernel that runs in conjunction with the Windows NT kernel. Programs running in the Hyperkernel subsystem can install interrupt handlers and access the CPU's hardware directly, avoiding the heavyweight and unpredictable Windows NT interrupt handling [Yat97, Rob98], and bypassing the Windows NT protection mechanisms, for easier interfacing to the navigation hardware. Hyperkernel also supports multi-threading, and allows IPC and shared memory between Hyperkernel threads and Windows NT threads. Finally, it is possible to build and develop programs for Hyperkernel using the standard Windows NT development tools (Microsoft Visual C++ 5.0), which is convenient. However, there is no debugger available for the Hyperkernel environment, and we must resort to the venerable "printf" style of debugging.

Our Hyperkernel program installs two interrupt handlers. The first handler is activated by the navigation interface board, and reads in IMU data from the board when its buffers start to fill. The second handler is activated by the serial port connected to the GPS receiver's data output, and reads GPS data messages. The main program runs two calculations. A dead-reckoning strapdown calculation runs whenever a new IMU data record is received; this calculation updates position information based on the accelerations and rotation rates from the IMU sensors, and does not use any other feedback. The second calculation is the Kalman filter update, which is performed when a GPS data message is received, combining the IMU and GPS data. Wheel encoder data is logged by a Win32 thread, and also passed to Hyperkernel via shared memory.

The navigation data is passed via shared memory to a Windows NT program for logging. The data can then be stored on disk or tape, and displayed using real-time plots. This real-time display is essential for monitoring the status of the navigation system, and allows the operator of the Argus platform to take corrective measures when the navigation system calculations begin to accumulate too much error. Such corrective measures include moving to a location with better GPS reception, or moving the platform in a circle to compensate for and recover from IMU problems.

Chapter 7

Power and Data Infrastructure

Argus contains a significant amount of hardware and cabling dedicated to supplying power to and routing data and control signals between the various subsystems of Argus. This chapter describes that infrastructure.

7.1 Power Connections

There are several different types of power required on Argus, but the primary source for power is a 12 VDC, 86 Amp-Hour gel-cell battery. This battery provides power directly to the CPU, which is equipped with a 12 VDC power supply. The CPU's power supply also supplies the console's LCD screen with power. The wheel encoders and the quadrature converter require +5 VDC, which is derived from the CPU's power supply.

The Smart Motors require an 18–48VDC power supply (they have an internal regulator for the microprocessor, encoders, and communication circuits), and work better at higher voltages [Ani96]. The motors receive power from the 12 VDC battery via two 40 volt DC-DC converters connected in parallel. These converters

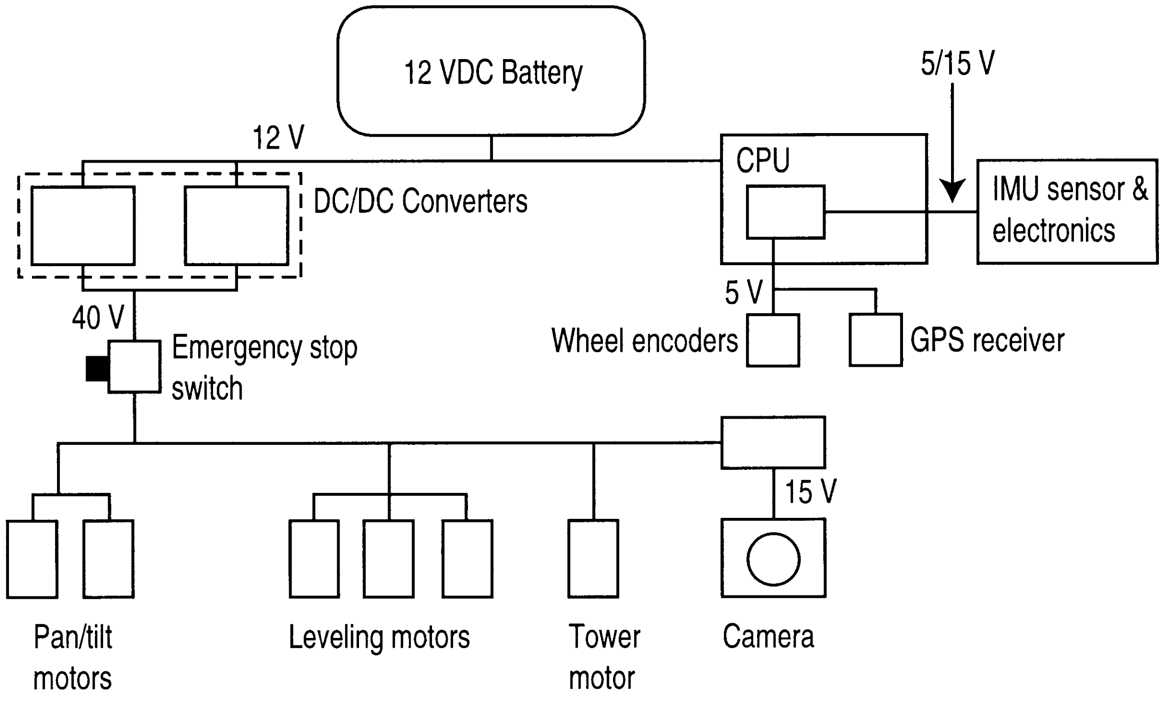

Figure 7-1: Power subsytem.

together provide up to 300 Watts of power to the motors. The motors are connected in parallel to the output of thae converters, via a "kill switch." The kill switch is a large, red, easily accessible emergency stop button, which can be used to immediately shut off the motor power in an emergency.

Finally, the digital camera requires a 15 VDC input through its battery charger connection. Although the camera has its own self-contained battery (which is not accessible without dismantling the camera case), this battery quickly drains when the camera is connected to a SCSI bus. We supply the camera with 15 VDC using a voltage converter from the 40 VDC motor power supply.

Figure 7-1 diagrams the power interconnect on Argus.

7.2 Data and Control Connections

Figure 7-2 shows how data and control signals are routed on Argus. The details of the various subsystems are described below.

Smart Motors The Smart Motors are all connected to the CPU via a multiport serial board. This board connects to a port concentrator box through a thick cable. The pan, tilt, and tower motors are each connected to a dedicated serial port on the concentrator, while the three leveling motors' serial data lines are connected in a daisy chain. The front leveling motor receives serial data from the CPU through the serial port concentrator; it echoes this data to one of the rear leveling motors, which then echoes the data to the other rear leveling motor. The data output of the last motor in the leveling subsystem daisy chain is connected to the data input of the CPU's serial port, and the CPU verifies that the motors received the right commands. In addition to simply echoing commands from the CPU, the front leveling motor injects new commands for the other motors into the daisy chain, in order to level Argus. The front leveling motor is also connected directly to a 2-axis tilt sensor via an analog to digital converter. The tower motor range sensor is wired in parallel with the tower motor's upper limit switch.

Wheel Encoders The two wheel encoders are connected to a microprocessor-based quadrature converter, which has an RS-232 interface. The converter is connected to a dedicated port on the serial port concentrator.

Camera Data and Control The digital camera sends data to and receives commands from the CPU via a SCSI interface. A 10 foot long cable connects the camera to the CPU. An additional small cable connects to the camera's multi-purpose

