CHAPTER 1 – INTRODUCTION

1.1 Introduction

The responsible management of our oceans has become national policy and is generating a variety of initiatives to properly implement this policy. One such initiative in Atlantic Canada is the Eastern Scotian Shelf Integrated Management (ESSIM) project. It is a regionally driven, collaborative planning process for addressing conservation, multiple use and sustainable development of the ocean environment. Key interests in the ESSIM area include fisheries, oil and gas development, maritime defense, submarine cables, science, recreation and tourism, and marine conservation [Canada,2004]. Key to this effort is the underlying scientific information. Modern Geographic Information Systems (GIS) provide a fundamental tool to display and analyze information in support of the decision making process for policy makers, scientists and field operatives. The confluence of various mapping, processing and display technologies has created a revolution in data management, moving from a series of point data sets into a regional coverage mapping at progressively higher levels of resolution. The use of digital cameras underwater provides extremely high-resolution images of ocean features. Digital video provides a context to tie the high-resolution photos to the lower resolution acoustic imagery. Geographically registered mosaics of video frames are useful in GIS to bridge the resolution gap between digital photo and acoustic images. These mosaics, in effect, provide the meso-scale mapping in support of seafloor activities. As part of the ESSIM initiative, the Department of Fisheries and Oceans (DFO) has embarked upon a habitat mapping program which has been in part responsible for the development of a Towed

Camera system known as TOWCAM. This system carries two cameras, one a high resolution digital still camera, the other a digital video camera.

The Canadian Navy has implemented a surveillance program in key strategic waterways which includes the detailed mapping and identification of objects on the seafloor. The primary mapping tools are acoustic, while divers and Remotely Operated Vehicles (ROV) perform identification tasks. Video mapping has the potential to more efficiently map and identify objects in highly cluttered areas. The navy is actively pursuing the use of enhanced video technology for seafloor surveillance. One initiative of note is the development of a Laser Underwater Camera Enhancement (LUCIE) that could be incorporated as a mapping tool [Fournier, 2003].

The motivation behind this report is based upon the naval requirement for efficient object mapping in highly cluttered environments. This requirement was vividly emphasized through personal experience during the 1998 Swiss Air 111 crash recovery operations, where acoustic mapping provided the sole regional overview. A video mosaic of the main crash site would have greatly aided recovery and investigation efforts. Subsequent to this activity, the author became aware of TOWCAM which was in its early development stages. Although not designed as a mapping system, it appeared that most of the basic instrumentation was incorporated in the system to support the generation of mosaics. When the TOWCAM Team was approached about the possibility of mosaicing the video, there was initially a moderate level of interest shown, which has since grown to a very keen interest.

1.2 Report Contents

This report consists of five chapters: Introduction, Background, Video Mosaicing, Creating Mosaics From TOWCAM, Conclusions and Future Directions. A series of appendices follow the chapters to describe the TOWCAM System Offsets, examine attitude effects on output mosaic sections and provide C program code and shell scripts written in support of this report.

Figure 1-1 An idealised image of TOWCAM in operation. Image produced and provided by Kelly Bentham, DFO Canada.

1.2.1 Chapter 2 – Background

Chapter 2 consists of three main subsections: Seabed Surveillance and Intervention Operations, Early TOWCAM Experience and the LUCIE System. In the 1980s Canada's Navy began a transformation from a single-threat focus on Anti-Submarine Warfare towards a multi-threat combat capable force. This broadening of focus created a requirement for seafloor surveillance and intervention capabilities in

support of Mine Counter-measures (MCM) Operations. This requirement was formally assigned to the navy in the 1994 Defense White Paper [Canada, 1994]. Since then an acoustic mapping of strategic national waterways has been implemented. Expertise in marine survey and GIS operations is under development and has been used in response to a variety of operational scenarios beyond the scope of normal warfare operations to include urgent response to civil emergencies and counter narcotic operations. The civil responses are almost always a cooperative effort involving other government departments or agencies and often including special assistance from academic institutions. Beyond the success of these individual operations there has been a nurturing of the spirit of cooperation and trust between these agencies that has resulted in a symbiotic relationship that greatly enhances the routine programs and capabilities of the individual organizations.

TOWCAM provides an example of a project that, despite limited resources, demonstrated superior effectiveness in accomplishing a task to that of highly resourced systems designed specifically for that mission. The LUCIE system significantly extends visual range underwater and thus has the potential of covering far wider swaths. Wider swaths improve the coverage rate of a mapping system and allow for a larger overlap between parallel tracks, thereby improving confidence of complete area coverage.

1.2.2 Chapter 3 – Video Mosaicing Considerations

There are many examples of the use of video and photography in the creation of mosaics. Traditional photogrammetric techniques have been ported to the digital environment. The widespread availability of inexpensive digital cameras has generated great interest in the creation of panoramic mosaics for recreational purposes as well as in

archeological research. Significant effort has been undertaken in the field of machine vision, particularly in support of robotic systems. The Iraq war has highlighted the use of Unmanned Airborne Vehicles (UAVs) in a surveillance role. Video from these UAVs is mosaiced to form maps for the support of operations. A variety of underwater video systems have been developed from which video mosaicing techniques have been applied. This chapter briefly describes the implementation of various video mosaicing techniques and issues associated with correction of systemic errors and distortions in video data. The chapter concludes with the description of the technique to be implemented for the creation of video mosaics using the TOWCAM as it is presently configured.

1.2.3 Chapter 4 – Creating Video Mosaics From TOWCAM

Chapter 4 first describes the TOWCAM vehicle and instrumentation as well as its data logging scheme. A description of the work of the author that details steps, software integration and development then follows to describe: the creation of the mosaic including preparation of the data, selection and extraction of video frames and support data, correction of distortions, projection of the image frame to a plane, the combination of the projected frames into a mosaic, and finally the export of the mosaic for use in a Geographic Information System (GIS) such as ARCGIS.

1.2.4 Chapter 5 – Conclusions and Future Directions

Chapter 5 concludes the report with a description of system enhancements that could improve the mosaic quality of the TOWCAM system and describes the potential future application of this work.

CHAPTER 2 – BACKGROUND

2.1. Seabed Surveillance and Intervention

A nation's ability to conduct surveillance, protect, control and respond to events in its maritime regions is a reflection of its ability to maintain national sovereignty. The ability to conduct undersea operations is part of the sovereignty protection task that was assigned to the Canadian Navy in the 1994 Defense White Paper [Canada, 1994] and has been further amplified through the Defense Planning Guidance [Canada, 2001] and Maritime Commanders Planning Guidance documents.

Figure 2-1 Deep Sea Intervention System (DSIS) ROV deployable to depths of 1400m DND Photo.

Seabed intervention involves the ability to work below the water surface to accomplish assigned tasks and to collect information on the marine environment and resources. It includes the means to search, detect, inspect and recover items of interest which are located on the seabed or anywhere within the sea's water column. The associated tasks include scientific research, geological survey, fisheries research, aircraft crash investigation and submarine rescue [Reddy, 2002]. These Surveillance and Intervention tasks encompass and translate to war fighting capabilities in Anti Submarine Warfare (ASW) and Mine Counter Measures (MCM) operations.

The primary systems currently employed in support of surveillance and intervention operations include a suite of single beam and multi-beam side scan sonar systems for reconnaissance and localization together with Remotely Operated Vehicles (ROV) (Figures 2-1 and 2-2), and divers (Figure 2-3) for target classification, identification and/or intervention.

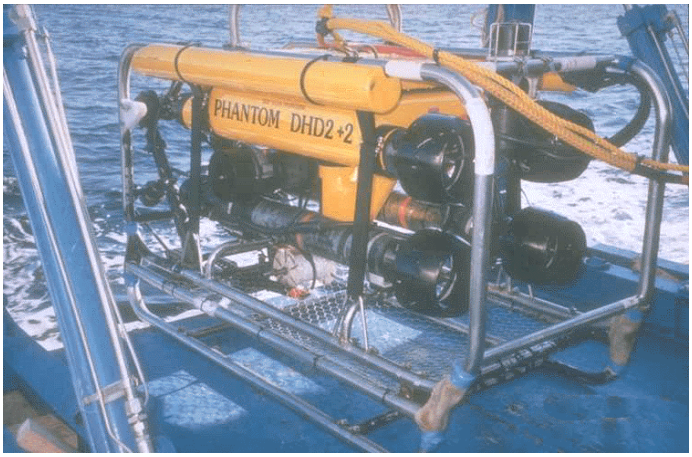

Figure 2-2 Phantom ROV is a light weight system fitted with sector scanning sonar, video camera and manipulator arm.

These systems have proven to be an effective combination in a range of operations including counter narcotic missions, aircraft crash search and recovery operations, Port Security activities and MCM Route Survey Operations. The navy currently has two ROVs and one Fleet Diving Unit on each of the East and West Coasts.

Figure 2-3 Diver suited up in CUMA2 Equipment, a mixed gas re-breather system for MCM Operations to a depth of ~80 metres. DND Photo.

The effectiveness of these systems is limited when the operation is in a high target clutter environment. Sonars are most effective in low contrast environments where the targets are observed in sharp contrast to the surrounding region, e.g. a mine on a sand bottom (Figure 2-4). In highly cluttered environments a sonar's ability to discriminate targets is severely degraded (Figure 2-5) [Klein, 1985]. Inspection and intervention operations are resource-intensive activities requiring the dedication of not only the diver or ROV Team but also their support infrastructure. This places a limit on the efficiency with which targets can be investigated. The navy has a vast underwater region to monitor, a growing security requirement to sustain, yet few diving and ROV assets to employ. A more efficient means to either reduce the number of targets to be inspected or to more quickly identify them is needed.

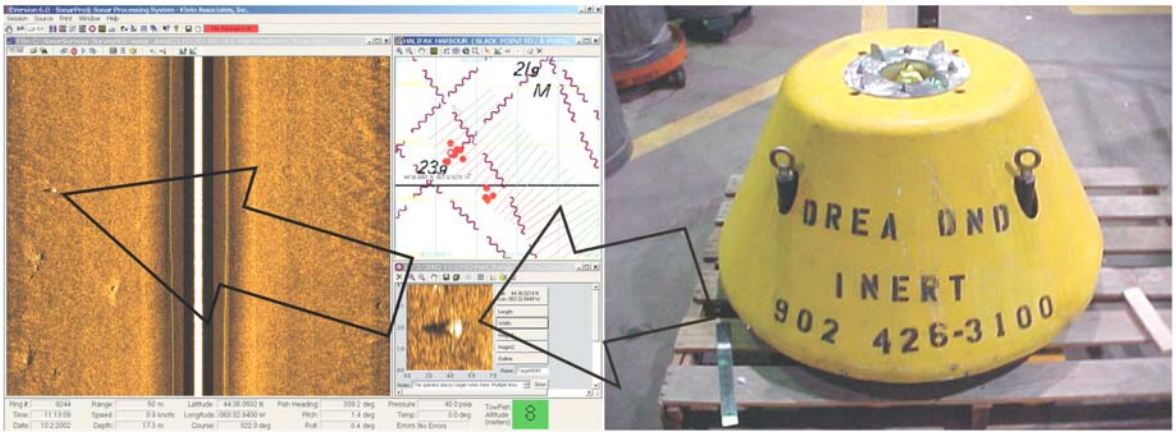

Figure 2-4 – High Resolution sonar image of mine shape on low contrast background (Sand Bottom). The shape is 80cm in diameter by 50cm high. DND Route Survey image 2002

The use of divers and camera-equipped ROVs permits the visual identification and examination of targets. The capability to direct ROV and diver assets onto targets is degraded in high clutter areas as the certainty of investigating the correct target is confused by the multiplicity of potential targets, inherent limitations in underwater navigation and the limited visual range.

Table 2-1 Resolution of Generic Seafloor Mapping Systems

System	Horizontal Resolution	Range	Qualitative Resolution Level
Multibeam Bathymetry	1-5 m	10-3000m	Meter
Sidescan	.1-.5 m	10-300m	Decimeter
Digital Video	.005-.02m	~3m	Centimeter
Digital Still	~.001m	~3m	Millimeter

A suite of systems are used to investigate the seabed employing the acoustic and visual spectrums in a tradeoff between resolution and coverage range (Table 2-1). Multibeam bathymetry systems are used to provide rapid coverage of relatively wide swaths with low resolution. Sidescan systems provide higher resolution coverage of generally narrower swaths at lower coverage rates (Figure 2-7). Digital video provides

superior resolution to sonar (Figure 2-6) while providing low resolution visual detail of very narrow swaths at very slow coverage rate. Finely detailed imagery is supplied through high quality digital still camera systems.

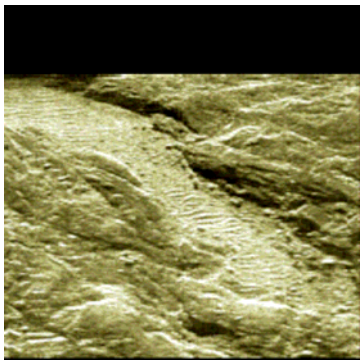

Figure 2-5 - Sonar image of glacial erratic boulders on bedrock outcrop. High contrast background limits ability to resolve targets of interest and would be classified as an "un-huntable bottom" for acoustic Mine Hunting Systems. Image courtesy of Klein Sonar Associates.

Rzhanov et al. [2000] suggest the ideal product from sea floor imaging efforts would be a complete 3D reconstruction of the scene with the resolution required to resolve essential details , and accurate geo-referencing such that subsequent maps could be compared. They go on to establish that the ideal is both difficult and costly to achieve and in many cases not necessary to meet objectives. This is in fact the case as it pertains to naval seabed intervention operations. Alignment of 2D sonar images of the 3D seafloor is routinely done through sidescan mosaics. Subsets of these mosaics are used as operational backdrops on electronic chart displays or produced in hard copy operational graphics for use in planning and pre-dive briefings (Figure 2-8). It is commonly assumed in the mosaic process that the sea floor is flat.

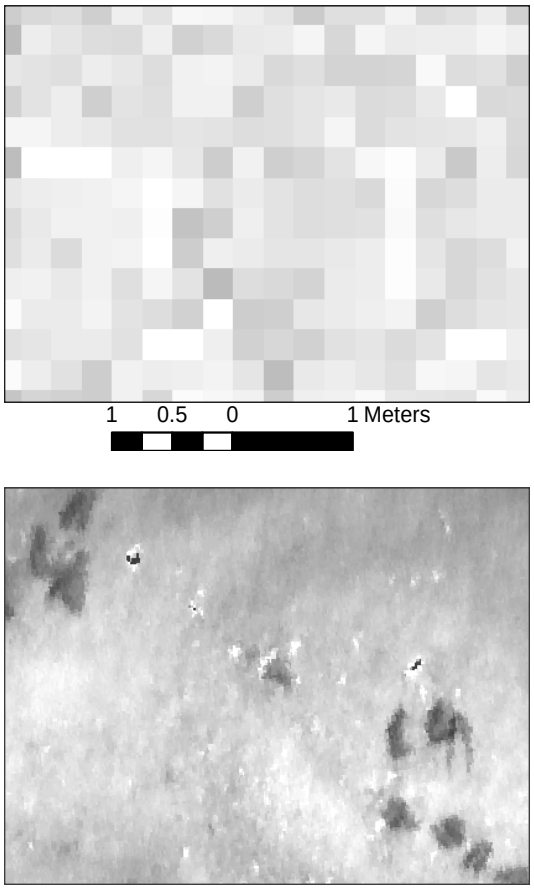

Figure 2-6 Comparison of resolution levels between 25cm side scan sonar mosaic(top) and 2 cm resolution video mosaic(bottom). The sonar image at this scale is meaningless (dominated by speckle) while the video image displays rocks marine growth and shell hash. The video mosaic was generated by the author using the procedures outlined in this paper.

Errors in a video mosaic due to terrain effects can be neglected as the areas mapped using optical imagery are relatively flat. The analysis errors incurred because of the flat bottom assumption are less than those related to navigational inaccuracies that must be considered if separate frames are used as opposed to a mosaic [Rzhanov et al., 2000]. The use of consumer grade attitude sensors has been shown to effectively reduce the complexity of the transformation model of the images from projective to affine. By estimating tilt angles, with modest accuracy, frames can be corrected for projection distortion and the mosaic quality is greatly improved [Rzhanov et al., 2000].

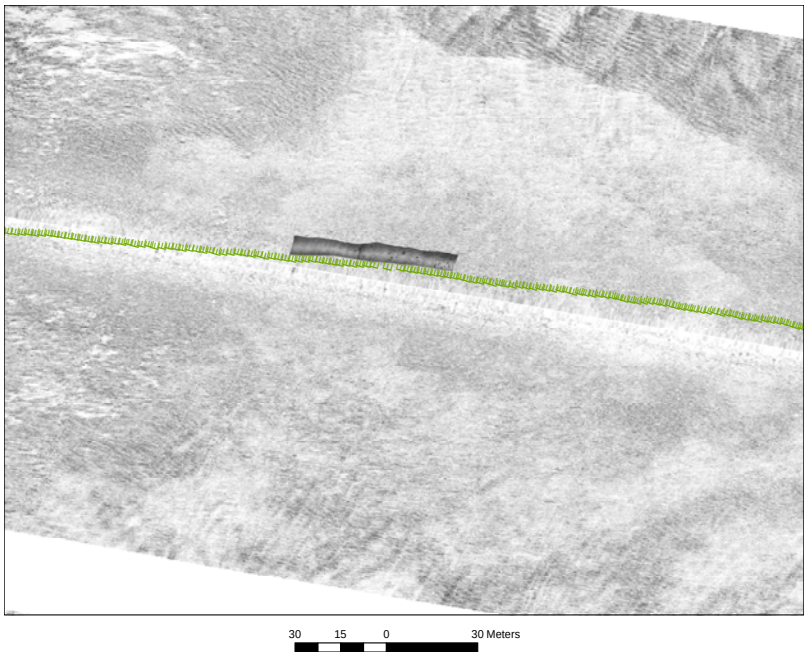

Figure 2-7 Sonar mosaics provide broad coverage (200m swath) useful in regional interpretation of seafloor characteristics. A 40m section of video mosaic is overlain for visual comparison of swath widths. The survey vessel track line is also shown. Sonar mosaic courtesy of DFO Canada.

The minimum auxiliary data to support the process is a time series of camera roll and pitch that is synchronized with the video frames. The synchronization must be maintained at the frame level, even though successive frames in a video are virtually never employed in developing a mosaic. The accuracy of the roll/pitch information should be on the order of one degree or better. The ability to determine heading provides an azimuth relative to north which is used in creating north up mosaics [Rzhanov et al, 2000].

Experience with sidescan mosaics has shown that data volumes and display of high resolution mosaics can be an issue. An uncompressed grayscale geotiff with 10cm resolution of a 1 sq km area represents a file size of 100mbytes. A 2cm resolution video geotiff in grayscale of the same coverage would be 2.5 gbytes. The mosaics are used as general back ground or reference images and indeed the raw data is kept near line for

recall when detailed analysis is required. A similar approach to the use and exploitation of video data would be appropriate.

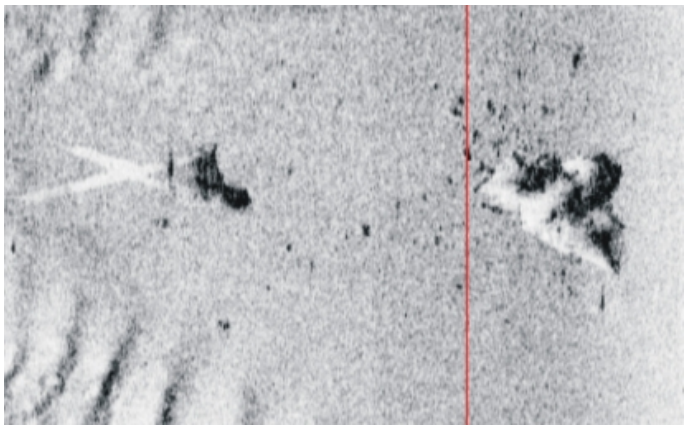

Figure 2-8 Example of sonar mosaic image of an aircraft crash site suitable for incorporation in to a planning graphic or use in a raster based electronic chart to support ROV or diver operations. DND Route Survey Image 2000.

The processed sidescan mosaic images are stored along with a track file that provides sensor location, time and source data file. This allows for ready retrieval of raw imagery on demand. The same method could be employed for video data. It is further suggested that it would not be necessary to pre-mosaic the entire video footage. The mosaic of a 10m swath shown at 1:1000 is a 1cm thick line, hardly useful (figure 2-7). A recent survey of the Saint Lawrence River covered a navigation channel 200m wide and 600km long. If this project was plotted out at a 1:1000 scale, the plot would be 20cm wide for 600m! It is proposed that the raw data be stored with pre-processed positioning and attitude data such that mosaic areas can be identified using the track file, and mosaics created dynamically for incorporation into operational graphics and GIS or for use in raster based electronic chart displays.

2.2 TOWCAM

TOWCAM is an underwater towed camera system that includes a video camera and digital still camera for imaging the sea floor (Figure 6). It is towed at slow speeds and owing to limitations on underwater visibility, in close proximity to the bottom at heights of 2-3m. This system has been developed by fisheries scientists of DFO for use in habitat studies, benthic stock assessments and trawl impact studies. The system has been assembled through the use of "in-house" resources and commercial off the shelf (COTS) components [Gordon et al,1998].

Figure 2-9 – TOWCAM II Vehicle fitted with video camera and digital still camera. (DFO Photo)

2.2.1 Towcam System Description

The system is instrumented with a flux gate pitch and roll sensor, a pressure depth sensor, an echo sounder for use as an altimeter and a transponder for Ultra Short Base

Line (USBL) acoustic positioning. Imagery is collected by a digital still camera with strobe flash and a digital video camera with Hydragyrum Medium arc Iodide (HMI) lights for illumination. The system is towed at speeds of about 2-3 knots at an operator-defined altitude above the bottom. This altitude is maintained by a feedback mechanism which controls the shipboard winch paying out and reeling in cable as necessary. Designed as a tool to inspect general features on the seafloor, it has on occasion been used to search for discrete items. TOWCAM has several limitations on it's effectiveness in this role, but it has demonstrated that with some sensor enhancements it could be a very useful tool for target search and identification.

2.2.2 TOWCAM In A Search Role

In 1996, during a NATO exercise, 36 exercise mines (Figure 7) were laid by US B52 bombers near Halifax for re-location by NATO Mine Hunting Forces. The mined area turned out to be on a bottom type that is classified as "un-huntable" by acoustic systems and the mines were never located by the mine hunting systems deployed during the exercise.

Figure 2-10 US Mk82 Bombs being loaded onto a B52 Bomber, a conversion of the nose arming mechanism changes this free fall bomb into the MK36 Destructor Sea Mine. (USAF Photo)

In the spring of 2001, TOWCAM was undergoing engineering trials in advance of its survey season. In order to challenge the control system, a bottom with rough terrain was sought for a part of the test. Discussion between the DFO Engineer and DND Route Survey identified the area of the lost mines as a suitable location to test the TOWCAM system while simultaneously testing it's utility as a search platform. Provided with a best estimate location of the position of two of the mines (estimated at +/- 200m) TOWCAM deployed to the location and in a period of just 4 hours located both of the mines in 45 metres of water (Figure 2-11).

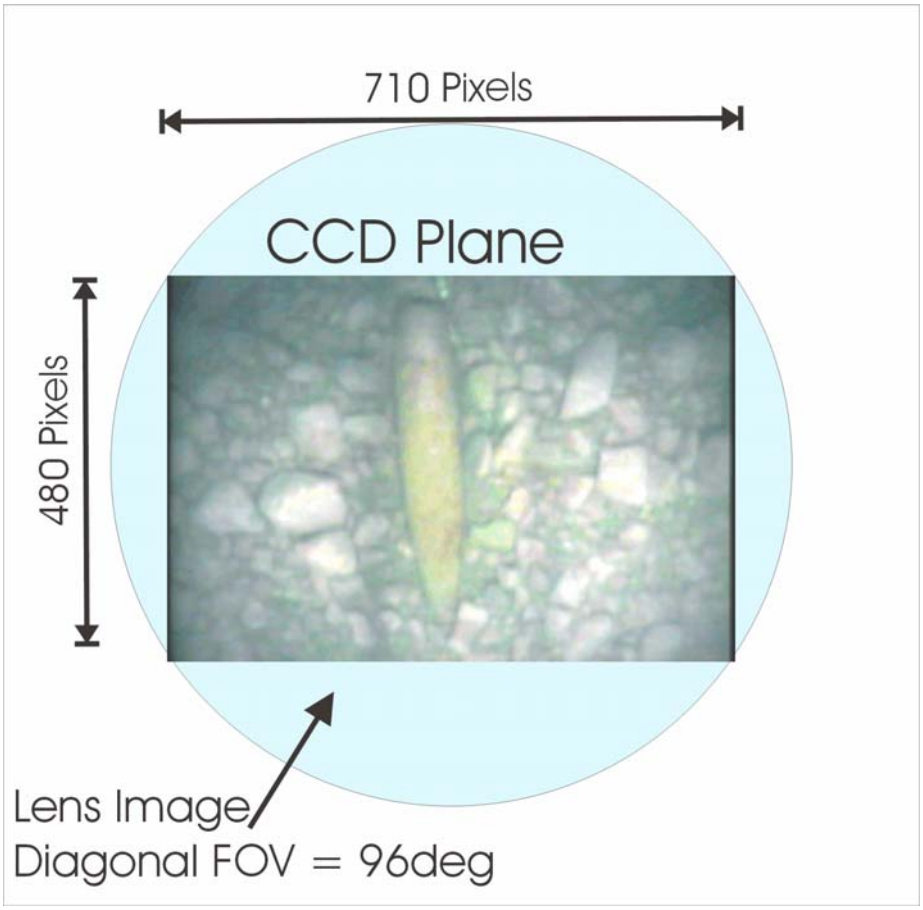

Figure 2-11 TOWCAM video screen capture of an airdropped 227kg Mark 36 "Destructor" Mine shape 1.67m long by 0.27m diameter on cobble bottom. The mine shape was undistinguishable from the background using sonar but is immediately apparent by video. DND Route Survey Image taken May 2001. The Video image was taken from an altitude of ~2m. With an assumed camera pitch angle of 0º the image plane is calculated to measure 3.34m wide by 2.03m high.

Ambient visibility was 3-4m and the bottom was composed of cobbles and boulders. There are limitations to TOWCAM in a search and identification role. The vehicle must be towed at very low speeds which seriously degrades a vessel's ability to follow or maintain a search track. Underwater visibility limits the altitude at which the bottom can be imaged and thereby limits the swath width which can be searched in this case to about 3m. The USBL positioning system in use provides a navigation accuracy on the order of 5m RMS thus (given the narrow swath) reducing the level of confidence of achieving complete area coverage in a search operation. Along track coverage poses no

problem as the video frame rate is ~30 frames per second which, conservatively assuming an along-track swath coverage of 1 metre, equates to a maximum tow speed of 30 metres per second (~60 Knots).

2.3. Laser Underwater Camera Image Enhancer (LUCIE 2)

The limited range of visibility underwater is due to a combination of attenuation in the medium and scattering from particulate matter. This limited visibility impairs the utility of video cameras on ROVs and, in the case of mine clearance operations, narrows the margins of safe operations for equipment and divers. Reducing this limitation has been the subject of research undertaken at the Defense Research and Development Center (DRDC) in Valcartier. A video camera system has been developed which limits the effect of volume scattering on an image, resulting in a marked increase in visual range underwater.

2.3.1 LUCIE Described

LUCIE combines a short laser pulse of several nanoseconds with a time-gated image intensifier in order to enhance the range of visibility underwater. A laser pulse is transmitted and after a suitable delay the camera gate is opened for several nanoseconds and an image is recorded (Figure 2-12). This means that the intense backscatter from the water column lying between the target of interest and the camera can be prevented from occluding the desired image. The LUCIE 2 system extends the range by a factor of 3 to 5 over conventional cameras when there is no natural illumination [Wiedemann et al, 2002].

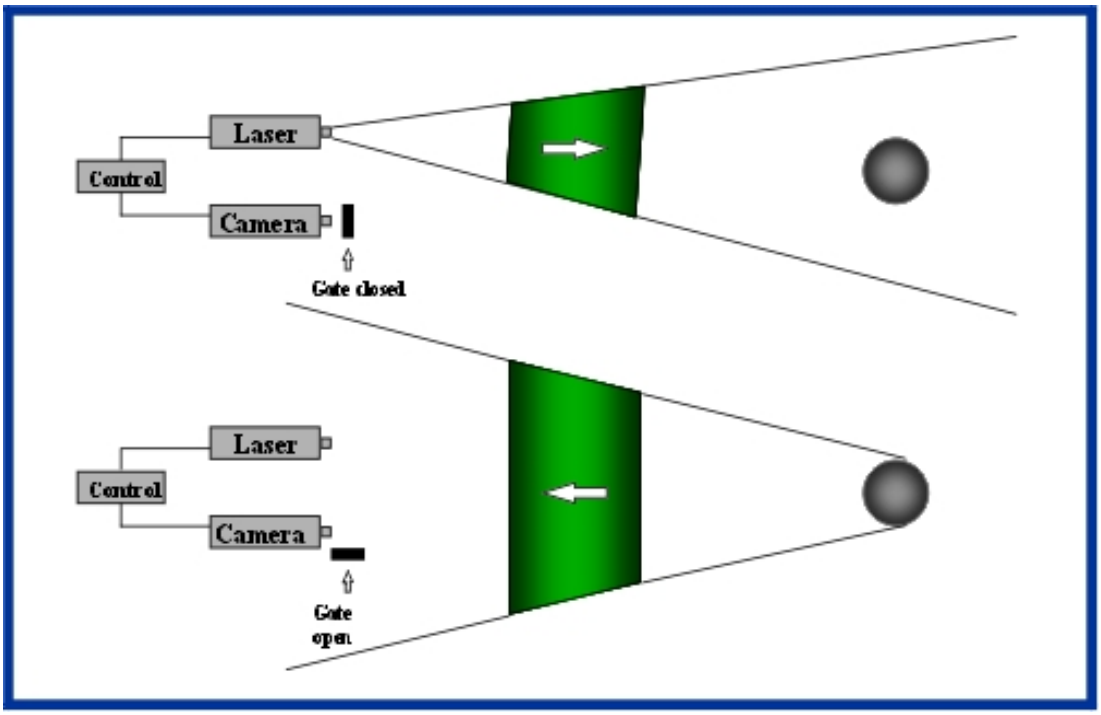

Figure 2-12 LUCIE Principle of Operation, after [Canada, 2000].

2.3.2 LUCIE As A Mapping Tool

This system has been under development for several years by DRDC at Valcartier for use in diver and ROV operations. Its deployment in a towed configuration as a mapping and inspection tool is an area of recent interest. Electro-Optical (E/O) systems such as LUCIE perform their best over bottoms with high contrast such as rocky or cobbled bottoms where there is little sand or sediment to mask targets. This is the environment where acoustic systems are greatly disadvantaged. Conversely, low contrast muddy bottoms are a problem for E/O systems owing to sediment coverage. This suggests a complementary role for these systems. LUCIE 2 employs a laser for illumination. The laser has a wavelength of 532nm with a repetition rate of 21Khz and a pulse length of 5ns. The illumination system is controlled by a holographic beam shaper that produces a flat illumination field with a 4/3 aspect ratio. Target ranges can be determined by use of high frequency sonar (Figure 2-14). The aspect ratio matches the

field of view (FOV) of a standard video camera. The intensity of the illumination varies by less than 5% over 90% of the FOV [Fournier, 2003].

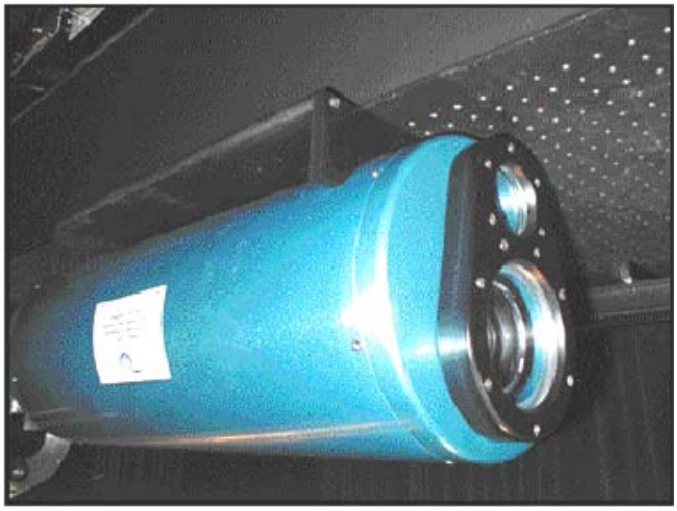

Figure 2-13 LUCIE 2 camera system 25cm in diameter and 70cm long, neutrally buoyant in water, after Weideman et al [2002].

2.3.2.1 LUCIE Optical and Video Specifications

The lens system is 10cm in diameter with a zoom range of 16mm to 160mm at an f=1.8. The lens has an auto-iris control and a fully motorized focus and zoom. Both the camera and the illuminator can be zoomed from 80 to 800 mr in water. The laser divergence and lens system can be slaved together to ensure maximum uniform illumination over the entire range of fields of view. The lens (top) and laser port bottom can be seen in Figure 2-13. The intensifier gate delay can be varied from 0 to 500ns and the gate width can be increased from 3 ns to 500 ns. The video output is digitized at full resolution 640 by 480 pixels at 30 frames per second. Given a 21KHz pulse repetition rate each camera frame is thus the average of 700 pulses. This technique has two notable advantages, the first is the elimination of laser speckle on the images and the second is a large reduction in eye safe range down to one meter from the aperture, thereby significantly reducing the risk of personnel injury [Fournier, 2003].