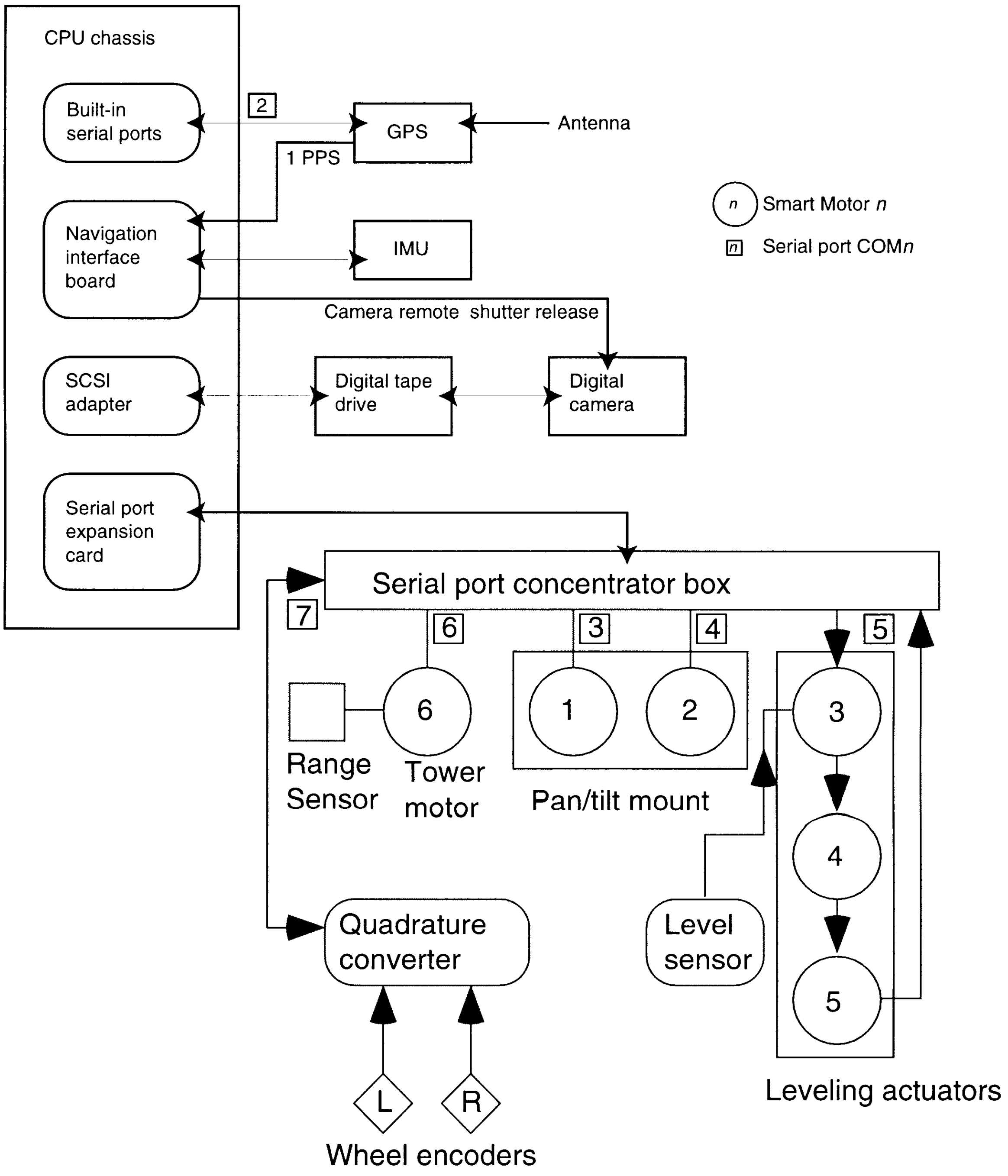

Figure 7-2: Data and control interconnect.

connector. The cable provides external power and the remote shutter release signal to the camera. The other end of this small cable is connected to the navigation interface board and the camera power supply.

Navigation Sensors The IMU, mounted in front of the camera, is also connected to the navigation interface board via a 10 foot long cable. As this cable contains a number of delicate conductors, it also contains a strain relief wire that is anchored to the IMU at one end of the cable, and to the connector at the other end of the cable. The GPS receiver's serial data output is connected to the CPU via a signal level converter on the navigation interface board, and the receiver's 1 PPS output is connected to the interface board, as described in §6.2.

Storage Tape and disk storage is connected to the CPU via a SCSI interface.

Chapter 8

Node Acquisition Throughput

Image acquisition throughput is a main criteria for evaluating the success of Argus. As mentioned before, our goal in constructing Argus is to enable speedy and accurate pose-image acquisition. The Argus image acquisition subsystem must provide good throughput to meet this goal. This chapter analyzes the throughput of the image acquisition subsystem, concentrating on the bottleneck introduced by our digital camera interface. We present results for the total node acquisition time that show that we can acquire each hemispherical node of pose-images in under 6 minutes (not including leveling and tower movement time), providing a major speed-up over manual pose-image acquisition. We also present benchmarks characterizing the camera's image download performance.

8.1 Acquiring a Node

When acquiring a node of pose-images, a sequence of several actions is performed by Argus and the operator of Argus. As discussed in §5.2, these actions include leveling Argus, panning and tilting the camera, taking pictures with the camera and downloading them, moving the camera tower, and wheeling Argus around between acquisition locations. For the purpose of looking at the image acquisition throughput of Argus, we will ignore the tower movement and leveling actions carried out by Argus. That is, we will consider these times fixed, as the movement times of the tower and leveling subsystem are limited by the safe and reliable operating speeds of their motors and associated machinery. We will also ignore the time taken to move Argus around between locations; this time is at least as fast as the time needed to manually move and set up a tripod and the other equipment required for manual pose-image acquisition. Instead, we will concentrate on the time required to take pictures with the camera, download them to the computer, and move between pan and tilt positions while acquiring a hemispherical mosaic.

Acquiring each individual pose-image requires four steps:

1. Command the camera to shoot a new picture, and read the camera pose at that time.
2. Download the new picture from the camera.
3. Store the picture together with its pose to disk or tape.
4. Move to the next pan and tilt position.

A naive implementation of the image acquisition subsystem would carry out each of these steps sequentially before acquiring the next image, to satisfy the ordering constraints that must be obeyed in implementing the image acquisition subsystem. The shooting, downloading, and picture storage actions must be carried out in sequence. We cannot store a picture until we have downloaded it, and we cannot download a picture until we have told the camera to shoot it. Also, we cannot shoot a picture with the camera until we have moved to the correct pan and tilt location.

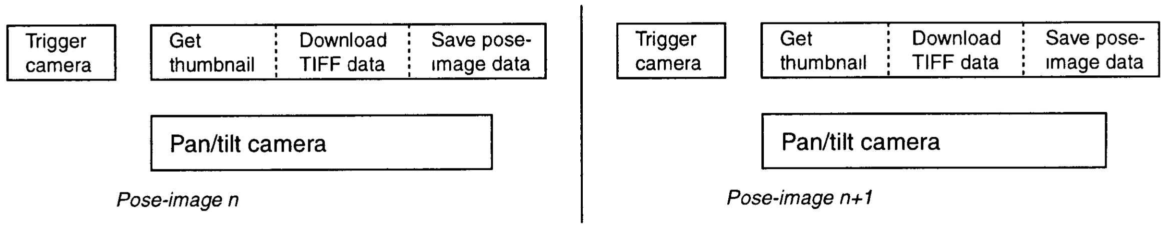

Figure 8-1: Overlapping actions in pose-image acquisition.

Action	Avg	Std dev	Min	Max	Total
Trigger camera	0.7	0.2	0.5	1.0	32.8
Pan/tilt camera	4.7	2.5	3.2	12.7	227.9
Acquire image from camera	6.5	2.2	5.1	13.8	307.8
		Total sequence time			365.2

Table 8.1: Times for acquiring a full node, which includes acquiring 47 images and 48 pan/tilt moves. This sequence followed is the same as that described in §5.2, but without leveling and moving the camera tower. Note that the total sequence time is much less than the sum of the trigger, pan/tilt, and image acquire times. Acquiring the image from the camera includes downloading the thumbnail, downloading the image, and saving all pose-image data to disk. Times are in seconds, and all data was saved to disk.

However, there is at least one major opportunity for parallelism: the only time that the camera has to remain still is when we actually shoot the picture. We can overlap the picture download and storage with the pan and tilt movements, as shown in Figure 8-1. Table 8.1 shows the time taken to perform pan-tilt moves, command the camera to shoot a picture, and download and store the resulting images for a representative hemispherical acquisition sequence. The times show that by overlapping moving and image acquisition, we reduce our total sequence acquisition time from about 10 minutes to just over 6 minutes, a 40% improvement over the non-overlapped time.

Table 8.2 shows statistics gathered for several sequences in a row. This data

Avg	Std dev	Min	Max
346.4	11.4	333.5	374.1

Table 8.2: Average sequence time. Statistics were gathered for 12 sequential hemispherical acquisition sequences, performing the same actions as described in §5.2, except for leveling and moving the camera tower. Times are in seconds, and all data was saved to disk.

shows that we can on average acquire a hemispherical node of images in less than 6 minutes. Tower movement times are on the order of an additional 10 to 30 seconds per node, and leveling times are on the order of an additional minute per node. Thus by using Argus we can acquire images far faster than manual pose-image acquisition. This is especially so when we consider the time that is required to manually level and align the camera tripod during manual acquisition.

8.2 Image Acquisition Performance

The average times listed in Table 8.1 show that in general, the image acquisition time between successive camera shots is greater than the time taken to move between two successive camera locations, by a few seconds. In fact, acquiring images from the camera always takes longer than the pan and tilt moves, except for the six long switchback moves at the end of each row in this particular sequence. Thus the total sequence time is limited by the image acquisition time for each pose-image. Table 8.3 shows timing statistics for the image acquisition only, broken down by the individual steps required to acquire an image. The largest components of this time are the TIFF image file download, and the thumbnail download and display.