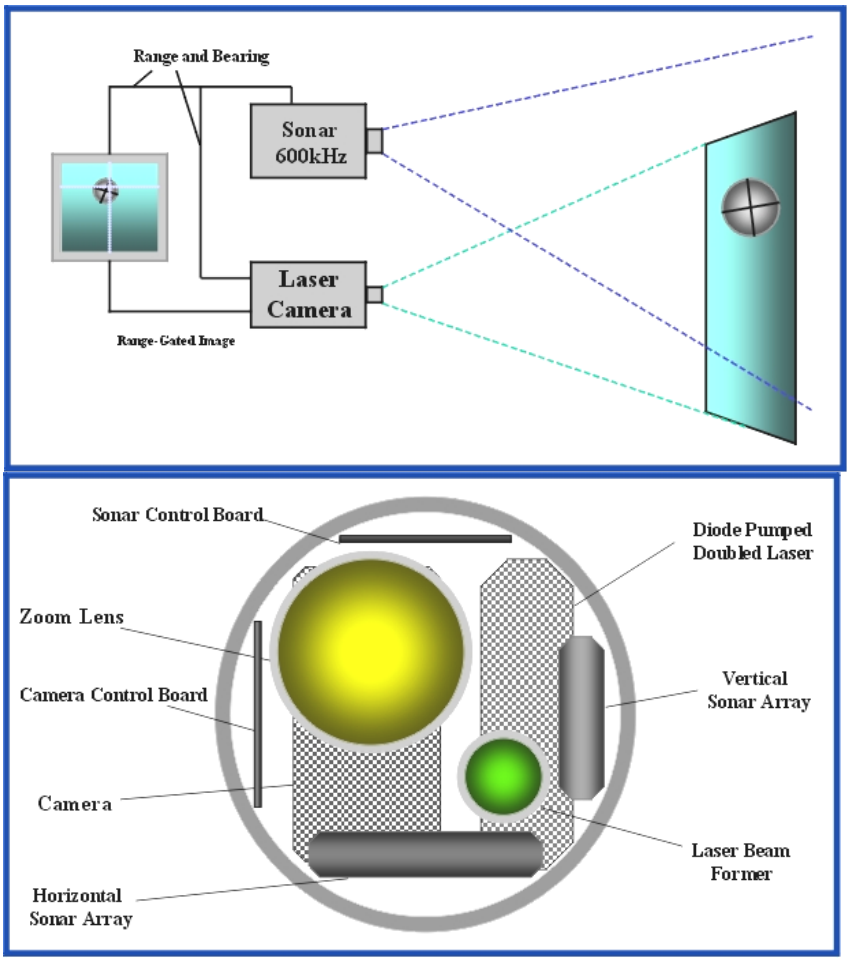

Figure 2-14 The high frequency sonar on LUCIE can be employed to provide initial target ranging to allow setting of laser pulse and camera gate parameters, after Canada [2000].

2.3.3. LUCIE in a Towed Configuration

The enhanced range achieved with the LUCIE system would allow for a wider swath of coverage in a towed configuration as the vehicle can be operated at greater altitude off the bottom. This also has the benefit of reducing interaction with the bottom which causes reduced visibility from disturbed sediments and reduces risk of impact damage or damage due to tangling with bottom obstructions. The wider swath also increases the reliability of achieving complete bottom coverage as the footprint can encompass the navigational error budget. The LUCIE 2 camera, as described, mounted in

a tow body would have a maximum 40º swath width. At a height of 10 meters, this corresponds to a swath width of 8.4 meters. The current measured real minimum resolution of the high-gain gated-intensified camera is 240 line pairs across. This gives a resolution on the order of 1.75cm on the bottom. As there are 640 pixels across the screen, the pixel size is 1.15cm [Fournier, 2003].

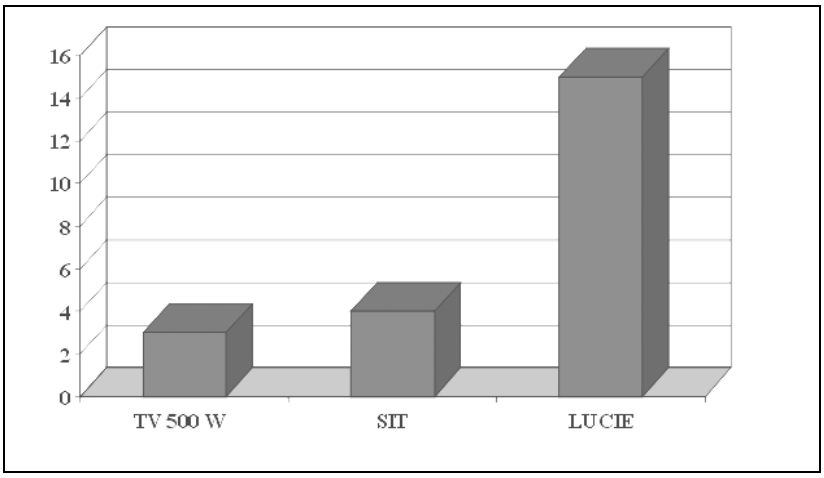

Figure 2-15 . Comparison of visible range results in 2m visibility water from tests conducted by DRDC Valcartier employing a standard video camera with quartz halogen illumination, a low-light Silicon Intensifier Target (SIT) Camera and the LUCIE system after Canada [2000]. The "Y" axis is visibility in metres.

Port approaches and harbours tend to have poor visibility conditions due to estuarine oceanographic and anthropogenic contributions to the water column. Furthermore, ROV and diver operations are severely hampered by the silt that gets stirred into the water column from their own activities. Two to three meters visibility is typical for harbour conditions. Figure 2-15 depicts the range advantage achieved by LUCIE. It should be noted that although the physical enhancement is by a factor of from 3 to 5 times, the operational enhancement due to reduced interaction with the seafloor sediments is a factor of ten [Fournier, 2003]. The navy is actively investigating the adaptation of LUCIE for use in a towed system for inspection and mapping operations.

2.4 Video Mapping as a Surveillance Tool

The demand for meso-scale imagery to better resolve seafloor targets has significant support. The use of rudimentary systems fitted with inexpensive attitude sensors such as found in the TOWCAM vehicle provides a cost efficient method of achieving this end. The ongoing developments in underwater electro-optics as demonstrated by the LUCIE System provides for enhanced coverage rates and vastly improved confidence in the thoroughness of coverage achieved.

Chapter 3 – Video Mosaicing Considerations

3.1 General

Video is an important tool for underwater operations. It is commonly employed to enhance diving and ROV operations, providing a continuous record of the experience as well as enabling the control and monitoring of the under-water activity. The technology is widely used, with off-the-shelf systems readily available. The system setup and operational costs are lower than many other sampling techniques and it provides data in small study areas invaluable for analysis. The application of video in larger study areas has been more limited [Williams and Leach, 2000]. The use of video mosaics created from transect coverage can provide a meso-scale view of larger study areas and is thus highly desirable as an aid to tie between small study areas.

3.2 Mosaicing Techniques

Video mosaicing is a routine activity. Whether from highly specialized unmanned airborne vehicles (UAVs) for intelligence and targeting systems, such as the USAF's Global Hawk (figure 3-1), or more modest underwater towed video systems, the demand for a coherent geo-registered composite image exists.

Figure 3-1 Global Hawk surveillance UAV is fitted with a range of electro-optical systems and can generate geo-spatially registered video mosaics in near real time. USAF Photo.

Early mosaicing methods were manual in nature, consisting of the joining together of hard-copy prints by laying them together and taping them to form a large sheet. Later image processing using computers has permitted stored images to be merged digitally [Marks et al., 1995]. Today, inexpensive consumer grade digital cameras are delivered with software to automatically generate panoramic mosaic scenes from a sequence of still photos figure 3-2.

Figure 3-2 Panoramic image generated from a sequence of four digital still photos with ~20% overlap. The software provided with the camera system seams the photos together by matching features, there are no lens or geometric correction factors applied. This is emphasized by noting that the wooden border to the garden is, in fact, a straight wall.

3.2.1 Underwater Mosaicking

The underwater environment introduces a variety of complications to the generation of geo-registered mosaics. Limited underwater visual range requires camera systems to be operated in close proximity to the bottom, and thus limits the width of coverage. The camera must be operated from a remote vehicle, either towed or autonomous, which complicates the positioning and attitude solution by introducing additional degrees of freedom to the solution, thus leading to greater positional uncertainty. The traditional approach is to point the camera as nearly vertical to the seafloor as possible in order to limit perspective distortion. This is often not feasible, as one of the main purposes for the video camera is to look ahead for potential obstructions.

3.2.1.1 Sensor-Only Mosaic

Haywood [1986] proposes taking images at precisely known coordinates with high precision, permitting image merging by computing frame-to frame motion parameters directly from the camera positions. Marks et al., [1995] describe a four parameter semi-rigid motion model for creation of near real-time ocean floor mosaics that Gracias and Santos-Victor [1998] say restricts the scope of applications to images whose retinal plane is closely parallel to the ocean floor.

3.2.1.2 Estimated Motion and Image Matching

It is common to use the pinhole camera model in computer vision applications to linearly map from the 3D projective space to a 2D projective plane. The 2D projective transformations can be used to model image motion. Creation of video mosaics is accomplished by registration of the image where image motion is estimated, the individual frames are fitted to a global model of the video sequence, and the mosaic is then rendered by applying a temporal operator over the registered and aligned images [Gracias and Santos-Victor, 1998].

3.2.1.2.1 Optical Flow

Optical Flow is a technique to compute frame transformation parameters needed to perform the affine transformation to align sequential video frames. Obodez and Bouthemy [1995] describe a robust method for the estimation of parametric motion models and have developed a software library called Motion2D which is an implementation of the robust, multi-resolution and incremental estimation method. The motion analysis is performed by identifying the 2D parametric models of the optical flow field using the polynomial models of the point coordinates in the image plane. Those

models include constant flow (translation), affine flow (first order polynomials in x and y) and quadratic flow. This method is effective on static scenes.

3.2.1.2.2 MPEG Motion Vectors

Jones et al. [1999] describe the use of the motion vectors encoded by the MPEG video compression scheme to estimate camera motion and thereby create video mosaics. The MPEG vectors determine the camera's pan, tilt and zoom factor which is then utilized to align the video frames. This method is faster at creating mosaics than those that rely on other image processing techniques to estimate camera motion. This method is liable to produce erroneous results when there is significant object motion within the video.

Figure 3-3 Mosaic of underwater pipeline, after Garcia and Santos-Victor (1998).

3.2.1.3 Oblique Video and Image Registration

Kumar et al., [1999] describe the near real-time registration of highly oblique aerial video collected by UAVs. Relying on the redundancy provided by significant frame-to-frame overlap in the video, they use key frames to compute the frame-to-frame motion which along with frame-to-frame alignment parameters permits the generation of a single extended view mosaic. They go on to achieve fine geo-registration by refining the estimate using the relative information between frames to constrain the general solution. The transformation is modeled in two parts by an external coordinate transformation that specifies the 3D alignment parameters between the reference and camera coordinate systems and an internal camera coordinate system to image transformation involving an affine transformation and non-linear lens distortion parameters. This transformation model is combined with Digital Elevation Model (DEM) data to completely specify the mapping between the video pixels and the reference imagery. High quality attitude and optical sensors coupled with precision positioning through GPS readily available to an airborne sensor serve to strengthen the solution.

3.3 Lens Distortion

Pers and Kovacic, [2002] state that,

"Radial lens distortion prohibits use of simple pinhole camera models in computer vision applications, especially when using wide-angle lenses, which result in barrel type distortion. The usual approach to radial distortion is by the means of polynomial approximation. Lens distortions are long-known phenomena that prohibit use of simple pinhole camera models in most of the computer vision applications. Being the most stubborn type of lens aberrations, they do not influence quality of the image, but have significant impact on image geometry [Slama, 1980]. Several types of lens distortions exist, however, radial distortion is usually the most severe part of the total lens distortion, especially when inexpensive wide-angle lenses are used."

Inexpensive off-the shelf cameras are commonly employed in underwater applications, thus radial lens distortion must be accounted for.

3.3.1 Radial Lens Distortion

According to Derenyi [1996], radial distortion displaces the image points in a radial direction away from the principal point. Outward displacement is considered positive and is commonly described as barrel distortion. The determination of lens distortion is an integral part of camera calibration. The three methods to specify distortion are graphically, in tabular form or by a mathematical model. The mathematical model is more commonly employed in the case of non-metric camera systems such as consumer grade digital cameras. Karras and Mavrommati [2001] state that,

"radial symmetric lens distortion Δr can be determined in a common solution with the other interior orientation parameters (through test-field calibration or bundle adjustment) but may also be estimated separately. Such independent knowledge of the radial distortion of a lens is useful in a variety of cases, for instance:

a. correction of image coordinates in rectification ;
b. correction of image coordinates in cases where "nominal" interior orientation values are used;
c. correction of image coordinates for test-field calibration or self-calibration algorithms which cannot recover Δr; and
d. for resampling digital images (rectified or otherwise) to free them from the distortion effect."

Radial symmetric lens distortion is expressed through the coefficients of a polynomial referring to the radial image distance r

$$\Delta r = k_0 r + k_1 r^3 + k_2 r^5 \quad (1)$$

and is split into its x and y components as

$$\Delta x = \Delta r\, x/r = x\,(k_0 + k_1 r^2 + k_2 r^4) \qquad \Delta y = \Delta r\, y/r = y\,(k_0 + k_1 r^2 + k_2 r^4) \quad (2)$$

Karris and Mavrommati [2001] identify a variety of techniques to estimate distortion parameters including:

a. from the curvature of straight lines;

b. with the rectification of regular grids;

c. from the rectification residuals; and

d. from the residuals of partial rectification.

More rigorous calibration may be achieved through the use of planar test fields and processing through a self-calibration package as envisioned by Zhang [1998] or through Finite Element Modeling and bundle adjustments as described by Li [1999].

3.4 Lighting Correction

Borgetto et al, [2003] describe underwater lighting as being non-uniformly distributed. Solving this problem during a pre-processing step improves both image quality and mosaic creation. Radiometric correction and homomorphic filtering are suggested as two avenues to resolve this problem.

3.4.1 CCD Camera Radiometric Correction

Model-based CCD camera radiometric correction is commonly used in remote sensing and astronomical imaging [Gonzalez, 1992]. The model subtracts a dark reference image (DR) from the acquired image and then divides the result by a uniformly- lit reference image (ULR) to obtain a corrected image. Implementation of this model requires an estimation of the DR and ULR images.

3.4.1.1 Dark Noise Image

CCDs emit a small signal even though no radiation is being detected. This is known as Dark Current and is the residual electronic noise in the system at any temperature above absolute zero. Each detector will have a slightly different response at

any given temperature and/or pressure resulting in an image of dark noise superimposed onto the collected image [Richards, 1999]. A DR image may be acquired by closing the camera objective in the desired ambient conditions and recording the image [Borgetto et al, 2003].

3.4.1.2 Uniformly Lit Reference Image

Borgetto et al, [2003] describe three techniques for obtaining a ULR image. The first is by convolving a sequence of images with a Gaussian filter to yield a smoothed image. The second is to acquire the image from a uniform bottom, in uniform lighting while diving. The third is by modeling the lighting conditions, which is possible in situations where no natural light is available.

3.4.1.3 Homomorphic Filtering

Homomorphic filtering is a generalized technique for image enhancement and/or correction. It simultaneously normalizes the brightness across an image and increases contrast [Gonzalez and Woods, 1992]. Borgetto et al, [2003] calculated the filter parameters qualitatively as they had no means to work out the parameters automatically, and reported good results but noted that operator intervention was necessary to set the parameters.

3.5 Mosaicing Scheme for TOWCAM Video

TOWCAM was built as a tool to study seafloor habitat rather than as a mapping system. It does include positioning capability and is fitted with engineering sensors to support its operation and system control. These sensors may be adequate to permit the creation of coherent, spatially-registered mosaics for support of habitat studies. Mosaics will be created from up to 25 second segments of video. This length is selected as it

allows an averaged position solution to be used while providing a mosaic of reasonable length (~30m). Frame-to-frame image correlation techniques will not be used in this implementation as the intent of this report is to focus on the use of available sensor data.

3.5.1 Camera Model

The approach will be to use a pinhole camera model, supported by the available sensor data, to project the image frames to an assumed flat earth creating a two-dimensional mosaic from the available video. TOWCAM is towed at very slow speeds and significant frame-to-frame overlap occurs (~98%) see figure 3-4.

Nominal Tow Speed (v) in m/s

$v := 1.5$

Central Pixel Resolution (p_res) in m for frame at 2m altitude and 30 degree pitch

$p\_res := 0.005$

Video Frame Rate (f_rate) in frames/sec

$f\_rate := 30$

Pixel Rows per frame (n_rows)

$n\_rows := 480$

Number pixel rows required (rows_req) for 100% coverage

$$rows\_req := \frac{v}{(p\_res \cdot f\_rate)}$$

$rows\_req =$

Percent Overlap = $\frac{(n\_rows - rows\_req) \cdot 100}{n\_rows} =$

Figure 3-4 Computation of TOWCAM nominal frame overlap

3.5.1.1 Image Correction

The least-distorted rows from each frame will be used to limit the effects of un-modeled distortions and misalignments. This approach permits a more rudimentary calibration of the camera lens to be performed to correct for radial distortion as

distortions in the upper and lower reaches of the image are no longer a factor. The application of a radiometric correction is also simplified by this approach as the correction needs only apply to the central few image rows which have a lesser variation in range from the lens between pixels. The mosaic will be produced in grayscale. Williams and Leach [2000] state that,

> "The video contains red, green and blue color channels which combine to form the color video image. Water absorbs red light wavelengths more readily than green or blue. Even with natural illumination, the red spectral information is removed when imaging from a distance greater than approximately 2m, with green light next to be absorbed at around 10 metres. This absorption produces imagery with a distinct blue-green coloration, with the reduced spectral range resulting in less apparent detail."

Generating the mosaic in grayscale will thus provide an image less sensitive to wavelength absorption with negligible loss of detail (see figure 3-5). The reduction from a 24 bit (color) to an 8 bit (grey) image reduces file sizes and processing effort by a factor of three.

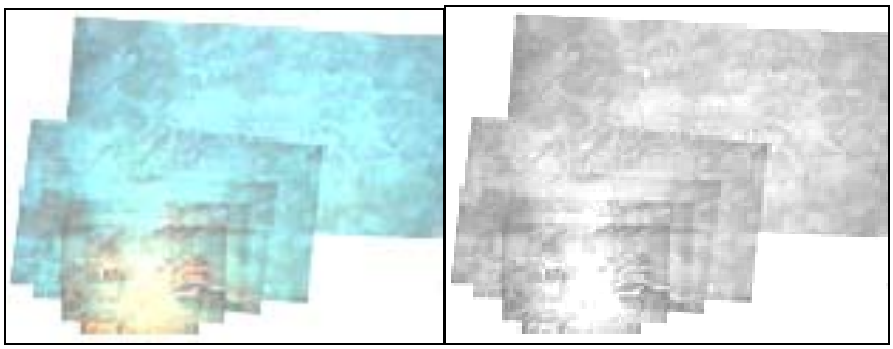

Figure 3-5 Comparison of color versus grayscale. The left hand image is an RGB color mosaic, while the image on the right is the grey value conversion of the same image. There is negligible loss of detail between the two images while the grey provides a more consistent overall picture. The images are after Williams and Leach [2000].

3.5.2 Export Mosaic to GIS

The geo-registered mosaics will be exported in JPEG format which can be read by common GIS systems. These mosaics may be combined within the GIS to generate

broad area coverage maps. The GIS selected for this paper is ARCGIS 8.1 from ESRI Corporation.

3.5.3 Conclusion

TOWCAM is a system designed to provide data for use by biologists and geologists. In the next chapter, a system to generate mosaics from the video, requiring limited operator input, will be described. These mosaics could prove useful in naval seabed surveillance pursuits.

Chapter 4 – Creating Video Mosaics From TOWCAM

4.1 Introduction

This chapter describes, in detail, the generation of a video mosaic from TOWCAM data collected aboard CCGS HUDSON on Sable Island Bank in October 2003. TOWCAM was not designed as a video mapping platform and so must not be expected to provide precision, geographically registered mosaics. The mosaic image is assembled using data from the sensors as fitted. The mosaic is projected to a flat earth plane with an approximate geographic position. In this chapter:

a. the sensor data logging system is discussed,

b. the camera calibration procedures is described; and

c. a detailed description is made of the data processing flow and software developed to generate a mosaic; and

The chapter concludes with a demonstration of an example mosaic generated using the procedures described.

4.2 Positioning and Attitude Sensing

The location and orientation of the camera is fundamental to the process. Serial navigation and attitude data strings from the GPS receiver, the ORE Trackpoint, and the ship's gyro compass are multiplexed into a common file on a first-in first-out (FIFO) basis. All navigation and telemetry data from the TOWCAM are recorded to the audio channel of the video file using a one-of-a kind modulator home built at the Bedford Institute of Oceanography (BIO) and are tagged with the GPS time to allow correlation of the data. TOWCAM is fitted with a flux-gate magnetic pitch and roll sensor but lacks a

heading sensor. The lack of camera heading data is problematic for geo-registration of the video frames. TOWCAM is also fitted with a pressure depth sensor, an altimeter, and an acoustic positioning beacon. The applicable offset and alignment information is contained in Appendix 1.

4.3 Camera Calibration

It is important to have an understanding of the image distortion inherent to the camera. The video camera in use lacked any technical documentation with respect to the CCD array size and the lens focal length. The video frames are assembled from the image projected onto the CCD array. Thus, the CCD array dimension was assumed to correspond to the output video frame pixel dimension (in this case 480 rows by 720 cols). In order to determine some usable lens characteristics, a simple pinhole camera model was assumed. An angular Field of View (FOV) was determined by imaging an object of known dimension from a known distance and radial distortion correction parameters could be established.

4.3.1 Calibration Procedure

A series of reference frames were collected by mounting the camera at a measured orientation and distance from a reference. The video camera from TOWCAM was made available to the author for a short period of time. A laminated paper grid with lines at 5cm interval and tick marks at one cm interval was generated to act as a calibrated reference plane. In retrospect, the lamination was to prevent or reduce humidity-induced distortion, however this created significant glare in the video (Figure 4-2). The DV Camera was mounted on a construction-grade self-leveling laser line level (Figure 4-1). All alignments and measurements were performed using construction grade

instruments (ie rafter square, bubble level and measuring tape). The DV Camera was inside an underwater housing and was not removed for the calibration. The collars on the underwater housing were of different diameters, a wooden shim was placed between the camera tube and level to keep the camera in parallel. The altitude measurements were made to the back edge of the lens window collar on the housing.

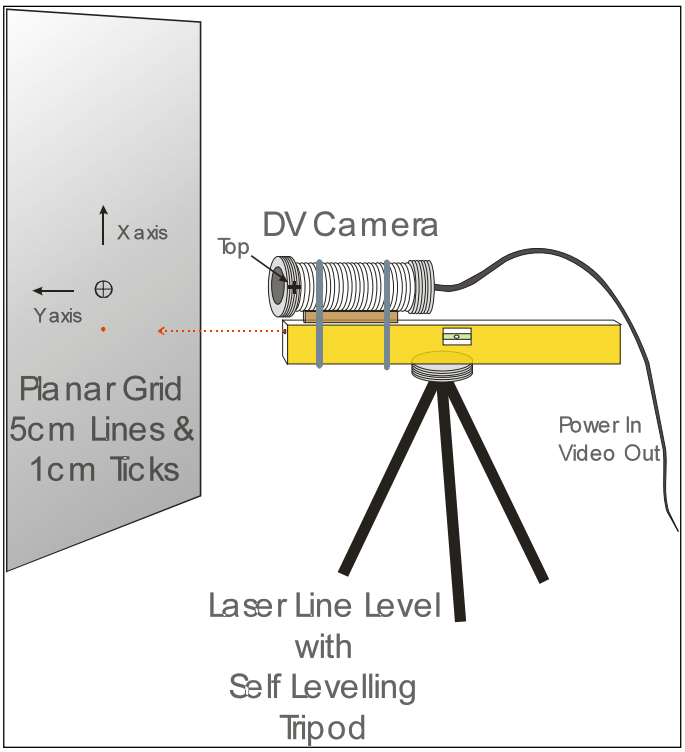

Figure 4-1 – Basic setup used for quick video camera calibration

A series of short video clips were recorded orthogonal to the x-axis of the grid at various angles of pitch along the y-axis. Owing to the wide-angle view of the lens, these shots were made at 20cm from the grid plane.

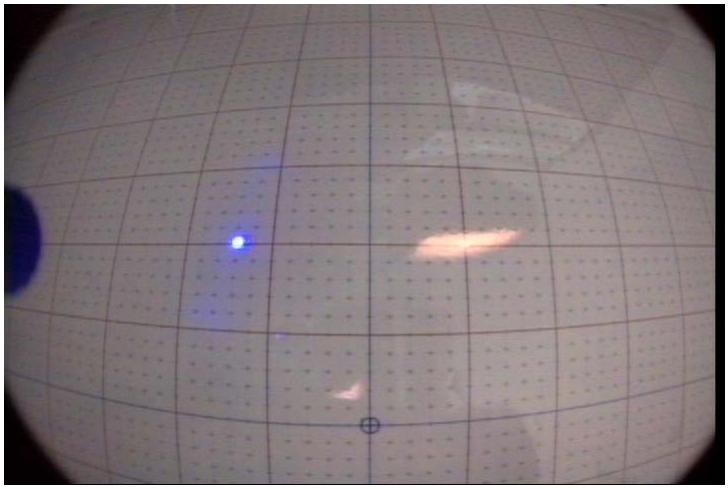

Figure 4-2- Video Frame of planar grid taken from 20cm height at 26.6 degrees pitch. The camera has been rotated 90º from vertical and the laser dot is 6.8cm to the left of bore site.

4.3.2 Calibration Results

The video was recorded to a DV mini Cassette Deck and subsequently transcribed to DVD format. Video frames were exported from the DVD using a software package called TRANSCODE version 0.67 [Bitterberg, 2003] running under LINUX Operating System [Mandrake, 2004]. The frames were exported in "ppm" format at 720 by 480 pixels with no data being recorded in the right margin of the video from pixel 710 to 720. Calculating two times the arctan of the observed grid value in the orthogonal image divided by the measured altitude of the camera, the diagonal field of view in air was determined to be 128º. Applying a refraction index for water of 1.33, the refracted diagonal FOV for the camera in water was estimated as 96º. The refracted FOV values seemed justified by comparison to the 1.67m x .27m mine in the underwater video frame shown in Figure 2-8.

4.3.2.1 Radial Distortion

The wide-angle view of the camera leads to radial distortion in the outer areas of the image. This distortion is greatly exaggerated by the oblique view of the video system

and so must be compensated for. Figure 4-3 exhibits the effect of barrel distortion from an orthogonal view while Figure 4-4 shows the exaggeration induced by an oblique view.

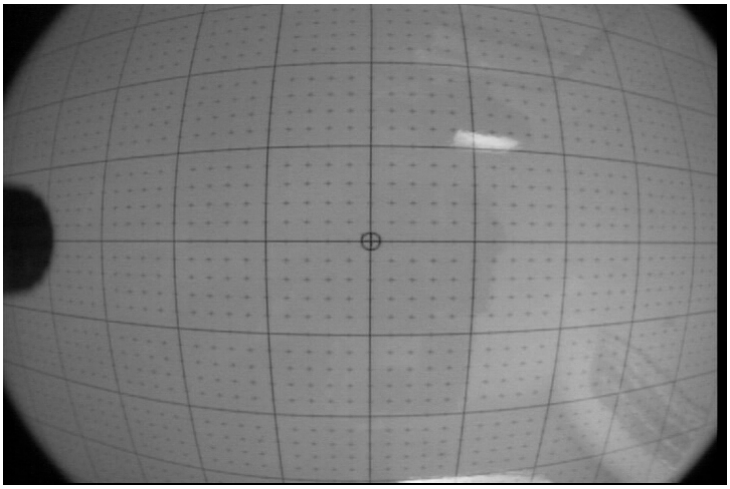

Figure 4-3 Preliminary calibration image of laminated paper grid taken orthogonal to vertical and horizontal axis at 20 cm above grid plane. The grid ticks are at 1cm interval and lines are at 5cm intervals. Note the characteristic barrel radial distortion.

The original image taken orthogonal to frame center from a height of 20 cm measures 480 pixel rows by 720 pixel columns. The vertical dimension imaged is 29.2.cm, while the horizontal dimension imaged is 49cm. The average vertical pixel angle is equal to arctan (14.6/20) divided by 240 = 36.129/240= 0.15º.

The average horizontal pixel value is equal to arctan (24.5/20) divided by 360 = 50.774/360= 0.14º. The scaling factor from horizontal to vertical is 15/14=1.07.

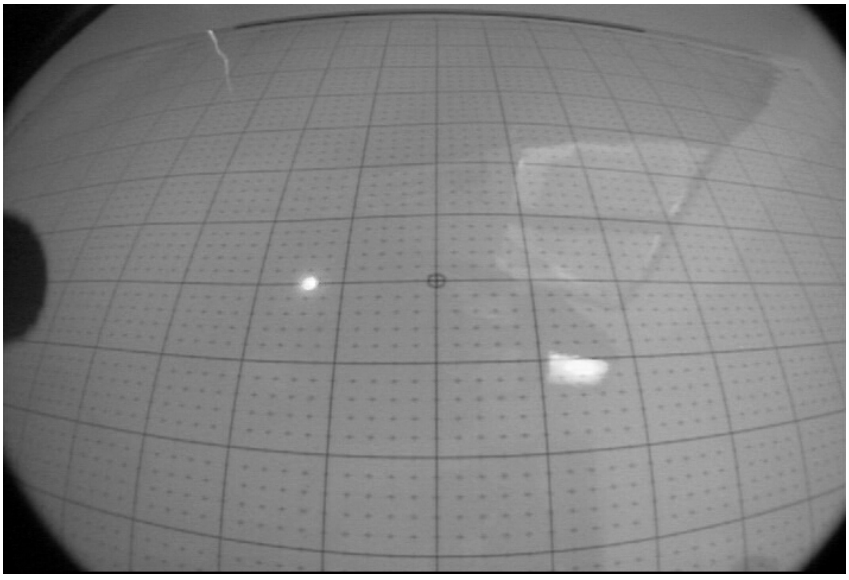

Figure 4-4 - Laminated paper grid imaged obliquely from an altitude of 21cm above grid plane at a pitch angle of 35 degrees. Note the presence of some dimpling in the grid due to its not being perfectly flat.

The polynomial radial distortion correction function as described in 4.5.4.1 was applied on a trial and error basis to determine reasonable values for the k_1 and k_2 parameters. The initial tests were applied to the orthogonal view at Figure 4-3 focusing on achieving satisfactory values for the central 100 rows of data. These results were then applied to the oblique view at Figure 4-4 which more closely represents the perspective to be experienced in TOWCAM operations. Figure 4-5 is the resultant image of Figure 4-4 re-projected to a plane using the polynomial radial distortion correction function described in equations (1) and (2) of paragraph 3.3.1.

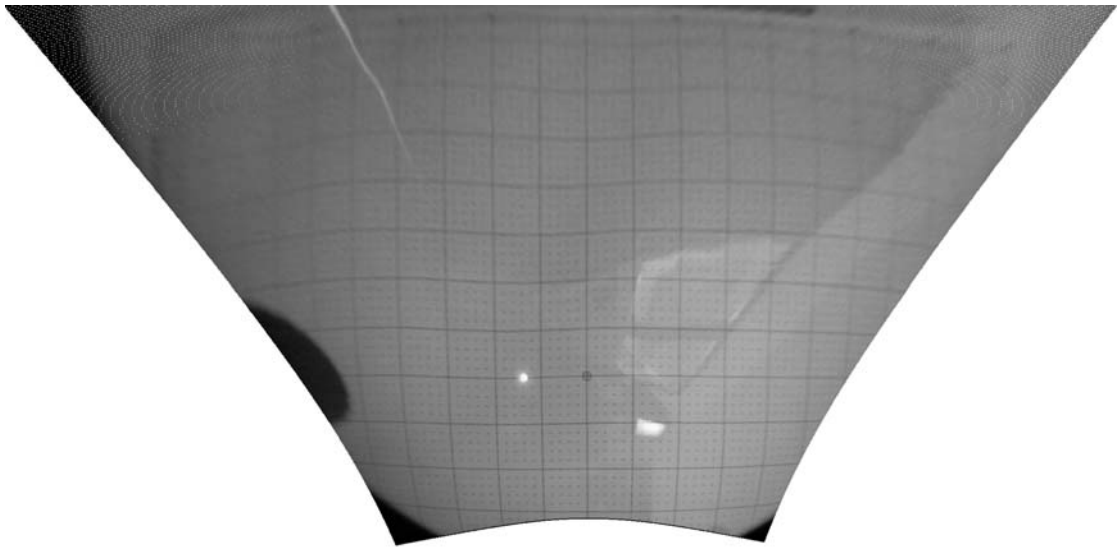

Figure 4-5 – Image from Figure 4-4 projected to a flat plane with radial distortion correction applied. The results are particularly good for the central rows where the grid lines are once more parallel and the grid spacing has become uniform. Note that the dimpling visible in the grid is accentuated.

4.3.2.2 Radiometric Correction

A Dark Reference image was not available and thus the DR image correction described at 3.4.1.1 was not applied.

The quality of the final mosaic image was enhanced by balancing the level of illumination across the video frames. Underwater lighting is provided by artificial illumination of HMI lights to augment ambient light. The sample image frame in Figure 4-6 displays a roughly circular radiance pattern. The final mosaic will contain only a few scan lines from the center of the image frames and thus a simple radial correction algorithm generating a corrected Digital Number (DN) value based on range from image center was chosen.

Figure 4-6 Video frame showing roughly circular (elliptical) radiance pattern from HMI illumination.

4.4 TOWCAM Data Description

The positioning and attitude data relevant to the system must be extracted and cleaned and then correlated with the applicable video frame.

4.4.1 Navigation and Telemetry Logging

All shipboard sensor serial data is multiplexed on a first-in first-out basis (FIFO) and logged to a file with an extension ".03e". The predominant serial format in this file is of the NMEA 0183 comma-delimited variety and is recorded at various rates as output by the particular sensor. The logging package employed is Regulus Survey© which is also employed by the ship as an electronic chart and navigation package. In addition it may be employed by other simultaneous activities and thus there are normally many other serial streams multiplexed that are irrelevant to the TOWCAM operation which must be filtered out.

```
$GPVTG,298,T,319,M,03.5,N,06.4,K*42
$HEHDT,287.4,T*26
$GPGLL,4404.41,N,06054.89,W,145311,A*3C
$GPZDA,145313,08,10,2003,03,00*42
$HEROT,-000.7,A*01
$GPRMB,,,,,,,,,,,,*66
$HCHDT,,T*07
$PASVW,03.5,A*1A
$GPGGA,145311,4404.4128,N,06054.8922,W,2,05,02.3,14.0,M,-21.3,M,03.5,0335*6C
$GPRMC,145311,A,4404.41,N,06054.89,W,03.5,298,081003,020.8,W*59
$IIVHW,287.6,T,,M,,N,,K*70
$VWVHW,,T,,M,2.8,N,5.2,K*59
$WIMWV,332,R,031,N,A*23
2 14:53:25 358 175.9    53.2      3.2    -45.2    28.0    654.1      0.4  -0.5
$POREB,2,145325,0,175.9,53.2,3.2,-45.2,28.0,0,0,0.4,-0.5*00
```

Figure 4-7 - Example lines of serial data found in a ".03e" file. The second last line is in the ORE native format, the last line shown has been re-written by the logging software as an NMEA 0183 record with the "POREB" identifier string.