We attempted to optimize the image acquisition time by omitting the thumbnail download and display step. However, this was not succesful. Because we use the

Step	**Avg**	**Std dev**	**Min**	**Max**	**Total**
Download and display thumbnail	2.7	0.6	1.5	3.3	127.7
Download TIFF file	2.9	0.3	2.5	3.8	134.5
Save TIFF and pose data	0.7	0.1	0.6	0.9	32.7
	Total image acquisition time				294.9

Table 8.3: Times for acquiring images in a node. Times are for 47 sequential image acquisitions in a single node. Times are in seconds, and all data was saved to disk.

camera in a mode where it only stores images in its memory, without writing them to its internal disk, the camera will not take a new picture until the last picture taken is downloaded from the camera. But the Kodak library functions that we use to acquire the TIFF data for each image are not the same functions normally used to download image data from the camera, and the camera is not always able to take a new picture unless we explicitly perform a thumbnail or RGB format download of the last picture.

Fortunately, the thumbnail display provides a valuable diagnostic function. The Argus operator can monitor thumbnails to ensure that the camera settings (exposure and focus) are correct. The thumbnails also assure us that the sequence control software is downloading and displaying the correct pictures (although the sequence control software does check the image timestamps).

Also, the Kodak library uses various caching strategies. Thus when the library downloads the thumbnail data, it also caches data that speeds up the following TIFF file data download. Downloading just the TIFF data for each image functions incorrectly, and is only marginally faster than downloading thumbnail data as well as the TIFF data.

Table 8.3 shows that it takes about 0.7 seconds to save the data for each pose-image to disk. As there are about 1.6 Megabytes of data in each pose-image, the image acquisition software is only achieving 2.3 Megabytes per second of effective disk-write bandwidth. This is disappointing, as the SCSI II standard is meant to provide data transfer rates of up to 10 Megabytes per second. We performed a disk benchmark test on the Argus CPU using the Adaptec SCSIbench program. Using a block size of 64 Kilobytes, the SCSIbench program reported approximate disk read bandwidths of 2.3 MB/sec for random reads, 7.3 MB/sec for sequential reads, and 3.7 MB/sec for same sector reads from our Seagate ST34371W disk connected to an Adaptec 2940 SCSI II interface. Thus the poor pose-image disk write performance is most likely due to the Windows NT filesystem (NTFS) performing non-sequential writes. When using our Quantum DLT4000 tape drive, we obtain 4–5 MB/sec of pose-image write bandwidth, almost fully utilizing the tape drive's maximum bandwidth of 5 MB/sec. Argus uses the tape drive to acquire pose-image datasets.

8.3 Camera Benchmarks

To better understand the limitations of the camera, we performed a benchmark test with the camera. Using a benchmark program (i.e. not the regular Argus software) we downloaded several distinct images from the camera, in each of the four formats supported by the Kodak interface library, on several CPU configurations. We performed the camera benchmark tests on the Argus CPU, a 133 MHz Pentium CPU with 64 MB of memory, in the following configurations: without any other software running, with the Hyperkernel navigation software running without the Kalman filter, and with the Hyperkernel navigation software running with the Kalman filter. The Kalman filter computations are normally carried out in real-time, and involve intensive calculations. We also performed the camera benchmark tests on a 200 MHz Pentium Pro machine with 64 MB of memory, and on a dual processor 300 MHz Pentium II machine with 256 MB of memory. However, the camera benchmark test was not multithreaded, and did not take advantage of both processors on the dual processor machine. All CPU configurations used an Adaptec 2940 SCSI II interface card.

As Table 8.4 shows, the thumbnail picture is the fastest picture to download from the camera, on any CPU configuration. This is unsurprising as the thumbnail image format contains two orders of magnitude less data than the other image formats. The RGB image format is the slowest format to download; the Kodak library performs image processing on the raw CCD data received from the camera to produce the RGB data. This extensive processing is apparent when comparing the benchmark performance of the different CPU configurations. The dual Pentium II machine downloaded and processed data into the RGB format over 5 times as fast as the Argus CPU with the navigation software and Kalman filter running in the background, a result of the Pentium II machine's much faster processor.

The TIFF format download was faster than the raw CCD format download across all CPU configurations. This is at first surprising, because in addition to the same data as the CCD data format, the TIFF format also contains thumbnail data and header information. It seems as if the TIFF data download should take at least as long as the raw CCD data download. However, the methods for downloading each of these formats are quite different, and this affects the download times.

The Kodak library implements the functions OpenPicture and GetPicture, which are used to get the actual pixel data for various pixel formats, including thumbnail, RGB, and raw CCD data, but not including TIFF format data. To get thumbnail, RGB, or raw CCD data for a picture, an application must first call Open-

CPU	Image type	Min	Max	Avg	Norm
Dual Pentium II 300MHz	thumbnail	0.45	0.45	0.45	0.22
256 MB memory	ccd	2.01	2.01	2.01	1.00
	tiff	1.50	1.57	1.53	0.76
	rgb	4.57	4.59	4.58	2.28
Pentium Pro 200 MHz	thumbnail	0.50	0.52	0.51	0.25
64 MB memory	ccd	2.35	2.37	2.36	1.17
	tiff	1.87	1.95	1.90	0.95
	rgb	6.33	6.34	6.34	3.15
Pentium 133 MHz	thumbnail	0.74	0.75	0.75	0.37
64 MB memory	ccd	2.76	2.77	2.76	1.37
(Argus CPU)	tiff	1.96	2.11	2.01	1.00
	rgb	13.87	14.13	13.96	6.95
Pentium 133 MHz	thumbnail	0.86	0.93	0.88	0.44
64 MB memory	ccd	3.11	3.16	3.13	1.56
(Argus CPU)	tiff	2.13	2.60	2.45	1.22
PlatformHK, no KF	rgb	17.20	18.34	17.52	8.72
Pentium 133 MHz	thumbnail	1.14	1.16	1.15	0.57
64 MB memory	ccd	3.95	4.01	3.97	1.98
(Argus CPU)	tiff	2.51	2.85	2.66	1.32
PlatformHK, with KF	rgb	25.86	27.48	26.21	13.04

Table 8.4: Image acquisition times for the Kodak DCS420 camera on various systems, using version 1.1.0 of the Kodak library. All times are in seconds, and each test was repeated 5 times. "Norm" is the average time for each test normalized by the average time for the TIFF file acquire on Argus, without any Hyperkernel software running. PlatformHK indicates that the Hyperkernel navigation software was running, and KF indicates whether or not the Kalman filter was activated.

Picture. This causes the Kodak library to initialize various internal data structures, and download header information about the picture from the camera's memory, where it is stored in a TIFF file format. The application then calls GetPicture to get the actual picture data, causing the Kodak library to download the rest of the picture data from the camera, performing image processing on the data as necessary.

To obtain TIFF format data from the camera, an application program calls the library's CopyImageToFolder function. This library function downloads all the TIFF data for an image from the camera and stores it in a specified disk file; the application can then load the TIFF data into memory from that disk file. Note that the Kodak library does not support a memory to memory transfer of the TIFF data. Nevertheless, even with an extra disk read and write, an application can get all the TIFF data into memory faster than it can obtain just the raw CCD data, as obtaining the TIFF data requires only one transfer from the camera, and has no internal library initialization overhead.

The Argus control software uses the TIFF format download to get images from the camera.

Chapter 9

Pose Measurement Accuracy

Accurate pose, like image throughput, is a crucial requirement which Argus must meet if we are to consider it successful. To evaluate the pose accuracy of Argus, we performed two simple experiments to look at the position and attitude performance of the Argus navigation system. This chapter describes those experiments, and discusses the results, which show that the navigation system currently delivers acceptable attitude measurements, reporting attitude within 1–2 degrees. However, the position measurements delivered by the navigation system are not currently good enough to be used for automatic 3D reconstruction from pose-images: the navigation system reports position changes of up to several meters while Argus is stationary.