4.4.1.1 Shipboard Navigation Data

The NMEA 0183 record IDs that are of interest include:

a. GPZDA for year, month and day;

b. GPGGA for time and position;

c. GPVTG for speed and course made good;

d. HEHDT or AGHDT for ship gyro heading; and

e. POREB for acoustic positioning of the TOWCAM vehicle.

It is of note that the ORE Trackpoint data is received and written in an ORE space-delimited format but is also re-written by the logging package with a user-specified NMEA 0183 talker and record identification of "POREB". The time written in this string is ORE System time which does not necessarily equate to GPS Time.

4.4.1.2 TOWCAM Telemetry Data

The sensor data from the TOWCAM vehicle and control system is recorded along with the most recent GPS Time. Fields of particular interest include pitch, roll, altitude

and depth. Unfortunately the vehicle does not have a heading sensor. The data is modulated and recorded to the audio channel of the Digital Video recording. The data is logged at a rate of one Hertz. Once the data is demodulated and exported to a disk file for processing, the file is given the extension ".03T". It commences with two lines of header data followed by sensor data in a comma-delimited text format (Figure 4-8).

```
Hudson Cruise: HUD2003059 Station: Sable Hot Hot
Recording session started on 08-Oct-2003
14:38:10, 10.48, 6.17, -14.17, 2.16, 43.07, 143759, 2.90, 2.50, 0.00, 60.90, 28.00, A, 1.90, 2.83, 0.00
14:38:11, 10.45, 11.33, -13.25, 2.05, 43.18, 143800, 2.90, 2.50, 0.00, 61.30, 28.00, A, 1.90, 2.78, 0.00
14:38:12, 10.45, 17.17, -14.17, 1.99, 43.29, 143801, 2.90, 2.50, 0.00, 61.30, 28.00, A, 1.90, 0.00, 0.00
```

Figure 4-8 First five lines from a ".03T" TOWCAM data file showing the two header lines followed by comma-delimited sensor records. The time in the first field is TOWCAM system time, GPS Time is found in the seventh field.

4.4.2 Video Data Recording

The TOWCAM Video is recorded to Digital Video (DV) Tape. The Video frames are superimposed with GPS time and ships position from the NMEA GGA string output from the GPS Receiver (Figure 4-9). The GGA string updates at 1 Hz; given a frame rate of 29.97Hz, 30 consecutive frames are tagged with the same position and time. The time/position string is written over the frame in rows 50 through 70. The string image commences with the six digit time in an HHMMSS format from column 50 through column 140. This video tagging provides the only point of reference for correlation of the telemetry and navigation data with the video frames.

Figure 4-9 Video frame showing time and position stamp. The position given is at the GPS Antenna, the time provides a tag to the navigation and telemetry data.

To permit video processing the NTSC Colour Video is exported from the DV Tape to DVD-R in MPEG2 format at 29.97 frames per second using a Pioneer DVD Recorder. The files are written with a ".VOB" extension.

4.5 TOWCAM Data Processing

All processing was performed using a PC fitted with a Pentium III (1000) CPU, 512mb RAM, DVD reader and 80GB disk drive. The LINUX Operating System was chosen owing to the broad range of readily-available, openly-accessible and flexible tools to manage and process these types of data. The open-source software packages employed included:

a. the GNU implementation of Awk and C-Shell Scripting for re-formatting of ASCII Data Records,

b. Transcode for video file access and frame export; and

c. NETPBM Tools and ImageMagick for bit map display, conversion and manipulation.

OMG Tools and additional programs were written in C to tie these tools together and re-project video images into geo-registered mosaics. The regular format of the text-based

navigation and telemetry data files allows for the use of shell scripting to extract pertinent data fields and re-combine them into coherent time-based records. The analog time stamp in the video data provides an avenue for the correlation of the text and video frames. The steps involved in generating a mosaic from a selectable time sequence are shown in Figure 4-10 and will be described in detail in subsequent sections. These steps were linked together and executed from the program called "mosProc.c" (see Appendix 2 section A2.1).

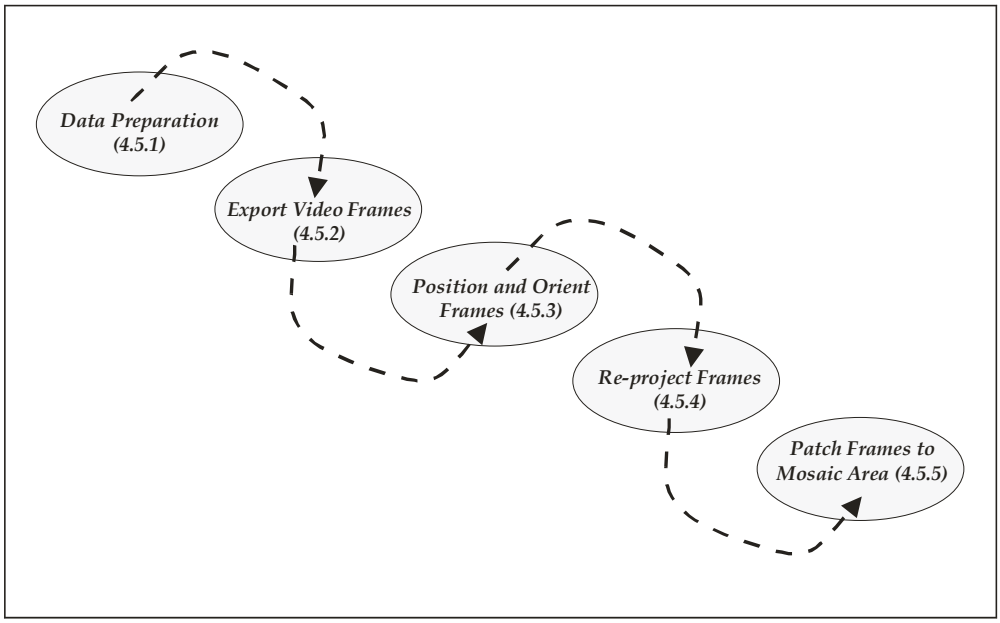

Figure 4-10 - Data Processing Sequence presented as five separate steps. The associated section numbers in parentheses are provided for reference.

4.5.1 Data Preparation

The processing strategy followed was to pre-process the data by organizing data files into a common directory tree as shown in Figure 4-11, and establishing time-based correlation of Video and ASCII data. Once the data was organized in this fashion, a series of custom-written scripts and programs were called from "mosProc.c" to execute the sequential steps.

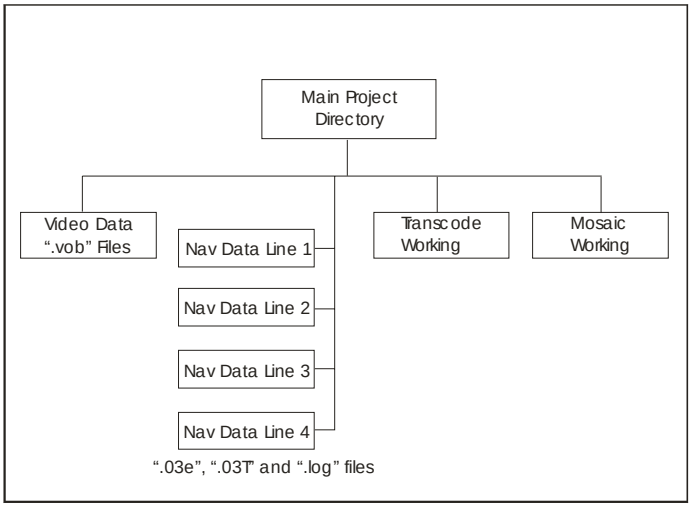

Figure 4-11 - Layout of project directory tree showing data and working sub-directories

A project directory tree was created with individual subdirectories for each survey line's navigation and telemetry data, one for the video files, and two "working" sub-directories for the extraction, re-projection and mosaicing of the video frames. A comma-delimited text file named "Vid_Info.txt" was created in the main project directory, giving file name, frame start time, stop times, sub-directory containing navigation and telemetry data, and the name of the video log file for use by the processing system to correlate navigation and telemetry data with the individual frame. An example of this file can be found in Figure 4-12.

```
VTS_01_1,125401.67,130459,CON238_Line_TC1A_to_B/,nav_seek_01_1,
VTS_02_1,143728.5,145459,CON238_Line_TC6B_to_A/,nav_seek_02_1,
VTS_05_1,023243.7,024359,CON269_Line_TC1B_to_A/,nav_seek_05_1,
VTS_03_1,003653.7,004959,CON269_Line_TC6A_to_B/,nav_seek_03_1,
```

Figure 4-12 Example of "Vid_Info.txt" file stored in the main project directory.

4.5.1.1 Time Synchronisation

The correlation of the video and ASCII data was done by the manual examination of a set of sequential frames for time rollover to establish a temporal benchmark. Once

identified, this point was used to calculate the video start time based upon the known frame number and the video frame rate. Because of inherent delays in the serial output of the NMEA sentences, this provided the most precise marker available.

Figure 4-13 Cropped sections of two successive video frames showing time rollover artifact from 004049 to 004050.

In order to determine the time of the first frame, Transcode was used to export the first forty frames of video. These frames were visually read using ImageMagick's "display" command to cycle through each frame until ime rollover was identified in a frame (see Figure 4-13). The start time was then linearly back-calculated based upon frame number, rollover time and frame rate.

4.5.1.2 Random Access Video Frames

A method to randomly access a specific location in the video file was required, as sequential access to a video file containing hundreds of thousands of individual frames is highly inefficient. Transcode has a function called "tcdemux" which can optionally generate an indexing file for randomly seeking frames using the "-nav_seek" switch in Transcode. This function was employed on each file to generate an indexing file for each video file with a file name that included the video file name and a ".log" extension. The ".log" files were stored in the subdirectory together with the applicable navigation and telemetry data files.

4.5.1.3 Reformat and Combine Data

A shell script called Merge_Data (see Appendix 2 section A2.5) was written to strip and reformat pertinent navigation and telemetry data. The ASCII ship navigation ".03e" and TOWCAM telemetry ".03T" files for each survey line were prepared into two comma-delimited ASCII files. One file, named "mosTrack.txt", merged the navigation and telemetry data records that were logged at one Hz. The second, "ORE_Data.flt" contained the ORE Trackpoint data which was logged at 0.5Hz.

4.5.1.3.1 Extract Navigation Data

The GNU implementation of the pattern matching language called "GAWK" was employed. The ".03e" file was scanned based on the desired record lines which were then stripped of appropriate fields, and written to a file named "NMEA_Data.tmp" . An additional field for seconds since midnight was also created from the hours, minutes and seconds (HMS) time field. A separate file for the acoustic trackpoint data called "ORE_Data.tmp" was created using the "POREB" record along with the time field from the most recent "GPGGA" record.

4.5.1.3.2 Merge Navigation and Telemetry Data

The ".03T" record was then similarly stripped of the desired fields and re-written to a file called "Towcam_Data.tmp". The two ".tmp" files containing NMEA and Towcam data were merged based upon a common time field using the Linux "join" command and saved to "Merge_Data.tmp". The Latitude and Longitude fields were converted to decimal degree format with a precision of ten decimal places and written along with the rest of the record to the file "mosTrack.txt".

4.5.1.3.3 Median Filter APS Data

The Ultra Short Baseline (USBL) ORE Trackpoint system on Hudson exhibits an excessively noisy azimuth computation as apparent from the graph in Figure 4-15. In order to reduce this noise, the file "ORE_Data.tmp" was passed though the program "filtnav.c" (see Appendix 2 section A2.2) to apply a five point median filter to the ORE Azimuth field and saved as "ORE_Data.flt".

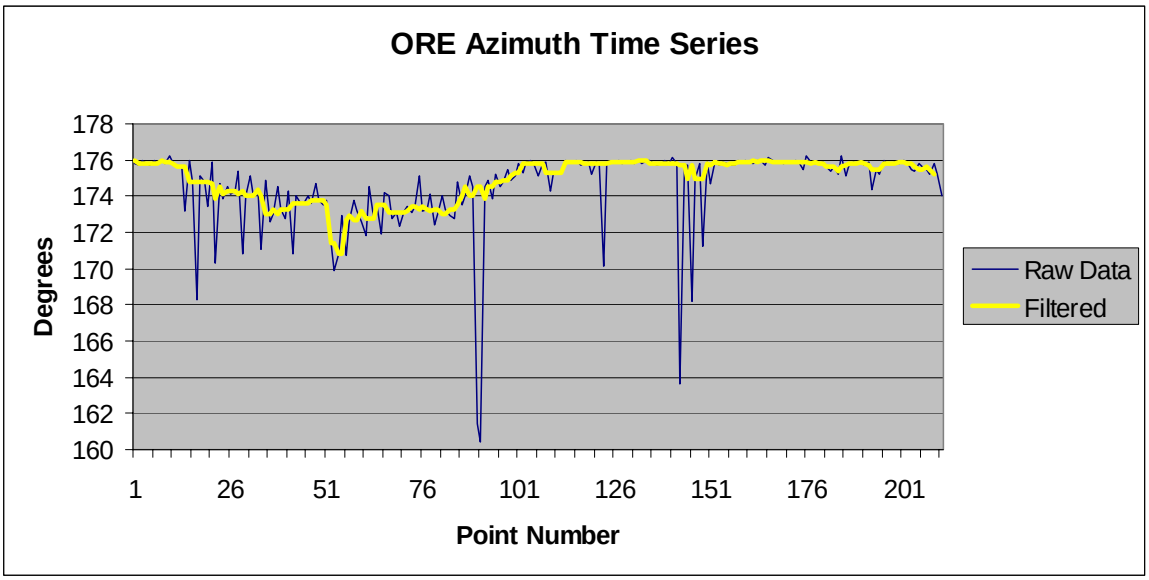

Figure 4-15 Graph compares a sample of five point median filtered ORE Azimuth data with raw logged data. The azimuth is relative to ship's heading. The raw data exhibits excessive noise, the data points are logged at an interval of two seconds (0.5Hz).

Figures 4-16 and 4-17 provide an example of, and details the record format used by the files "mosTrack.txt" and "ORE_Data.flt".

"mosTrack.txt" Record Format
Comma delimited fields, CR/LF separated records.

Sample Record:

"125401,44.0668300000,-60.9095583333,4404.0098,06054.5735,099,05.7,106.6,2003,10,08,46441,6.00,-11.17,1.81,46.80"

Field Name	Description	Example Value	Source Record
GPS Time	HHMMSS	125401	GPGGA
Latitude	DD.dddddddddd North + South -	44.0668300000	Calculated from GPGGA
Longitude	DD.dddddddddd East + West -	-60.9095583333	Calculated from GPGGA
North Latitude	DDMM.mmmm	4404.0098	GPGGA
West Longitude	DDDMM.mmm	06054.5735	GPGGA
GPS Course	DDD	099	GPVTG
GPS Speed	Km/Hr	05.7	GPVTG
Ship Heading	DDD.d	106.6	HEHDT
Year	YYYY	2003	GPZDA
Month	MM	10	GPZDA
Day	DD	08	GPZDA
Time	Seconds since midnight	SSSSS	calculated from GPGGA
Towcam Pitch	D.dd + nose up - nose down	6.00	".03T" Field 3
Towcam Roll	D.dd + Stbd Wing Up - Stbd Wing Down	-11.17	".03T" Field 4
Towcam Altitude	M.mm Metres	1.81	".03T" Field 5
Towcam Depth	M.mm Metres	46.80	".03T" Field 6

Figure 4-16 Record format for combined 1 Hertz Navigation and telemetry data . The GPS Time is only output by the receiver in integer seconds.

"ORE_Data.flt" Format
Comma delimited fields, CR/LF separated records

Sample Record

"125403,175.6,73.7,5.3,-68.0,28.0,0,0.4,-0.5"

Field Name	Description	Example	Source Record
GPS Time	HHMMSS	125403	GPGGA
ORE Azimuth	DDD.d	175.6	POREB
ORE Slant Range	MMM.m	73.7	POREB
X	MM.m metres	5.3	POREB
Y	MM.m metres	-68.3	POREB
Z	MM.m metres Manual Depth from ORE	28.0	POREB
Error Code	Error Num.	0	POREB
Ore Pitch	DD.d	0.4	POREB
Ore Roll	DD.d	-0.5	POREB

Figure 4-17 Record format for 0.5 Hertz Acoustic Tracking Data

4.5.2 Export Video Frames

A sequence of frames is exported by entering the video time of the start of the frames of interest and the number of seconds of video to be processed. The program "mosProc.c" identifies the appropriate video file from the Vid_Info.txt file, then generates a shell script called "tCode" which executes the transcode command to extract the sequence of video frames. The "tCode" script is stored in the TC_Working directory and is re-written each time the frame export routine is run, an example TCode script is shown in Figure 4-18. According to Lindley[1991], a video image is generated by two sweeps of an electron beam across the display surface of a CRT device. The display of the video frame is interlaced with two fields of video per frame. Standard NTSC video consists of 525 horizontal lines of video information updated 30 times a second. Since

one frame occurs thirty times a second, each field requires half of that time or 1/60 of a second to display. The 525 lines in a frame are split equally between the two fields, thus each field contains 262.5 lines. Each line requires approximately 63.5 microseconds to display. Of the 262.5 lines of video information per field, 18.5 lines are used for vertical blanking and display synchronization, leaving 244 lines per field or 488 lines per frame for visible image information. Of the lines used for vertical blanking up to twenty lines per frame may be used. The TOWCAM Video is recorded in color. For the purpose of generating the video mosaic, it was decided to use gray-level imagery. Export of the frames is slowed by a factor of two, however, the reduced size of the files (by a factor of 3) and coincident reduction in mosaic data processing volume justifies this decision.

```
transcode -i ./VIDEO_TS/VTS_03_1.VOB --no_split -y ppm,null -z -K -o ./TC_Working/ --
nav_seek ./CON269_Line_TC6A_to_B/nav_seek_03_1.log --frame_interval 1 -c 7081-7321
```

Figure 4-18 Example "tCode" shell script calling "transcode" to extract a sub-set of video frames.

The command line switches used in Figure 4-18 are as follows:

a. The –i switch identifies the input video file. The --no_split switch merges the stored interlaced video fields into a single frame,

b. The frames are exported in NetPBM's Portable Gray Map (PGM) format by setting transcode's -y switch to PPM Format and setting the –K switch which selects black and white mode,

c. The –z switch inverts the frames providing a heads up view of the frames which would otherwise appear upside down,

d. The frames are saved in the "TC_Working" subdirectory selected with the –o switch in numerically sequential file names starting at "000000.pgm". The ".pgm" frames exported comprised 704 columns and 480 rows of 8 bit data,

e. The --nav_seek switch enables the use of the designated “VOB” navigation file to permit random access to individual frames.

f. The –frame_interval switch set to “1” directs that all frames will be exported, and

g. The –c switch is followed by the start and stop frame number to identify the subset of frames to be exported from the video. The start and stop frame number are calculated in seconds from the first frame of the file using a frame rate of 29.97 frames per second.

ImageMagick’s “display” command was then used to sequentially display every fifth exported frame. This allowed the operator to verify that the appropriate video sequence had been output.

4.5.3 Position and Orient Video Frames

Horizontal positioning of the data is done in decimal degrees of Latitude and Longitude in WGS 84. Offsets are rotated to meters north and east, then subsequently converted to decimal degrees of Latitude, in the case of north offset, and decimal degrees of Longitude, in the case of east offset. The conversion from units of length to arc units on the WGS ellipsoid is performed by calculating the length in metres of a minute of arc at the local latitude on the ellipsoid. The C function shown in Figure 4-19 performs the approximation of this length on the elliptical arc.

```
/*LenLatMin and LenLonMin functions based on Admiralty Manual of Navigation Vol1 page 44/45 */
/*WGS params from NIMA TR8350.2 dated 4 July 1997 */
/*created by J Bradford Feb 2004 */

double LenLatMin(double lat) { /* Length in metres of a minute of Lat */

double a, f, e, Len, Q, x;

        x = lat*M_PI/180; /* convert to Radians */
        a =  6378137.00; /* WGS 84 Semi Major Axis */
        f = 1/298.257223563; /* WGS84 Flattening */
        e = sqrt((2*f-f*f)); /* eccentricity */

        Q=(1-e*e*sin(x)*sin(x));
        Len = a*(1-e*e)/sqrt(Q*Q*Q)*sin(M_PI/10800);/* length in metres */

        return(Len);
        }

double LenLonMin(double lat) { /* Length in metres of a minute of Long at a given Lat*/

double a, f, e, Len, Q, x;

        x = lat*M_PI/180; /* convert to Radians */
        a =  6378137.00; /*WGS84 Semi Major Axis */
        f = 1/298.257223563; /* WGS84 Flattening */
        e = sqrt((2*f-f*f)); /* eccentricity */

        Q=(1-e*e*sin(x)*sin(x));
        Len = a*cos(x)/sqrt(Q)*sin(M_PI/10800);/*length in metres*/

        return(Len);
        }
```

Figure 4-19 C functions LenLatMin and LenLonMin used to calculate the length in metres of a minute of latitude and longitude on the ellipsoid at a given latitude generated from formulae obtained from HMSO(1987).

4.5.3.1 Estimation of Camera Position

As described in paragraph 4.4.1, navigation and telemetry data are available at a sample rate of one hertz while acoustic positioning data are available at one half hertz (see paragraph 4.5.1.3). The program newNav.c (Appendix 2 section A2.3) reads these data into a structured array that brackets the time period covered by the frame selection. The video frame rate is 29.97 hertz, given the much lower data rates for the positioning and orientation data, linear interpolation is used to generate an integrated positioning and orientation solution for each frame. The offsets for shipboard and TOWCAM vehicle sensors are detailed in Appendix 1.

The absolute positioning accuracy of the GPS and Acoustic Positioning (AP) data is on the order of a few meters while the resolution of the imagery is a few millimeters. An averaged position solution is used in an attempt to manage this disparity. The initial position of the shipboard AP Transducer is calculated from the GPS solution with fixed offsets from the GPS Antenna to the AP Transducer applied and rotated using vessel heading input. Vessel pitch and roll data were unavailable and not applied. Subsequent positions of the AP transducer were computed using an average course and speed calculated from the data spanning the mosaic interval. These positions were then corrected for heading offset using a linear interpolation between epochs of heading data.

Position of the Towcam AP Transponder was calculated based upon an average azimuth of the AP Data for the entire period (generally ~30 seconds). This average was applied to the linearly interpolated vessel heading to generate a smoothed True Azimuth. The linearly interpolated slant-range data from the AP system were then converted to a horizontal range using the Towcam Depth measurement reduced by the AP Transducer depth (figure 4-20). The depth information provided by the AP system was ignored as this represented a manually input depth value, that was normally only updated at system start time. The horizontal range was used together with the True Azimuth to calculate the offset position in Latitude and Longitude of the TOWCAM AP Transponder.

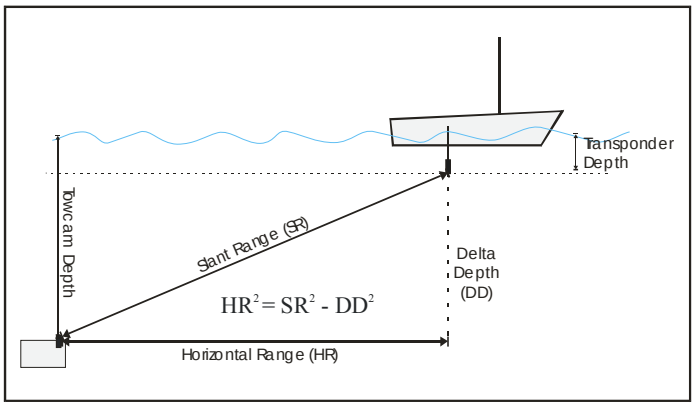

Figure 4-20 Horizontal range of TOWCAM is calculated with the measured depth reduced by the transponder depth. Sound velocity profiles are not available thus a fixed velocity is set and no refraction is applied.

The camera lens horizontal position was calculated by application of the offsets from the AP Transponder to the Lens. The offsets were rotated using interpolated pitch and roll from measured data and an estimated vehicle heading. The camera was operated at a low altitude (~2m) and aimed well forward of nadir (~25 degrees). One of the key assumptions employed in re-projection of the video frame is that the bottom is flat. This is often the case, however, uneven clutter conditions between those found at nadir and the camera image footprint can result in significant perspective distortion. Figure 4-21 depicts the inherent problem associated with varying cluttered conditions. In order to mitigate this problem, a predicted altitude at frame epoch was calculated by extrapolating the depth at a distance ahead corresponding to the location of the Optical Center of the image (given a nominal altitude of 2 meters). This altitude was corrected for offset from sounder transducer to lens with roll and pitch corrections applied. The estimated three-dimensional position along with the interpolated orientation data is written out as a "world file" with a filename corresponding to each video frame and ".pmw" as an extension.

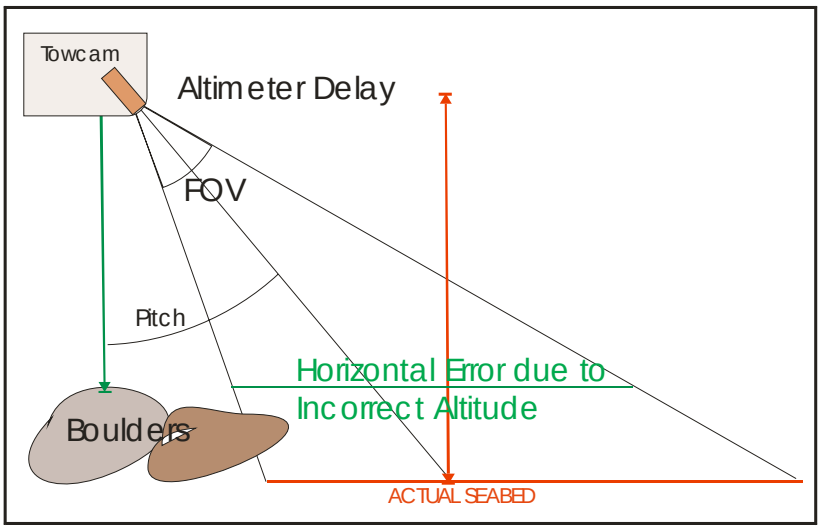

Figure 4-21 Horizontal Projection Error due to incorrect altitude is mitigated by applying a delay to the altimeter reading used, such that the altitude used is the altitude over the camera footprint.

4.5.4 Re-Project Video Frame

The program projNew.c (see Appendix 2 section A2.4) was used to project the image plane to a ground plane. Since camera heading (yaw) data was not available, the projection was made to a "heads up" orientation. The only rotations applied were for pitch and roll angle. Each frame was read sequentially, a radiometric correction applied, then the image was corrected for radial distortion, projected to the ground plane and then re-written to the Mosaic Working directory as a ".pgm" file. A line was added to the header of this file to include the data from the ".pmw" file. A weighted value file was correspondingly written as a ".pgm" file with the prefix "wts_". The frame-to-frame overlap is about 98%, the weighted value file was used to select the data from each frame for use in the final mosaic.

4.5.4.1 Radiometric Enhancement

The image data was radiometrically enhanced based on the radial distance (in pixels) of image pixels from the image center using the formula,

$$DN_{value} := DN_{value} + \left(\frac{Range}{Constant}\right)^3$$

where DN_{value} is in the integer range of 0-255, Range is the radial distance from image center and constant is a linear enhancement factor which through trial and error was set to a value of 100.

4.5.4.2 Correction of Radial Distortion

The radial distortion correction function from Derenyi[1996] is

$$r_d = k_0 r + k_1 r^3 + k_2 r^5$$

where r_d is the radial distortion, r is the radial distance and k_0 k_1 k_2 are coefficients determined through camera calibration. To correct the image coordinates for this distortion, the radial distance is computed as

$$r = (x'^2 + y'^2)^{1/2}$$

where x' and y' are the coordinates of an image point. The corrected x and y coordinates are then solved using

$$x = x'(1-r_d/r) \text{ and } y = y'(1-r_d/r).$$

4.5.4.3 Re-projection Function

The re-projection function is also adopted from Derenyi [1996]. The coordinates in the image plane are defined by u rows and v columns with the image center defined as (u_0, v_0). The focal distance f is the distance from the image center to the projection center of the lens. The two dimensional ground plane is defined using a three dimensional right hand coordinate system X, Y, Z; where Z = 0. The spatial coordinates of the projection center L are X_L, Y_L, Z_L. The orientation angles about the X, Y, and Z axes are defined respectively as roll (ω), pitch(φ) and yaw(κ). The rotation matrix M_r is defined as

$$M_r := \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

with elements

$$m_{11} := \cos(\phi)\cdot\cos(\kappa)$$
$$m_{12} := \cos(\omega)\cdot\sin(\kappa) + \sin(\omega)\cdot\sin(\phi)\cdot\cos(\kappa)$$
$$m_{13} := \sin(\omega)\cdot\sin(\kappa) - \cos(\omega)\cdot\sin(\phi)\cdot\cos(\kappa)$$
$$m_{21} := -\cos(\phi)\cdot\sin(\kappa)$$
$$m_{22} := \cos(\omega)\cdot\cos(\kappa) - \sin(\omega)\cdot\sin(\phi)\cdot\sin(\kappa)$$
$$m_{23} := \sin(\omega)\cdot\cos(\kappa) + \cos(\omega)\cdot\sin(\phi)\ \phi\cdot\sin(\kappa)$$
$$m_{31} := \sin(\phi)$$
$$m_{32} := -\sin(\omega)\cdot\cos(\phi)$$
$$m_{33} := \cos(\omega)\cdot\cos(\phi)$$

The image coordinates are projected to the ground plane using the collinearity equations adapted from Derenyi [1996] where

$$X := X_L + \left(Z - Z_L\right)\cdot\frac{\left[m_{11}\cdot\left(v - v_0\right) + m_{21}\cdot\left(u - u_0\right) + m_{31}\cdot(-f)\right]}{m_{13}\cdot\left(v - v_0\right) + m_{23}\cdot\left(u - u_0\right) + m_{33}\cdot(-f)}$$

$$Y := Y_L + \left(Z - Z_L\right)\cdot\frac{\left[m_{12}\cdot\left(v - v_0\right) + m_{22}\cdot\left(u - u_0\right) + m_{32}\cdot(-f)\right]}{m_{13}\cdot\left(v - v_0\right) + m_{23}\cdot\left(u - u_0\right) + m_{33}\cdot(-f)}$$

It may be noted above, that $Z\text{-}Z_L$ = the altitude of the camera lens. Once projected to the ground plane the image value is stored in an image array established on a pre-set pixel resolution of 1cm. The projected image array will have gaps in coverage which are filled by a routine called "fillquad" which is a "C" implementation of the Bresenham Algorithm [Bresenham, 1965]. The projected image, Figure 4-22, is then written out to file in ".pgm" format.

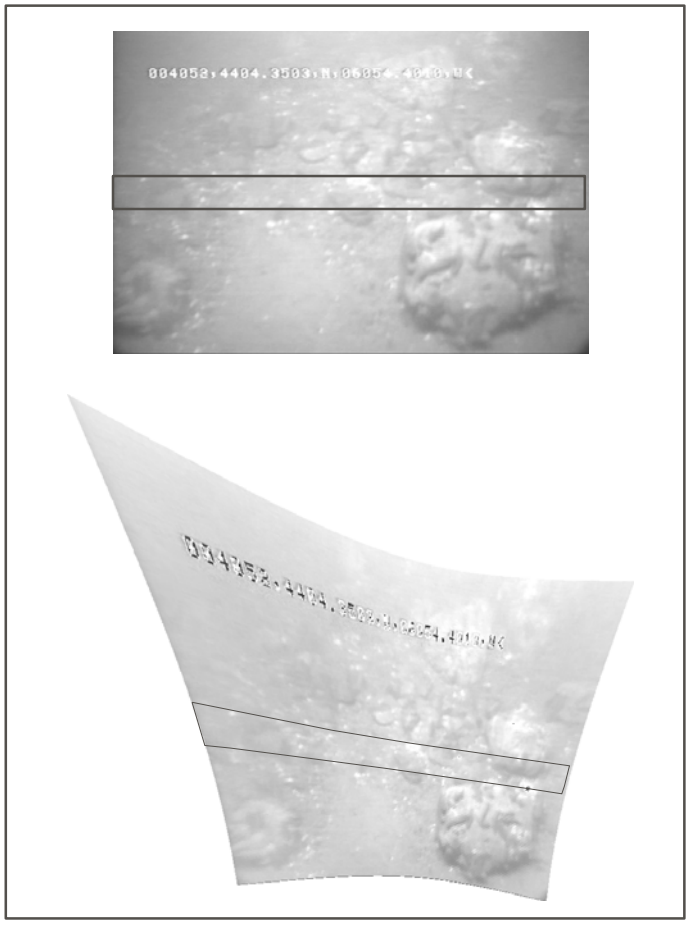

Figure 4-22 - Original video frame (top) and re-projected frame (bottom). The black rectangle in the original frame delimits the middle fifty rows of the image. The re-projected image has had radiometric and radial corrections applied.

In order to improve processing speed, and owing to the redundancy/overlap in coverage from frame to frame, only fifty rows from the center of the frame are re-projected for

mosaicing. A command line switch is available should one wish to adjust the portion of the image re-projected. The projected image is output with two additional comment lines in the header, the first to describe the camera lens xy-location in meters with respect to the top left corner of the image and the pixel resolution of the image, the second includes the data from the ".pmw" file with a $PMW prefix applied to the line.