9.1 Accuracy Between Nodes

This experiment was designed to give a feel for the application level, or "real-world" performance of the Argus navigation system, across several nodes, or acquisition locations. We do not concentrate on the detailed statistics and perfor-

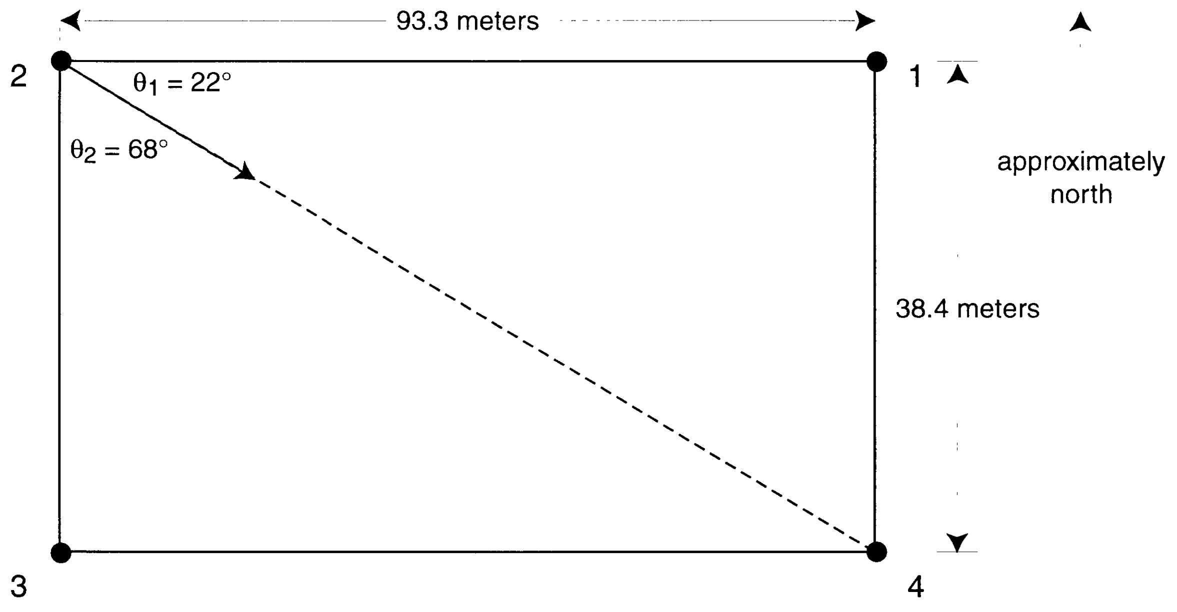

Figure 9-1: Navigation experiment layout.

mance of the internals of the navigation system, but instead look at the quality of the final pose data reported by the navigation system. Not only is the navigation system still undergoing substantial development, but an in-depth discussion of navigation performance details is beyond the scope of this thesis.

We chose four points around the edges of the quadrangle at the center of Technology Square (Figure 1-1). These points formed a rectangle of 93 by 38 meters (Figure 9-1). After taking Argus outside and initializing the navigation system, we moved Argus to each of the four points in turn. At each point we sighted the pan-tilt head to the other three points around the rectangle, using the camera. We visited the points in order, and finished by revisiting the first point and repeating our initial sights.

The camera focusing screen is etched with a small circle in its center; we aligned the camera by visually centering this small circle on a marker stick held vertically at the point we were sighting towards. The camera was never tilted in the pan-tilt head, only panned. Whenever we moved Argus, the camera was locked straight forward (to keep the camera and the wheel encoders in the same reference frame for the Kalman filter). For each sight, the camera was left pointing at the sight for at least ten seconds. After ten seconds, the time of day (as reported by the GPS receiver) was recorded.

After taking the sights, we post-processed the mobile and base-station GPS data, IMU data, and wheel encoder data using the Kalman filter. The recorded times were used to extract the processed pose data for each sighting.

9.1.1 Results

The processed heading (yaw) data reported for each sight is shown in Table 9.1, and the position data is shown in Table 9.2. We only look at the heading data, as we were not able to establish ground truth for the tilt (pitch) and roll of the camera. However, the quality of the heading data is a good indicator of the overall attitude data quality, as heading data typically has the worst error characteristics of the three degrees of freedom in attitude [Kai97]. The first heading sight is treated as true, and ideal headings for the other sights are derived from this first sighting using the geometry of the sights.

As Table 9.1 shows, the heading measurements are typically very accurate, with most measurements within 1 or 2 degrees of the expected heading. The Kalman filter's one-σ certainty measurement is within 0.5 degrees for most of the heading measurements, indicating a high confidence in the heading estimate. These results are within the accuracy of the experiment; the process of measuring out the rectangle and positioning and aiming Argus at each point could have introduced a similar magnitude of errors.

Table 9.2 shows that the position measurements are relatively poor. The one-σ

Sighting	Time	Meas	Ideal	Err	One-σ
1→2	13:20:34	-98.7	–	–	0.83
1→3	13:23:40	-121.0	-120.7	0.3	0.85
1→4	13:24:50	-173.2	171.3	15.5	0.85
2→1	13:27:30	81.8	81.3	0.5	0.44
2→4	13:28:20	102.5	103.3	0.8	0.44
2→3	13:29:30	171.8	171.3	0.5	0.45
3→2	13:32:40	-8.4	-8.7	0.3	0.44
3→1	13:34:30	60.5	59.3	1.2	0.45
3→4	13:35:10	81.6	81.3	0.3	0.46
4→3	13:38:10	-97.5	-98.7	1.2	0.42
4→2	13:38:50	-76.5	-76.7	0.1	0.42
4→1	13:40:10	-7.5	-8.7	1.2	0.43
1→2	13:42:30	-96.9	-98.7	1.8	0.37
1→3	13:43:20	-118.2	-120.7	2.5	0.38
1→4	13:44:30	173.3	171.3	2.0	0.41

Table 9.1: Inter-node heading accuracy. All numbers are in degrees, mod +/– 180, except that "Time" is the HH:MM:SS time of day of the sighting, as reported by the GPS receiver. "Meas" is the heading reported by the Kalman filter, while "Ideal" is the calculated heading based on the first sighting and the geometry of the sighting marks. "Err" is the difference between the ideal and measured headings, and "one-σ" is the standard deviation of the heading estimate at each time, as reported by the Kalman filter.

Loc	Time	Measured N	Measured E	Measured D	One-σ
1	13:20:34	4.7	-25.2	-5.5	6.6
1	13:23:40	8.6	-28.4	5.3	5.6
1	13:24:50	8.8	-27.9	5.6	5.6
2	13:27:30	7.6	-87.6	9.6	4.0
2	13:28:20	8.4	-90.2	10.6	3.9
2	13:29:30	9.5	-88.1	20.6	3.8
3	13:32:40	-19.3	-78.5	13.0	3.8
3	13:34:30	-20.5	-82.2	14.4	3.8
3	13:35:10	-21.4	-82.4	13.3	3.7
4	13:38:10	-13.8	-16.7	15.0	3.3
4	13:38:50	-16.5	-16.8	13.9	3.3
4	13:40:10	-20.7	-23.4	9.6	3.3
1	13:42:30	10.4	-26.0	10.6	3.1
1	13:43:20	12.4	-22.2	14.0	3.0
1	13:44:30	13.5	-16.4	18.7	2.8

Table 9.2: Position accuracy. All measurements are in meters, except for the time, which is in HH:MM:SS format. Measurements are expressed in north-east-down coordinates, relative to the GPS base station on the roof of our office builfing (building NE43). "One-σ" is the standard deviation of the position estimate at each time, as reported by the Kalman filter.

value for each measurement is on the order of a few meters. Also, for measurements taken at the same location but at different times, the reported coordinates changed by about 2 to 7 meters. Finally, the variance in the height measurements, a range of about 23 meters, is very large; all the measurements were taken while on a fairly level concrete surface. What is promising, however, is the monotonically decreasing one-σ value produced by the Kalman filter; this indicates that the filter measurements and estimates are getting better over time.

In general the poor positioning performance can be directly attributed to poor GPS performance. Jumpy GPS data causes the Kalman filter's position estimate to jump around also. Several factors contribute to the GPS errors. First, we performed the test in a worst case environment for GPS performance: the Technology Square quadrangle is surrounded on all four sides by 9-story concrete office buildings, which contribute to multipath problems (Appendix A). Second, the buildings also block many GPS satellite signals from reaching the mobile receiver at all. At the beginning of the test the mobile receiver could only reliably receive signals from 3 satellites; throughout the rest of the test the receiver could reliably receive signals from 4 satellites, with 5 or 6 satellites' signals available intermittently. As a minimum of 4 satellite signals are required to produce a 3 DOF position fix, the GPS position fix was never determined very robustly, even with differential corrections. Finally, we suspect that there may be problems with the way that we are configuring the GPS receiver and recording data, as the GPS position, even with differential corrections applied, is extremely poor.

9.2 Accuracy Within a Node

This experiment was intended to examine the pose accuracy of Argus within a single node. That is, it was intended to examine the relative angle accuracies reported by Argus for each camera position within the same node.

We started the Argus navigation system in a location next to Technology Square with good GPS signal reception. We then wheeled Argus into the Technology Square quadrangle and acquired a few nodes of images with Argus. After post-processing the navigation data as in the previous experiment, we extracted the reported attitude data for a single representative node. We used the camera trigger timestamp of each image in the node to index into the post-processed navigation data. Images were acquired in the same locations as described by the sequence in §5.2.