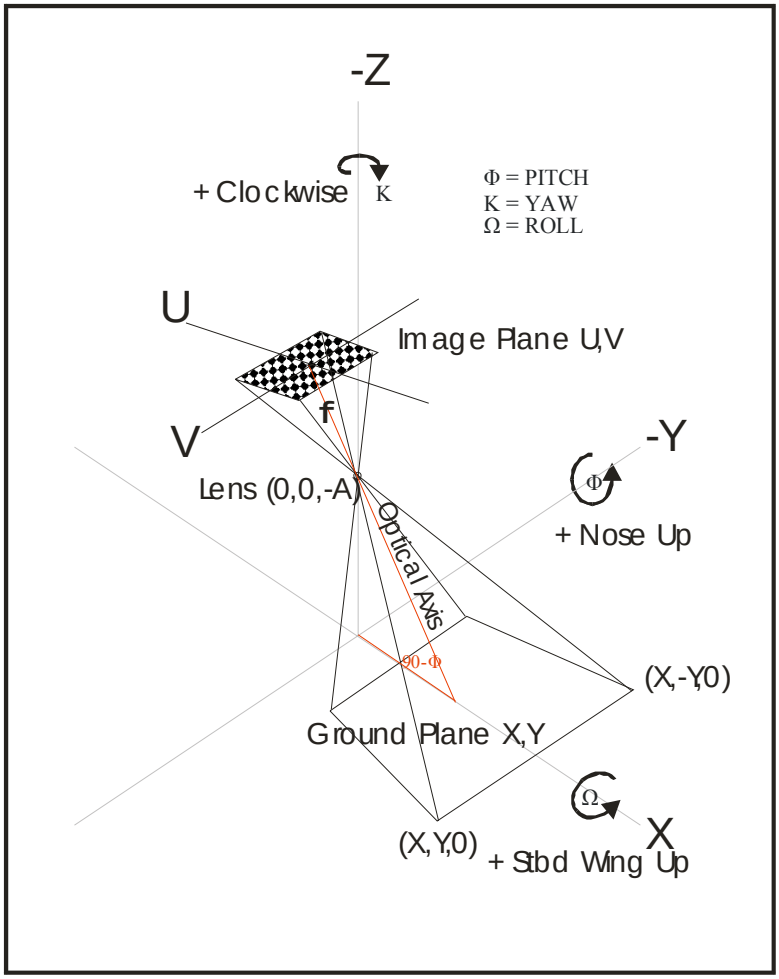

Figure 4-23 - Image and Ground Plane coordinate system parameters and orientation relationships.

4.5.4.4 Create Image of Weighted Values

An image of weighted values is generated for use during the mosaic process to provide a criterion for the selection of image data to be applied to the final mosaic image.

The physical dimension of the weighted image corresponds to that of the re-projected image while the image value is replaced with a weight value. An eight-bit weight value of the image is generated based on the pixel rows relationship to the middle row of the original image. The middle row is assigned the zero value and each subsequent row towards the top and bottom of the image is assigned a progressively larger value based upon the range of the middle pixel of the row from the center of the image. Figure 4-24 is the weighted value image for the frame described in Figure 4-22.

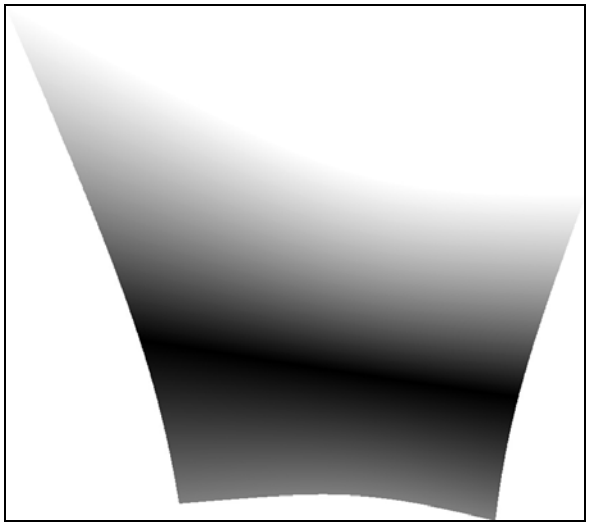

Figure 4-24 Image of weight values. Owing to their closer range to the central pixel row, the pixels in the lower half of the image are assigned a lower (more favourable) value than those in the upper half.

4.5.5 Patch Frames to Mosaic and Export

The re-projected frames were converted into OMG format and then a mosaic was created and exported as a geographically registered "jpeg" file which can be imported to ARCGIS for publishing purposes. An area for patching the mosaic together is established by using the shell script "corners.sh" (Appendix 2) to establish the extreme boundaries of the frame sequence identifying the coordinates of the upper left and lower right corners. The script then calls the OMG tool "make_blank" to generate the header file that will be used for each image. The header file was specified to use a world Mercator projection

and generate the image at 2cm resolution, which reduced file size and processing time by a factor of four. The reduction of resolution also compensated for sequential frame mis-alignments, with little effect on the ability to resolve targets in the final product.

4.5.5.1 Convert to OMG Format

The pgm files (Figure 4-24) are subsequently converted to OMG format using the OMG "tojhc" script, manually assigned an estimated towfish azimuth, and written out as ".mos" files. The estimated towfish azimuth was based upon the average calculated course made good of the TOWCAM vehicle and can be modified by application of an offset from the observed optical flow of the video sequence.

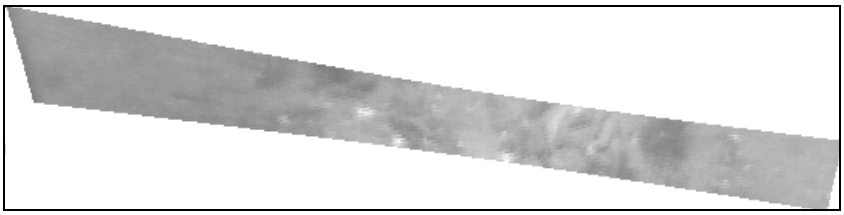

Figure 4-25 Re-projected section of video frame output by projNew.c. The section represents fifty middle rows from a single frame output at a resolution of 1cm per pixel. The orientation is camera head up.

4.5.5.1.1 Manual Intervention to Improve Frame-to-Frame Correlation

The use of an average speed in positioning of the camera along with potential scaling errors due to lens and sensor alignment and calibration inaccuracies will result in positional offsets between subsequent re-projected frames. This offset if large enough will distort or blur the final image. An improved relative positioning of the frames was achieved by comparing them sequentially in their geo-spatial context using the OMG "jview" utility. The location of distinct shell and pebble features (Figure 4-25) near the center of every tenth frame was noted and used to establish a horizontal translation (xy pixel shifts) for use in frame-to-frame registration. Discrete points on boulders were not used as their elevation resulted in significant frame-to-frame displacement due to parallax.

These translations were interpolated to develop an xy shift for each frame and the ".pgm" files were re-written with the xy shift applied in the header. This proved to be a time-consuming effort. For the sake of economy of effort this procedure was routinely by-passed, as satisfactory results were achieved once adequate smoothing of the telemetry had been applied.

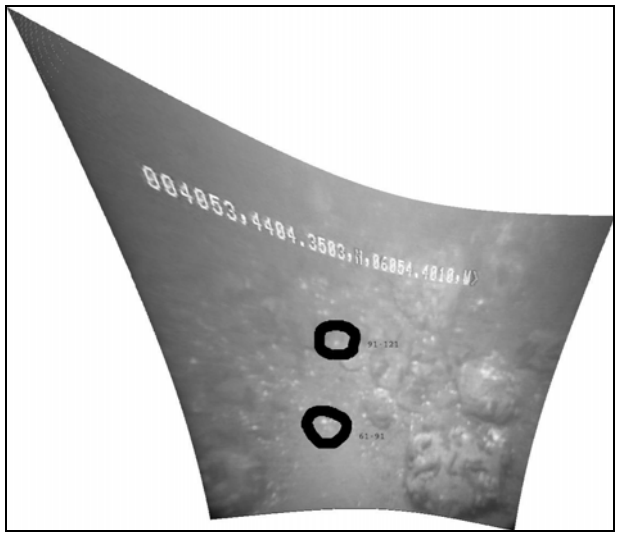

Figure 4-26 Radial distortion corrected frame, projected to ground plane using pitch, roll and camera altitude parameters. Gaps between data points in the projected image have been filled using the Bresenham Algorithm. Distinctive features are picked to compare frame to frame registration. The circled features are shells on the seabed that pass near frame center. The annotation is the sequence of frames for which the particular object was used as the tracking point.

4.5.5.2 Patch Frames to Mosaic

The OMG "patch_area" utility was then used to assemble the mosaic by sequentially reading the image frames and their corresponding weighting file to select image data for writing to the mosaic image file. Once the mosaic was created, the OMG tool "jview" was called to display the image. A histogram of the image was created by toggling the "h" key, and an appropriate stretch was applied by using the left mouse to set the lower value and the right mouse to set the upper value.

4.5.5.3 Export Geo-registered Mosaic

The OMG tool "stretchacres" was used to apply a histogram stretch and export the mosaic to a "pgm" file. This file was converted to JPEG format using the "ImageMagick" "convert" command. A jpeg world file ".jpw" was generated using the upper left coordinate of the mosaic area, pixel resolution, and the row and column dimensions of the output file. The mosaics may now be read into a GIS application for display, analysis and presentation. A sample of individual mosaic sections exported as jpegs along with graphs of attitude data are contained in Appendix 3.

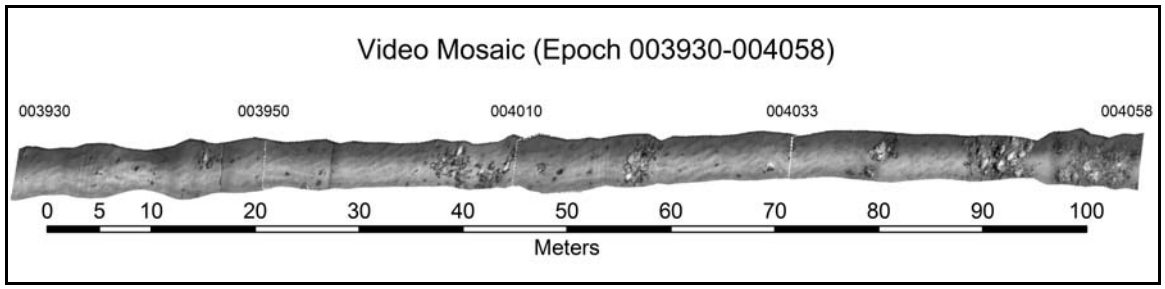

Figure 4-27 Composite of a series of four 25 sec mosaic sections tied to form an 88 second long mosaic. The mosaics were imported to ArcMap and seamed using the geo-referencing tool.

4.6 Conclusion

A mosaic has been generated from available sensor and video data based upon the entry by an operator of a randomly selected start time and length of video. In Figure 4-24 a 110 meter section of the seafloor ~5 meters in width is presented. Within this mosaic bed-forms, individual boulders and clusters of boulders are clearly displayed. If examined at its full resolution, the detail of individual scallop shells can be identified. This mosaic provides a meso-scale view of the seafloor with a resolution an order of magnitude better than that available from traditional acoustic data.

Chapter 5 – Conclusions and Future Directions

5.1 General

This report has described video mosaic production using available sensor data and simple correction routines to project the video image onto a plane. The horizontal position accuracies of these mosaics will be consistent with that achieved in the production of side scan sonar mosaics. The quality of the mosaic image in terms of resolution, positioning and coverage could be enhanced by additional instrumentation, increased sensor data rates, more thorough calibration and alignment routines, and the incorporation of software enhancements.

5.2 Recommended Instrumentation Enhancements

The following instrument additions, modifications and enhancements are suggested to improve the quality of mosaic production from TOWCAM. These improvements will also benefit the positioning associated with the still camera system.

5.2.1 Measure Camera Heading

The single simplest and least expensive enhancement that would have significant benefit is the addition of a heading sensor to the camera vehicle. The lack of heading data has necessitated the estimation of camera heading using the course made good which in itself is an estimation. The pitch and roll sensor currently fitted could easily be replaced by a commonly available flux gate heading, pitch and roll sensor. If fitted with a flux gate sensor it would be important to calibrate the heading by verifying the sensors fixed alignment and performing routine self-calibration procedures.

5.2.2 Measure Vessel Pitch and Roll

Horizontal positioning accuracy of the remote vehicle could be significantly improved by the addition of pitch and roll information for the towing vessel. A significant lever arm exists between the GPS Antenna and the Acoustic Tracking Transducer, only the vessel heading has been accounted for. If the pitch and roll sensor from the camera were replaced as per paragraph 5.2.1, it could be incorporated into the shipboard instrumentation to provide vessel pitch and roll data. These data would improve acoustic positioning solution.

5.2.3 Collect Data at Higher Frequency

Navigation and telemetry data are collected at one hertz (0.5 Hertz for Acoustic Positioning). The video frame rate is 30 Hz necessitating interpolation to match data points. Increased data rates would serve to improve the frame-to-frame motion solution as well as the camera positioning solution. Moderately priced GPS receivers with 5 to 10 Hz output rates as well as inexpensive heading, pitch and roll sensors with 10 Hz output are available. Output from the currently fitted pitch and roll sensor is at 1 Hz if the sensor is not replaced but is capable of a higher data rate then this should be used. TOWCAM is routinely operated at ranges of less than 150m (~0.2 second round trip time) from the acoustic tracking system and the interrogation frequency could therefore be increased from 0.5 Hz to the system maximum of 1 Hz.

5.2.4 Improve Camera Calibration

A rudimentary calibration of the camera out of water was performed, applying a simple refraction index correction to model underwater conditions. A more rigorous calibration conducted including in water calibration would be beneficial to account for

the refractive index. In water calibration could be performed in a pool setting or perhaps a rigid calibration grid and mounting frame could be constructed to attach to the vehicle for testing in harbor. A dark noise reference image should also be collected, preferably during survey operations in pressure and temperature conditions representative of the survey operation.

5.2.5 Add a Doppler Speed Log

Pitch and heave of the vessel are transmitted as motion via the tow-cable and induce local accelerations of the towed vehicle. A Doppler log would provide an avenue to compensate for these local velocity changes in the frame to frame motion solution. These logs are relatively expensive, its use could be supplanted by motion estimation techniques suggested in the next section.

5.3 Recommended Software and Processing Enhancements

The routines applied to produce the mosaic were “cobbled together” from open source software, self written C programs and shell scripts, and modifications to Ocean Mapping Group processing tools. Suggestions to enhance the final product, as well as to streamline and simplify its generation, are outlined in this section.

5.3.1 Use Image Motion Estimation

This project did not seek to use image motion estimation techniques, relying only on sensor based data, interpolations, medians and means. The Motion2D software library described at paragraph 3.2.1.2.1 could be incorporated to enhance frame to frame alignment.

5.3.2 Adjust OMG Header to Match Frame Mosaic

Each projected frame image is written to a mosaic file using the header for the entire mosaic coverage area. The individual frames cover a minute portion of the mosaic area and thus most of the frame mosaic contains no data. This practice is wasteful of disk space and significantly slows processing due to file size, disk access delay and the requirement to seek operator input. If each frame mosaic had a header corresponding to its coverage processing time would be significantly reduced.

5.3.3 Adapt to Common User Interface

The user interface is from command line entry and toggle screen selection. The sonar data at BIO is processed using a TK/TCL interface called AGCMENU. The processing routines described here could be modified to be accessed through AGCMENU thereby simplifying the users task of generating mosaics.

5.3.4 Incorporate Dark Reference with Radiometric Correction

A dark noise reference image was not available for application to the radiometric correction function. This reference image should be recorded and incorporated to the processing routine.

5.4 Future Directions

The utility of the mosaics in seafloor reconnaissance will serve to drive further improvements and development in this field. The increase in underwater visibility achieved by the LUCIE system promises a reliability of coverage that will make towed video into a tool for routine use in localized mapping and search operations. Further exploration of the use of TOWCAM in this type of role is warranted as it provides an inexpensive avenue to develop operational experience. This experience and expertise will

enhance the identification system operational and engineering requirements for an effective naval video seafloor surveillance system.

References:

Bitterberg, G.T., (2003), Transcode Processing, Online, http://www.zebra.fh-weingarten.de/~transcode/

Borgetto, M., V. Rigaud and J.F. Lots, (2003). *Lighting Correction for Underwater Mosaicking Enhancement*. IFREMER, Toulon, France

Bresenham, J.E., (1965). *Algorithm for Computer Control of a Digital Plotter*. IBM Systems Journal, 4(1):25-30

Caccia, M. , (1999). *Vision for estimating the slow motion of unmanned underwater vehicles*. Consiglio Nazionale delle Ricerche Instituto Automazione Navale, Genova, Italy

Canada, (1994). *White Paper on National Defence.* Online , Department of National Defence , Ottawa. www.forces.gc.ca/site/Minister/eng/94wpaper/white_paper_94_e.html

Canada, (2000). DREV Optical Oceanography, Active Underwater Imaging Presentation, Defence Research and Development Canada, CD-ROM

Canada, (2001). Defence Planning Guidance 2001, Online, Department of National Defence, Ottawa. www.vcds.forces.gc.ca/dgsp/pubs/rep-pub/dfppc/dpg/dpg2001/intro_e.asp

Canada, (2004), National Round Table on the Environment and Economy, Website: www.nrtee-trnee.ca/eng/programs/Current_Programs/Nature/Case-Studies/Essim-Case-Study-Brief_e.htm

Cannata, R.W., M. Shah, S.G. Blask and J.A. Van Workum, (2000). *Autonomous Video Registration Using Sensor Model Parameter Adjustments*. 29th Applied Imagery Pattern Recognition Workshop (AIPR'00), October 16 - 18, 2000 Washington, D.C.