9.2.1 Results

Table 9.3 summarizes the reported attitudes for each camera position. The roll at each camera position should be negligible, as Argus was levelled and the camera only pans and tilts; the pitch should be 0, 20, 40, 60, and 80 degrees for rows one through five respectively. The headings for each position should progress in thirty degree increments for rows one through three, forty degree increments for row four, and 180 degrees for row five. In Table 9.3, the error column shows the heading error for each position. This error was calculated by treating the world heading of the first camera position as true; for each following camera position we calculated the ideal world heading according to the moves determined by the sequence, and subtracted this ideal heading from the reported heading to produce the error figure.

	Pitch	Roll	Yaw	Error
Row 1	0.4	-1.0	-157.6	0
	0.8	-0.7	-128.4	-0.8
	0.9	-0.3	-98.3	-0.7
	0.8	0.1	-68.4	-0.8
	0.5	0.5	-38.5	-0.8
	0.1	0.6	-8.4	-0.8
	-0.3	0.6	21.5	-0.9
	-0.6	0.4	52.1	-0.3
	-0.8	-0.1	81.2	-1.2
	-0.8	-0.6	110.8	-1.6
	-0.6	-0.9	140.5	-1.9
	-0.1	-1.2	170.3	-2.1
Row 2	20.5	-1.5	-158.2	-0.6
	20.9	-1.2	-129.1	-1.5
	21.0	-0.7	-98.8	-1.2
	20.9	-0.2	-68.6	-2.0
	20.5	0.1	-38.6	-2.0
	20.1	0.3	-8.4	-0.8
	19.7	0.2	21.4	-1.0
	19.4	-0.1	51.2	-1.2
	19.1	-0.5	80.7	-1.7
	19.2	-1.0	110.3	-2.1
	19.4	-1.4	139.9	-2.5
	19.9	-1.7	169.7	-2.7

	Pitch	Roll	Yaw	Error
Row 3	39.9	-2.2	-159.1	-1.5
	40.3	-1.8	-129.8	-2.2
	40.5	-1.2	-99.3	-1.7
	40.3	-0.7	-69.0	-1.4
	40.0	-0.3	-38.8	-1.2
	39.7	-0.1	-8.7	-1.1
	39.2	-0.2	21.1	-1.3
	38.9	-0.5	50.7	-1.7
	38.7	-1.1	79.9	-2.5
	38.7	-1.7	109.4	-3.0
	39.0	-2.2	138.7	-3.7
	39.4	-2.6	168.2	-4.2
Row 4	59.6	-4.3	-161.4	-3.8
	60.3	-3.6	-121.8	-4.2
	60.5	-2.1	-80.6	-3.0
	60.2	-0.8	-39.5	-1.9
	59.6	-0.2	1.0	-1.4
	58.9	-0.5	40.6	-1.8
	58.4	-1.8	80.1	-2.3
	58.5	-3.1	117.8	-4.6
	58.7	-4.4	156.4	-6.0
Row 5	79.7	-13.3	-170.3	-12.7
	79.6	-1.5	20.0	-2.4

Table 9.3: Intra-node heading accuracy. All numbers are in degrees. Yaw is the heading, or pan value; pitch is the tilt value. The Kalman filter's one-σ value was less than 0.6 degrees for all the attitude measurements reported here. Error is the heading error for each camera location, as described in the text.

As the table shows, the reported roll for each position is within one or two degrees except for at larger tilt angles, where the reported roll climbs rapidly. This is a result of both alignment errors in the pan-tilt head as well as true measurement errors. The IMU axes were not completely aligned with the pan-tilt axes. Also, at low tilt angles roll is easily measured against gravity, but at high tilt angles the roll measurement is more error prone and subject to accelerometer and gyroscope bias errors.

The reported pitch was mostly within one degree of the commanded camera position, except that at higher tilts the reported pitch was within one and a half degrees of the camera position. Note that for the first pictures in each row, the reported pitch is slightly high, while for the last pictures in each row, the reported pitch is slightly low. This is because the camera tower was tilted slightly to right during this experiment; this tilt was apparent upon visual inspection of Argus. Although the Argus base was levelled, the camera tower was not exactly aligned to the base. For the first half of the pictures taken in each row, the camera points to the left of the pan-tilt head center, and thus tilts up slightly; for the second half of the pictures the camera points to the right of the pan-tilt head center, and thus tilts down slightly.

Neither the pitch nor roll measurements reflect any significant sensor drift over the six or seven minutes required to acquire the node.

The heading measurements reported by the navigation system are more complicated to interpret. As the error column in Table 9.3 shows, the error magnitudes gradually increase as the camera pans across each row; however, the magnitudes also increase across all the rows. There are multiple explanations for these errors, including sensor error and mechanical error, or misalignment.

The gradual climb in error magnitudes across the whole node could be explained by drift in the heading gyroscope. Because Argus remained still during this test, there were no velocity measurements which could be used to compensate for the gyroscope drift. However, the errors are not completely attributable to the gyroscope drift, as the error magnitude is "reset" at the beginning of each row; that is, the heading error for the first picture in each row was less than the heading error for the last picture in the previous row. Another possible explanation for the generally increasing heading error across the rows in the sequence is that the IMU's heading readings are more subject to error at higher pitches, as none of the IMU's gyroscopes are in line with the pan axis when the camera is tilted. However, this explanation does not account for the increasing errors within each row of camera positions, which is most likely due to mechanical errors such as those described below.

The camera mount is driven by belts on toothed wheels. Because the belts are not infinitely tight, the belt driven system will always suffer from slack and backlash. One explanation for the increasing magnitude of negative error as the camera pans through each row is that some of the pan motor motion is being absorbed by slack in the drive belt on each pan move. Thus on each pan move the camera is being panned just a little less than it should be, resulting in more negative positioning error after each move. The large relative error between the first and second camera position in each row can be explained by backlash, as the camera pans to the left just before the first picture in each row, and then pans to the right for each following picture in the row. Alignment errors in the pan-tilt head could also account for increasing heading errors as the camera tilts up. Finally, the Smart Motors which drive the camera head are not infallible; their movements are possibly subject to internal controller error.

Overall, this experiment shows that there are many possible sources for pose error on Argus; we have listed a few of them here. However, the resulting attitude data is quite good, with an average heading error of less than two degrees for each camera position. This accuracy is sufficient to perform intra-node pose optimization in software using the images from the camera [CMT98], to produce the final pose estimates used for reconstruction. Finally, much of what we have termed "error" in this analysis is not error at all, but a measurement of the actual position of the camera and IMU on Argus. A manual pose-image acquisition effort would never be able to measure and record these small perturbations of the camera from its commanded position. However, by using our integrated navigation system, we get feedback about the actual position of the camera, not just the assumed position of the camera.

Chapter 10

Future Work

The Argus pose-image acquisition device is an extremely complicated set of equipment and mechanisms, with a rich set of sensors. As such, it offers a myriad of opportunities for future work. This work can be divided into two categories: near term, essential work should be completed to meet the immediate requirements of the City project, and less essential but interesting work can be carried out over a longer term.

10.1 Essential Work

We can divide the essential near-term work into three main groups: improving the image acquisition subsystem, improving the navigation subsystem, and acquiring the first large-scale dataset.

10.1.1 Image Acquisition Subsystem Improvements

We can improve the image acquisition subsystem in two main ways: we can increase the speed at which it acquires images, and we can increase the resolution of the data which it acquires. Image acquisition speed is currently limited by image download from the camera and the Kodak interface library. However, there are some techniques which may improve image download throughput from the camera.

Increasing Image Throughput

First, it would be useful to experiment with caching all the images for a node on the camera's local PCMCIA disk. This would save the image download time from each picture in the node. Of course, disk space is limited, and we would have to download each node's pictures from the camera between nodes. But this download of about 50 pictures can be overlapped with moving Argus between node locations and performing leveling operations. The camera benchmark data (Table 8.4) and disk write times measured on the Argus CPU indicate that it would take from three to five minutes to download all the image data for a node from the camera's local disk and save it to the Argus CPU's disk or tape.

Second, we can replace the Kodak interface library with a custom SCSI command level driver for the camera, which would enable us to avoid the overhead experienced in adapting the functions of the Kodak library to our task. The library was designed to be used in a situation where the camera is used in the field, and images are then downloaded later, so acquisition time is not a concern. One example of where a new SCSI level driver could help us is in caching the images locally on the camera disk and then downloading them between nodes. The camera supports commands which allow direct access to its local PCMCIA disk. A custom SCSI driver could access all the images on this local disk with a single request for the 50 or so images in a node. This large data transfer may utilize the SCSI bus between the camera and the CPU more effectively, speeding up the download.

Using a custom SCSI driver, we would also be able to download TIFF data directly into application memory, instead of being forced to read the TIFF data from the CPU's disk, where it is copied by the Kodak library. Given the disk throughput that we experience on Argus (2–3 MB/sec, or 0.7 seconds per image), we could save up to 1.4 seconds from the acquisition time for each image by avoiding the intermediate disk write and read, which is a savings of about one minute per node.

Finally, we can slightly increase the image acquisition time of Argus by changing the order in which images are acquired within each hemispherical node. We currently acquire images in several rows, panning the camera from left to right throughout each row. However, this entails a large pan movement from the extreme right pan position to the extreme left pan position. We could instead acquire adjacent rows by moving in opposite directions, and thereby avoid the long pan movement back to the start of each row. This would save about twenty to thirty seconds per node.

Increasing Image Resolution

There are only a few ways that we could immediately increase the image resolution. The first, and easiest method, is to simply replace the current camera with a higher resolution model. Kodak manufactures a DCS 460 camera, which is identical to the DCS 420 used on Argus, except that the DCS 460 has a higher resolution imager (3060 by 2036 pixels), and a wider field of view. This camera should be software compatible with the DCS 420; however, it does produce images that are four times as large as those produced by the DCS 420, and will probably slow down image acquisition by almost a factor of four.

It may also be possible to increase image resolution by supersampling with the camera. That is, acquire images that overlap each other by large amounts, but are slightly offset, e.g. by a fraction of a pixel. Then by sampling from all the images that cover a given area, we should be able to reconstruct a higher resolution of that area. Again, this would slow down image acquisition time by some constant factor. This technique could also allow lower resolution, but faster cameras to be used.

10.1.2 Navigation System Work

There is a significant amount of work to be done with the navigation system on Argus. This work covers three main areas: tuning, verification and alignment, and bringing all the navigation computation online.

Tuning the Navigation System

Our current navigation system, as described in [Kai97], was originally designed with high speed, continuously moving applications in mind, such as airplanes and cars. Argus, on the other hand, is often stationary. This means that the relatively jumpy GPS measurements received by the navigation system are not always properly corrected by the inertial navigation system when Argus is not moving. We must tune the Kalman filter, the heart of the navigation system, to have better characteristics for our largely stationary application.

The filter must also be modified to take better advantage of all the inputs available to it: the wheel encoders, the leveling system's level sensor, and other feedback about what Argus is actually doing. For example, the current filter does not currently use the information that Argus is not translating (moving horizontally or vertically) when we acquire a node. This is an important input which can help the filter determine the error characteristics of its sensors. We can also input the commanded pan and tilt motor movements into the filter; if the filter does not measure the expected movement, it can signal an error condition. This is important, as redundant measurements and control inputs can be used to verify each other.

Aligning and Verifying the Navigation System

Once the navigation system is properly tuned, taking the fullest advantage of all its various inputs, it will be important to align and calibrate the system. As the navigation system accuracy increases, calibrating the system become even harder. To get highly accurate attitude data, we need to properly align the the camera to the IMU's reference frame, and align the IMU's reference frame to the earth. Also, the wheel encoders and level sensor must be properly calibrated.

We will also need to perform a complete evaluation of the navigation system accuracy. This will entail accurately determining the ground truth against which to evaluate the navigation system, and then measuring ground truth points using the navigation system on Argus. Determining the ground truth will itself be a challenging task, especially as we will be trying to determine the accuracy of GPS measurements with a precision on the order of centimeters.

Online High Accuracy Pose Computation

We currently use GPS base station data to do kinematic GPS data post-processing after acquiring pose-images with Argus. This high-accuracy GPS data is then "re-played" through the Kalman filter with other sensor data that was logged during an acquisition, to produce high accuracy pose-estimates. We have not yet implemented an online version of this processing, but a high accuracy on-line Kalman filter with kinematic GPS processing would be extremely useful for several reasons. First, real-time kinematic GPS processing would be better able to alert the Argus operator of poor GPS data and other anomalies which the operator may correct. Second, having high-accuracy pose data available online would enable real-time visualizations and other computations that can help the operator of Argus acquire a better pose-image dataset. For example, we could use high-accuracy pose data to determine the coverage of a dataset. If a dataset does not cover an area densely enough, the operator can be prompted to move Argus to the appropriate locations to improve the quality of the dataset.

10.1.3 Scaling to Large Datasets

We need to acquire a truly large-scale dataset with Argus. We should be able to acquire a dataset covering our office park in about a day. The real challenge will be to acquire datasets covering the entire MIT campus. These datasets will push Argus in several areas: speed, data storage, navigation drift, and mission planning. Pose-image data covering the MIT campus could require up to 10,000 nodes, or 500,000 individual pose-images. It would take about 100 days to acquire this much pose-image data, using Argus for 10 hours each day in its current configuration, which can acquire about 10 nodes per hour; faster pose-image acquisition would be extremely advantageous. A pose-image acquisition effort of this size would produce on the order of 1 Terabyte of data, whereas the Argus tape drive can only store about 20–40 Gigabytes, or 400–500 nodes of pose-images. A large data acquisition will also tax the back-end components of the entire City project: networking and online storage. Acquiring data over a relatively long period of time (several days) and a relatively large area will highlight any problems in the repeatability of navigation data. And of course, as with any large or ambitious project, we will have to carefully plan our data acquisition sessions, including a plan of attack and a methodology for choosing sites at which to acquire hemispherical nodes.

10.2 Long Term Work

Long term work that can be carried out on Argus could include exploring different paradigms for collecting pose-image data, and work focused online reconstruction techniques. The current paradigm that we use for acquiring pose-images with Argus is one of "punctuated acquisition" — we move Argus to key locations, where we remain stationary while we acquire a hemispherical node of pose-images. Not only do we stop moving between nodes, but we also stop moving between images within a node.

With a fast enough camera, and a sufficiently short shutter time, we could sweep out each hemisphere of images with a single continuous pan and tilt motion. The camera would quickly snap pictures at key locations, just as it does now. However, node acquisition would be much faster. This continuous motion paradigm naturally extends to the next level: we could also try to continuously collect images while Argus is being moved around an area, without having to stop to acquire each node. It would be a challenge to properly design and implement a system to continuously acquire pose-image, as we wish to acquire an extremely dense set of pose-images covering the environment around us. We would have to acquire images at high speed, with extremely good synchronization between the camera and the navigation system.

On-line reconstruction algorithms would also provide a ripe area for exploration based on the Argus infrastructure. Installing a more powerful CPU on Argus would allow incremental reconstruction algorithms to run, which could take low-resolution pose-images as their input. We can imagine a system in which the Argus operator "paints" the 3D geometry of the real world with Argus, while examining the reconstructed geometry on an interactive display. The reconstruction display can highlight areas in the scene where more data is needed, and the operator can position Argus to acquire data in these areas. The display could also even indicate to the operator exactly how to move Argus. This is perhaps one of the most exciting areas for future work with Argus.

Chapter 11

Conclusion

The preceding chapters have presented the design of and selected implementation details of the Argus pose-image acquisition device. Argus, in its current form, meets most of the important goals set for it.

Argus has proven easy to maneuver and convenient to handle; its physical design allows us to freely move Argus on sidewalks, ramps, and other open spaces. Overall, Argus is also quite easy and simple to operate: to acquire pose-images, we only have to switch Argus on, run the appropriate program, and push a few buttons when Argus is at the right locations. The automated systems on Argus run smoothly and robustly, and eliminate tiresome and error-prone tasks for the operator. Also, the navigation system delivers global attitude estimates that are at least as good as those which are acquired manually.

The positioning accuracy of Argus, however, does not yet meet the requirements that Argus was built to satisfy. The required positioning accuracy will come as improvements are made to the navigation system, and as the navigation system's inputs are widened so that it uses all the information available to it. Despite the poor position estimates, the Argus software has been flexible and convenient for implementing and evaluating changes to the navigation system.

The digital camera has not fully met our expectations. Some effort is required to obtain good images with camera, as there is no way to automatically adjust the camera's aperture for each image. A manually determined and configured average aperture setting must suffice each of the 40–50 images at acquired at each node. Nevertheless, the camera provides high quality images at a respectable speed, and is the key component in the Argus image acquisition system; this system provides a significant improvement in speed and performance over acquiring images manually.

Argus will allow us to quickly and painlessly collect large pose-image datasets of our environment. These datasets will be valuable for the City project's research into 3D reconstructions, and will also be valuable in their own right.

Appendix A

GPS Information

A.1 GPS Receiver Capabilities

The GPS receiver that we use has several capabilities which are necessary for our navigation system, beyond the basic GPS receiver functionality.

Differential capability means that the receiver can use real-time corrections to the GPS transmissions, which enable the receiver to calculate an improved position fix. The United States Coast Guard has installed a series of stations to transmit these corrections along U.S. waterways; these corrections are also available through commercial services. The corrections are typically transmitted by radio, although some services also use pager network data links. We have mounted a base station GPS receiver on the roof of our laboratory's building. This base station will allow us to produce more accurate differential corrections than those transmitted by the Coast Guard and others.

1 PPS output is a GPS receiver electrical output that is synchronized to global GPS time. It is a pulse whose leading edge indicates the start of a new "GPS second." We use this pulse to help timestamp our images and synchronize image acquisition with navigation system pose estimates.

Carrier phase capability enables the GPS receiver to determine the phase of the GPS carrier signal that the receiver is observing. Under favorable circumstances carrier phase capability allows position to be determined to within tolerances that are smaller than the wavelength of the GPS signal (19 cm), by using post-processing techniques. There are techniques, such as kinematic GPS, that allow you to perform similar carrier phase processing in real time, using a data link between the base station and the mobile receiver. We plan to eventually use kinematic GPS on Argus, by using radio modems to provide a data link to the base station.

A.2 GPS Performance

GPS technology, used as initially designed and conceived, can provide positioning accuracy across the world to within 25 meters. However, this accuracy can be affected by signal degradation, blocking, and other interference. Also, the position of the GPS satellites in the sky (known as the satellite geometry) can affect GPS performance; certain geometric configurations provide more robust and accurate position estimates.

Another factor that heavily influences GPS performance is the U.S. military's interference with the GPS signal. This is known as "Selective Availability" (SA). The U.S. Department of Defense, which operates the GPS satellites, reserves the right to degrade GPS performance in the interests of national security, and normally does. When SA is activated GPS position accuracy is reduced to about 100 meters.

GPS Technique	Potential Position Accuracy
Standard (SA on)	100 meters
Standard (SA off)	25 meters
Differential corrections	1–5 meters
Post-processing, kinematic, and carrier phase techniques	1 mm–100 cm

Table A.1: Potential positioning accuracy of different GPS techniques. Data from [Mot96, page 3.16].

However, using differential corrections, the GPS position quality can be as accurate as 1 to 5 meters. The differential corrections allow the GPS receiver to remove some of the effects of SA and other signal degradation problems.

Finally, using post-processing or kinematic GPS techniques, we can obtain position accuracies measured in the centimeter or even millimeter ranges, by looking at the GPS carrier signal phase information. But these techniques are *extremely* sensitive to signal quality. Using post-processing, we have at best seen position accuracies within tens of centimeters for GPS data collected around our office park.

Table A.1 summarizes the accuracy we can expect from a GPS receiver using different techniques. These accuracies depend strongly upon the quality of the GPS signal received, which is heavily affected by the environment. Large buildings and heavy foliage can significantly degrade the GPS signal quality, resulting in disappointing position accuracies.

In fact, our application is one of the most demanding possible applications for GPS. We are collecting data in an urban environment, one of the worst environments for GPS data. This is often referred to as the "urban canyon" problem, which describes how signals are blocked by tall buildings. Signals bounce around the concrete and steel buildings, producing problems with multipath: the bouncing causes the same signal to reach the GPS receiver at different times, confusing the receiver and corrupting the position accuracy. We have yet to perform a complete evaluation of GPS performance in our urban area (Cambridge), but initial results are promising.

[HW94] and [Log92] describe the complete GPS system, and [vD98] provides a good discussion of GPS accuracy statistics. [Kai97] and [Lev97] discuss the use of GPS data in multi-sensor navigation systems, and [ASP96] has a good overview of using GPS data in large area image acquisition systems.

Appendix B

Smart Motors

"Smart Motors" [Ani96], manufactured by Animatics Corporation, are self-contained units, combining a servomotor, microprocessor, shaft encoder, and a serial interface in a single rugged package. Each motor also has inputs for two independent limit switches, one for each direction of travel, and a separate multi-purpose digital I/O connector.

The microprocessor executes user commands which are sent to the motor over an RS-232 serial link. These commands can be combined to create a motor program, which is downloaded to the motor and stored on a removable EEPROM microchip. When the motor is initially powered up or reset, it will run any program that is stored in the EEPROM. Motor commands include commands to move to a specified encoder count, or move at a specified velocity. Commands can also set and use the values of a limited number of named variables, or tell the motor to run a stored program. Finally, each command sent can be individually addressed to a specific motor.

We use the motor in the latter fashion. We first set a motor variable, and then command the motor to execute its stored program. The program uses the value of the variable to determine its actions, e.g. to calculate an encoder count to move to. The complete command sent to the motor looks like "a=20 RUN"; this command might tell the pan motor to move to a pan position of 20 degrees.

Multiple motors can also be connected in a master-slave relationship, where one motor coordinates the actions of a group of motors. The motors are connected in a serial daisy-chain, and configured to echo out any data that they receive on their input. The first motor in the chain can control the actions of the others by sending individually addressed commands to the other motors in the chain. Our leveling subsystem is one example of this. The first motor in the chain is connected to the level sensor, and calculates how the other motors should move in order to level the platform.

However, one caveat about using the motors in a daisy chain fashion is that we must be careful about timing and synchronization issues. Because the motor's serial interface is implemented without buffers, using a polling loop, the motor will stop running any stored program when it receives characters on the serial port, to try and minimize serial data overruns. If a master controlling motor sends commands to a slave motor while the slave motor is executing a program, the slave motor will stop executing its program. This can have possibly disastrous consequences. For example, if a slave motor program is in the middle of executing a "move at velocity ω" command when the slave motor receives a new command over the serial port, the program will stop executing so the motor can process the newly arrived command. But the interrupted program on the slave motor will never resume and command the motor to stop moving, and the slave motor will continue turning.

The inability to poll the motors while they are running a program is also a problem for the software module controlling the pan-tilt mount. We would like to have a non-blocking version of this module to use in the sequence control software, so that the sequence software can perform other actions (i.e. write data to disk and tape, download images from the camera) while waiting for the pan and tilt motors to finish their moves between images. We have implemented this using multiple threads: one thread blocks waiting for the pan or tilt motor to finish moving, while another thread is able to poll a shared variable to see if the motor is still moving, but still do other work as necessary. One benefit of this solution is that it isolates the interface for polling the motors from the actual implementation of the motor control software. It would have been possible to implement the motor polling by actually checking the serial port for new data from the motor (possibly indicating the end of the motor motion) between other tasks, but this approach requires that the polling software is aware of the details of the host-motor communication protocol.

Appendix C

Leveling Algorithm

This Appendix describes the leveling algorithm used on Argus. This algorithm was developed in conjunction with Michael Bosse, and is derived from work originally done by Peace River Studios and Kevin Wilson.

There are three actuators: *front*, back left, and back right. We will refer to the back left and back right actuators as *left* and *right* respectively. *Lowering* or *extending* an actuator causes it to move down from the base of the Argus platform towards the ground. *Retracting* an actuator causes it to move up, away from the ground.

The algorithm:

```
// First move all actuators down until they hit
// the ground.

read front/back tilt

// Front/back tilt will not change until front
// actuator hits the ground.
do
   lower front actuator
until front/back tilt changes
```

```
read left/right tilt

// Left/right tilt will not change until left or
// right actuator hits ground.
do
   lower right actuator
until left/right tilt changes

read left/right tilt

do
   lower left actuator
until left/right tilt changes

// All actuators are now in contact with the ground.
// Carry out actual leveling.

read left/right tilt

if left side is lower than right side
   do
      lower the left actuator
   until left/right tilt is level
else if right side is lower than left side
   do
      lower the right actuator
   until left/right tilt is level
end if

read front/back tilt

if front is lower than back
   do
      lower the front actuator
   until front/back tilt is level
else if back is lower than front
   do
      lower the front actuator
   until the front wheel is slightly off the ground

   do
      slightly lower left actuator
      slightly lower right actuator
   until front/back tilt is level

   // Recheck front/back tilt
   if front is lower than back
      do
         lower front actuator
      until front/back tilt is level
   else if back is lower than front
      do
         retract front actuator
      until front/back tilt is level
   end if
end if
```

Appendix D

Vendors and Components

This appendix lists contact information for the vendors with whom we have collaborated, or whose products are used on Argus.

- Mechanical infrastructure
  Peace River Studios
  9 Montague Street
  Cambridge
  MA 02139
  tel: (617) 491-6262
  fax: (617) 491-6703
  `http://www.peaceriverstudios.com`

- Navigation system
  Korbin Systems Incorporated
  69 Salem Street
  Andover
  MA 01810
  tel: (508) 470-4514
  fax: (508) 470-8664
  `http://www.korbin.com`

- Smart Motors
  Animatics Corporation
  3050 Tasman Drive
  Santa Clara
  CA 95054
  tel: (408) 748-8721
  fax: (408) 748-8725
  `http://www.animatics.com`

- Camera: DCS 420 professional digital camera
  Eastman Kodak Company
  Rochester
  NY 14650
  tel: (800) 235-6325
  `http://www.kodak.com`

- Hyperkernel
  Imagination Systems Incorporated
  5752 Princess Anne Road
  Virginia Beach
  VA 23462
  tel: (757) 497-8200
  fax: (757) 597-2062
  `http://www.imagination.com`

- SCSI-II interface: AHA-2940
  Adaptec, Incorporated
  691 South Milpitas Boulevard
  Milpitas
  CA 95035
  tel: (408) 945-8600
  fax: (408) 262-2533
  `http://www.adaptec.com`

- Tape drive: DLT4000
  Quantum Corporation
  715 Sycamore Avenue
  Milpitas
  CA 95035
  tel: (408) 894-4000
  fax: (408) 894-3218
  `http://www.quantum.com`

- Battery: GC12V100
  Interstate Batteries
  30c Nashua Street
  Woburn
  MA 01801
  tel: (617) 932-0076
  fax: (617) 932-0079
  `http://www.interstatebatteries.com`

- CPU: Peak 510 single board computer
  Optimum Industrial Computers
  315 West Brokaw Road
  Santa Clara
  CA 95050
  tel: (408) 327-0800
  fax: (408) 327-0808
  `http://www.optimumind.com`

- Quadrature/RS-232 converter: SEC-232m
  Fischer Computer Systems
  445 Bay Street
  Angwin
  CA 94508
  tel: (707) 965-2414
  fax: (707) 965-3687
  `http://www.fcs.net/fcs`

- Wheel encoders: model 2010
  Encoder Technology International
  Fallbrook
  CA 92028

- DC-DC converter: Mega Module
  Vicor Corporation
  23 Frontage Road
  Andover
  MA 01810
  tel: (978) 470-2900
  fax: (978) 470-2614
  `http://www.vicr.com`

- Serial port concentrator: AccelePort Xem (ISA) adapter,
  with PORTS/8em concentrator module
  Digi International
  11001 Bren Road East
  Minnetonka
  MN 55343
  tel: (612) 912-3444
  fax: (612) 912 4952
  `http://www.dgii.com`

Bibliography

[Ald92] Aldus Developers Desk. *TIFF 6.0 Specification*, June 1992.

[Ani96] Animatics Corporation. *Animatics SmartMotor User's Manual*, January 1996.

[ASP96] *Digital Photogrammetry: An Addendum to the Manual of Photogrametry*. American Society for Photogrammetry and Remote Sensing, 1996.

[BCS96] M.-O. Berger, C. Chevrier, and G. Simon. Compositing computer and video images sequences: Robust algorithms for the reconstruction of camera parameters. In *EUROGRAPHICS 96*, pages C23–C31, 1996.

[CMT98] Satyan Coorg, Neel Master, and Seth Teller. Acquisition of a large pose-mosaic dataset. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[CT97a] George T. Chou and Seth Teller. Multi-image correspondence using geometric and structural constraints. In *Proceedings of the 1997 Image Understanding Workshop*, 1997.

[CT97b] Satyan Coorg and Seth Teller. Matching and pose refinement with camera pose estimates. In *Proceedings of the 1997 Image Understanding Workshop*, 1997.

[CT98] Satyan Coorg and Seth Teller. Automatic extraction of textured vertical facades from pose imagery. Technical Report TR-729, MIT LCS, 1998.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modelling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH 96, Computer Graphics* Proceedings, Annual Conference Series, 1996, pages 11–20, August 1996.

[Eas95] Eastman Kodak Company. *Kodak Professional Digital Camera System: Programmer's Reference Manual*, November 1995.

[Eas97] Eastman Kodak Company. *Kodak Professional Digital Camera Application Programming Interface Specification (PDC-API)*, version 1.0 edition, April 1997.

[EMST91] Gerhard Ertl, Heimo Müller-Seelich, and Behnam Tabatabai. MOVE-X: A system for combining video films and computer animation. In *EUROGRAPHICS 91*, pages 305–314, 1991.

[Fau96] Olivier Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1996.

[FLR95] Olivier Faugeras, Stéphane Laveau, and Luc Robert. 3-D reconstruction of urban scenes from sequences of images. Technical Report 2572, INRIA, June 1995.

[GB96] Dorota A. Grejner-Brzezinska. Positioning accuracy of the GPSVan. In *The Institute of Navigation Proceedings of the 52nd Annual Meeting*, June 1996.

[How89] M. C. Howatson, editor. *The Oxford Companion to Classical Literature*. Oxford University Press, 1989.

[HTKW94] Michitaka Hirose, Kazuhisa Takahashi, Tomoki Koshizuka, and Yoichi Watanabe. A study on image editing technology for synthetic sensation. In *ICAT '94 Proceedings (The Fourth International Conference on Artificial Reality and Tele-Existence)*, pages 63–70, 1994.

[HTMW96] Michitaka Hirose, Kazuhisa Takahashi, Taku Morinobu, and Yoichi Watanabe. An alternate way to generate virtual worlds: A study of image processing technology for synthetic sensations. *Presence*, 5(1):61–71, 1996.

[HW94] B. Hoffman-Wellenhof. *Global Positioning System: Theory and Practice*. Springer-Verlag, 1994.

[HWE97] Michitaka Hirose, Shinjiro Watanabe, and Takaaki Endo. Image-based virtual world generation. *IEEE Multimedia*, pages 27–33, January 1997.

[HWE98] Michitaka Hirose, Shinjiro Watanabe, and Takaakai Endo. Generation of wide-range virtual spaces using photographic images. In *Proceedings of the IEEE Virtual Reality Annual International Symposium*, pages 234–241, 1998.

[Ima97] Imagination Systems Incorporated. *Hyperkernel Reference Manual*, rev. b edition, 1997.

[JLF95] William Jepson, Robin Liggett, and Scott Friedman. An environment for real-time urban simulation. In *1995 Symposium on Interactive 3D Graphics*, 1995.

[Kai97] James E. Kain. PAMS final report. DARPA SBIR Contract DAAH01-97-C-R208 final report, December 1997.

[Kal60] Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*, pages 34–45, March 1960.

[Law93] Anthony Lawrence. *Modern Inertial Technology: Navigation, Guidance, and Control.* Springer-Verlag, 1993.

[Lev97] Larry J. Levy. The Kalman filter: Navigation's integration workhorse. *GPS World*, pages 65–71, September 1997.

[Lip80] Andrew Lippman. Movie-maps: An application of the optical videodisc to computer graphics. In *SIGGRAPH 80 Proceedings*, pages 32–42, 1980.

[Log92] Tom Logsdon. *The Navstar Global Positioning System,.* Van Nostrand Reinhold, 1992.

[MLPT97] J. P. Mellor, Tomás Lozano-Pérez, and Seth Teller. Dense depth maps for epipolar images. Technical Report 1593, MIT AI Lab, 1997.

[MMC$^+$97] J. D. M. McKeown, C. McGlone, S. D. Cochran, Y. C. Hsieh, M. Roux, and J. Shufelt. Automatic cartographic feature extraction using photogrammetric principles. In *Digital Photogrammetry*, pages 195–212. ASPRS, 1997.

[Mot96] Motorola Position and Navigation System's Business. *Oncore User's Guide*, May 1996.

[Rob98] Dale Roberts. Interrupt behavior in Windows NT. *Dr. Dobb's Journal*, pages 74–82, April 1998.

[SEM97] Jeffrey L. Star, John E. Estes, and Kenneth C. McGwire, editors. *Integration of Geographic Information Systems and Remote Sensing*. Cambridge University Press, 1997.

[TB96] Charles K. Toth and John D. Bossler. Feature collection by the GPSVan: Analog video or stereo digital video. In *Proc. ASPRS/ACSM Annual Convention*, volume 2, pages 109–115. ACSM/ASPRS, April 1996.

[Tel97] Seth Teller. Automatic acquisition of hierarchical, textured 3D geometric models of urban environments: Project plan. In *Proceedings of the 1997 Image Understanding Workshop*, 1997.

[Tot95] Charles K. Toth. Experiences with a fully digital image acquisition system. In *Proc. ASPRS/ACSM Annual Convention*, volume 2, pages 18–24. ACSM/ASPRS, February 1995.

[vD98] Frank van Diggelen. GPS accuracy: Lies, damn lies, and statistics. *GPS World*, pages 41–45, January 1998.

[WB97] Greg Welch and Gary Bishop. An introduction to the Kalman filter. `http://www.cs.unc.edu/~welch/kalman/kalman.html`, 1997.

[Yat97] Charlie Yates. Note from Korbin Systems Inc., 1997.

4580 - 40