Derenyi, E.E., (1996), *Photogrammetry: The Concepts*, 2nd Edition. Dept. of Geodesy and Geomatics Engineering, UNB, Fredericton, NB

Fournier, G.R., D. Bonnier, J.L. Forand, and P. Pace, (1993), *Range-gated underwater imaging system.* Optical Engineering, Vol. 32, pp. 2185-2190

Fournier, G.R., (2003). *Study of an Electro-Optic Sensor for Use in Identification and Gap-Filling on an Underwater Towed Vehicle.* DND Report ECR-2003-050, DRDC, Valcartier PQ

Fournier, G.R., D. Bonnier, J.L. Forand, and P.Pace, (1994). *LUCIE – laser enhanced underwater camer.,* Sea Technology, Vol 35, No.12, pp. 55-59

Gonzalez, R.C., and R.E. Woods, (1992). *Digital Image Processing*. Addison-Wesley, Reading, MA

Gordon, D.C., E.L.R. Kenchington, K.D. Gilkinson, D.L. McKeown, G. Steeves, M. Chin-Yee, W.P. Vass, K. Bentham, and P.R. Boudreau, (2000). *Canadian Imaging and Sampling Technology for Studying Marine Benthic Habitat and Biological Communitie.* ICES 2000 Annual Science Conference, Theme Session on Classification of Marine Habitats, CM 2000/T:07, Bruges, Belgium

Gracias, N., and J. Santos-Victor, (2001). *Trajectory Reconstruction with Uncertainty Estimation using Mosaic Registration*. VisLab-TR 04/2001, - Robotics and Autonomous Systems (Elsevier), 35(3-4), June 2001

Gracias, N., and J. Santos-Victor, (1998). *Automatic Mosaic Creation of the Ocean Floor*. Oceans '98, Nice, France, 257-262

Haywood, R., (1986). *Acquisition of a micro scale photographic survey using an autonomous submersible.* Proceedings of Oceans 86 Conference, New York

Heuer, J. and A. Kaup, (1999). *Global Motion Estimation in Image Sequences Using Robust Motion Vector Field Segmentation*. Proceedings ACM Multimedia 99, Orlando, Florida, 30 October - 5 November 1999, pp. 261-264

Jones, R.C., D. DeMenthon, D.S. Doerman, (1999). *Building mosaics from video using MPEG motion vectors*, LAMP-TR-035, Language and Media Processing Laboratory, Institute for Advanced Computer Studies, University of Maryland, College Park, MD

Karras, G.E. and D. Mavrommati, (2001). *Simple Calibration Techniques for Non-metric Cameras*. CIPA International Symposium, Potsdam, 18-21 September 2001

Klein, (1985). *Addendum to Klein Side Scan Sonar Record Interpretation Manual.* Klein Associates Inc., Salem, N.H.

Kumar, R., S. Samarasekera, S. Hsu, and K. Hanna, (1999). *Registration of highly-oblique and zoomed in aerial video to reference imagery*. Sarnoff Corporation, Princeton, NJ

Kumar, R., H.S. Sawhney, J.C. Asimuth, A. Pope, S. Hsu, (1998). *Registration of video to geo-referenced imagery*. ICPR '98, Brisbane Australia, 16-20 August, 1998

Lindley, C.A., (1991). *Practical Image Processing in C.* John Wiley and Sons Inc., New York

Li, X., (1999). *Photogrammetric Investigation into Low-resolution Digital Camera System.,* Thesis 6264, University of New Brunswick, Fredericton, NB

Mandrake Softwatre Ltd., (2004), Mandrake Linux Version 9.1, Online, http://www.mandrakesoft.com/

Marks, R.L., S.M. Rock and M.J. Lee, (1995). *Real-Time video mosaicing of the ocean floor*, IEEE Journal of Oceanic Engineering. 20(3) pp. 229-241, July 1995

Mather, P.M., (1999). *Computer Processing of Remotely Sensed Image,* 2nd edition. John Wiley and Sons, Toronto

McKeown, D.L.,(2003). Towcam Data Cover Letter, Personal Communication 15 December 2003

Odobez, J.M. and P. Bouthemy, (1995). *Robust Multiresolution Estimation of Parametric Models*. Academic Press, Journal of Visual Communication and Image Representation, Vol. 6,No.4, December 1995, pp. 348-365

Richards, J.A. and X. Jia, (1999). *Remote Sensing Digital Image Analysis,* 3rd edition. Springer-Verlag, Berlin

Reddy, G.L., (2002). Seabed Intervention – Requirement for a dedicated military seabed operations vessel. Canadian Forces College, CSC 28 Syndicate Paper, Toronto

Rzhanov, Y., G.R. Cutter, L. Huff (2000). *Sensor Assisted Video Mosaicing for Sea Floor Mapping.* Center for Coastal and Ocean Mapping (C-COM), University of New Hampshire, Durham

Pers, J. and S. Kovacic, (2002). *Nonparametric, Model-based Radial lens Distortion Correction using Tilted Camera Assumptio.* Faculty of Electrical Engineering, University of Ljubljana, Trzaska, Slovenia

Sawhney, H.S. and R. Kumar, (1997). *True Multi-Image Alignment and its Application to Mosaicing and Lens Distortion Correction.* IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 450-456, 1997

Sawhney, H., S. Hsu, and R. Kumar, (1998). *Robust Video Mosaicing through Topology Inference and Local to Global Alignment*. European Conference on Computer Vision, Germany, 1998

Slama C. (Ed.), (1980). *Manual of Photogrammetry, Fourth Edition*. American Society of Photogrammetry,1980.

Spindler, F. and P. Bouthemy, (1998). *Real-time estimation of dominant motion in underwater video images for dynamic positioning*. IEEE Int. Conf. on Robotics and Automation, ICRA'98, Volume 2, Pages 1063-1068, Leuven, Belgium

Trucco, E., Y.R. Petillot, I. Tena Ruiz, K. Plakas and D.M.Lane, (2000). *Feature Tracking in Video and Sonar Subsea Sequences with Applications*. Academic Press, Computer Vision and Image Understanding 79, pp. 92-122

Weidemann, A.D., G.R. Fournier, J.L. Forand, P. Mathieu and S. McLean, (2002). Using a Laser Underwater Camera Image Enhancer for Mine Warfare Applications: What is Gained?. Report to Canadian National Search and Rescue Secretariat, Ottawa

Williams, I. and J. Leach, (2000). *Video Remote Sensing for Seafloor Mapping*. 10th Australasian Remote Sensing and Photogrammetry Conference, Adelaide, Australia

Bibliography:

Brown, A., D. Sullivan, (2002), *Precision Kinematic Alignment Using a Low-Cost GPS/INS System,* NAVSYS Corporation, Proceedings of ION GPS 2002, Portland, Oregon

Hsu, S. H.S. Sawhney, and R. Kumar, (2001), *Automated Mosaics via Topology Inference,* Sarnoff Corporation

Cannata, R.W., Mubarak Shah, S.G. Blask, and J.A. Van Workum, *Autonomous Video Registration Using Sensor Model Parameter Adjustments,* Harris Corporation, Melbourne, Fla.

Gonzalez, M.G., P. Holifield and M. Varley, *Improved Video Mosaic Construction by Accumulated Alignment Error Distribution,* British Machine Vision Conference, Department of Engineering and Product Design, University of Central Lancashire, Preston

Spies, H., (2003), Introductory Perspective Geometry, Course Slides, Computer Vision Laboratory, Dept. of Electrical Engineering, Linkoping University, Sweden

Spies, H., (2003), Camera Models and Calibration, Course Slides, Computer Vision Laboratory, Dept. of Electrical Engineering, Linkoping University, Sweden

Spies, H., (2003), Parameter Estimation and Calibration, Course Slides, Computer Vision Laboratory, Dept. of Electrical Engineering, Linkoping University, Sweden

Williams, I.M., J.H.J. Leach, V. Wadley and B. Barker, (1998), *Creation of Models for the Measurement of Marine Species Using Along Track Video (ATV),* Oceans 98, Proceedings Vol. 3,pp 1797-1801, IEEE/OES, Piscataway, NJ.

Zhigang Zhu, A.R. Hanson, and E. M. Riseman, (2001), *Theory and Practice in Making Seamless Mosaics from Airborne Video,* UM-CS-2001-001, Computer Vision Laboratory, Dept. of Computer Science, Unversity of Massachusetts, Amherst

Zhigang Zhu, E. M. Riseman, A.R. Hanson, and H.Schultz, (1999), *Automatic Geo-Correction of Video Mosaics for Environmental Monitoring*, TR #99-28, Computer Vision Laboratory, Dept. of Computer Science, Unversity of Massachusetts, Amherst

APPENDIX 1- SENSOR OFFSETS

A1.1 Shipboard GPS and Acoustic Tracking Transducer Geometry

The MX412 GPS receiver was used by CCGS HUDSON throughout the cruise. The antenna is 25.0m forward and 2.9m to starboard of the Acoustic Tracking Transducer. The Acoustic Tracking Transducer's draft is 6.0m.

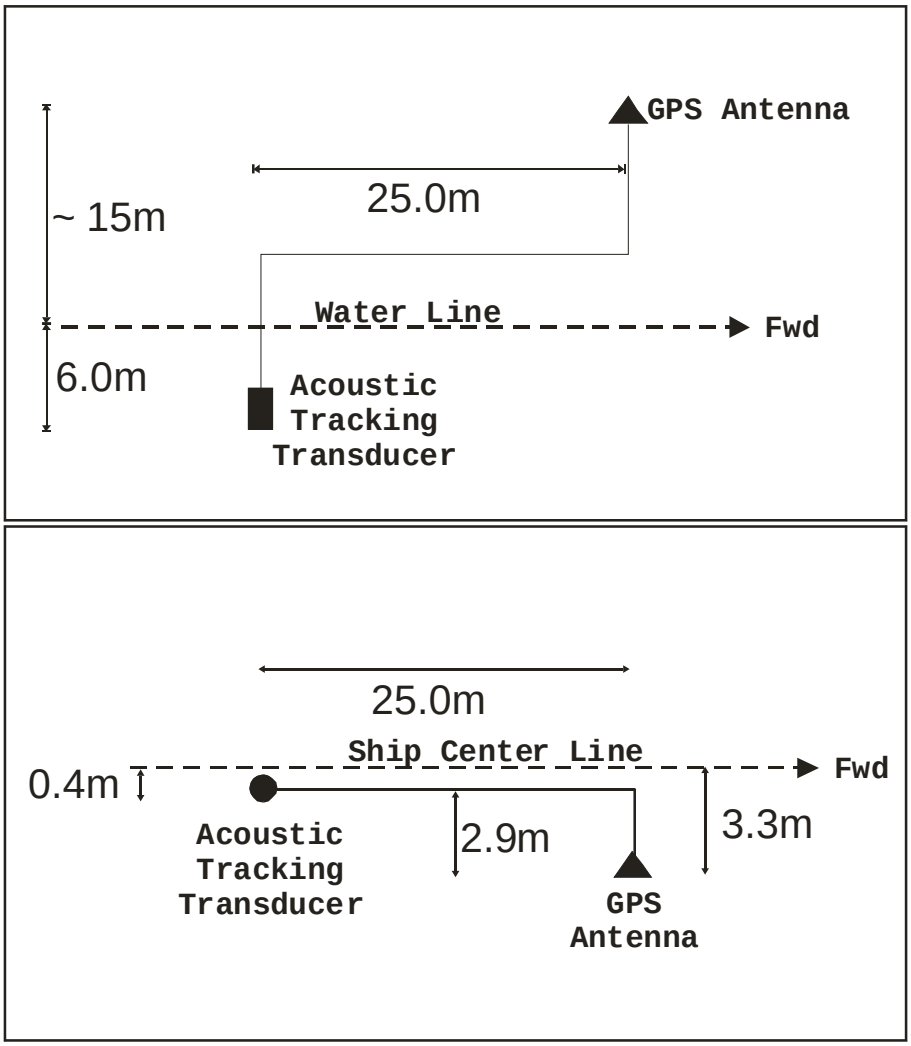

Figure A1-1 CCGS Hudson sensor layout, profile and plan views after McKeown 2003.

A1.2 TOWCAM Vehicle Geometry

Table A1-1 Towcam Sensor Offsets after McKeown 2003

Sensor	Fwd of Stern	Stbd of Centreline	Ht. Above Base	Angle
Altimeter	825.5mm	0mm	203.2mm	vertical
Video camera	260.4mm	0mm	95.3mm	25^0 fwd of vertical
Still camera	482.6mm	0mm	127mm	8^0 fwd of vertical
Strobe	1079.5mm	0mm	215.9mm	12^0 aft of vertical
Laser scale	292.1mm	0mm	158.8mm	25^0 fwd of vertical
P & R sensors	127mm	317.5mm	330.2mm	Horizontal
Acoustic Beacon	460.4mm	80mm	585.3mm	Horizontal

Note:
Distances are measured to lenses of optical devices, to altimeter transducer face and to after end cap of pitch and roll sensor pressure case. The video camera CCD is approx. 200mm aft of the Trackpoint beacon transducer, 80mm to port of it and 490mm below it.

Appendix 2 - Program Source Code & Shell Scripts

A2.1 mosProc.c

```
/* ------------------------------------------------------------ */
/*                  mosProc.c                                   */
/*                  TOWCAM Video Mosaic Processor               */
/*                  J. Bradford      2004                       */
/*                                                              */
/*                          UNB GGE                             */
/*                                                              */
/*                                                              */
/*                          1 Apr 04                            */
/*                                                              */
/*          This program provides a toggle
            selection menu to prepare the  nav and
            telemetry data, select video for export
            and project selected frames to a plane              */
/* ------------------------------------------------------------ */

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#include <string.h>



int startTime, mosLen, frameInt, option_select;

FILE    *infile, *outfile, *erroutfile, *vidinfo;

unsigned char in_strg[128] ="./";
char sysCall[256]= "transcode -i ./VIDEO_TS/";/*used by loadVidInf and
export_frame*/
const char delimiter[] = ",";
char *token;
char * line = NULL;
 size_t len = 0;
 ssize_t read;

typedef struct camera_telem{
                    int Time;
                    float Pitch;
```

```
                float Roll;
                float Yaw;
                float Alt;
                } camera_telem;

typedef struct vid_inf{
                const char *Name;
                float Begin;
                float End;
                const char *NavDir;
                const char *NavLog;
                } vid_inf;

vid_inf Vid;
camera_telem camAtt;

        float vidStartHr, vidStartMin, vidStartSec;
        float mosStartHr, mosStartMin, mosStartSec;
        double vidOffSec;
        int startFrame, numFrames;



main () {

/*----------------eo prelim file nauseau----------------------------------*/

/*----- Start of Program Flow --------*/
/*linearly interpolated data*/
/*startTime = 0;
mosLen = 5;*/
frameInt = 1;


option_page();

exit (0);

}


/*   end of main program *//*   end of main program *//*   end of mainprogram */

int err_exit(){

 printf("Exit due to Error, major bummer! \n");
```

```
exit (0);
} /* eo error exit */

/*---------------------------------------------------------------*/

int getTelem() {

        printf("%s\n", in_strg);
        strcat(in_strg,Vid.NavDir);
        printf("%s\n", in_strg);
        strcat(in_strg,"/mosTrack.txt");
        printf("%s\n", in_strg);

        infile = fopen (in_strg,"r");
        if (!infile) {
                printf (" Could not open %s for reading, bummer \n", in_strg);
                err_exit();
        }

while  (((read = getline(&line, &len, infile)) != -1) &&
       camAtt.Time != startTime) {

                /*printf("%s", line);*/
                token = strtok (line,delimiter);
                camAtt.Time = atoi(token);
                token = strtok (NULL,delimiter);/*decimal deg north*/
                token = strtok (NULL,delimiter);/*decimal deg east*/
                token = strtok (NULL,delimiter);/* Lat */
                token = strtok (NULL,delimiter);/*Lon*/
                token = strtok (NULL,delimiter);/*Hdg*/
                token = strtok (NULL,delimiter);/*smg*/
                token = strtok (NULL,delimiter);/*cmg*/
                token = strtok (NULL,delimiter);/*yr*/
                token = strtok (NULL,delimiter);/*mo*/
                token = strtok (NULL,delimiter);/*day*/
                token = strtok (NULL,delimiter);/*sec*/
                token = strtok (NULL,delimiter);/*attPitch*/
                camAtt.Pitch = atof(token);
                token = strtok (NULL,delimiter);/*attRoll*/
                camAtt.Roll = atof(token);
                token = strtok (NULL,delimiter);/*alt*/
                camAtt.Alt = atof(token);
                token = strtok (NULL,delimiter);/*depth*/
                }
```

```
        printf("Time: %d %.2f %.2f %.2f\n", camAtt.Time, camAtt.Pitch,
camAtt.Roll,camAtt.Alt);
        fclose(infile);
return(0);
} /* eo get telem*/

int loadVidInf() {

        vidinfo = fopen("Vid_Info.txt","r");
while  ((read = getline(&line, &len, vidinfo)) != -1) {
        token = strtok (line,delimiter);
        Vid.Name = token;
        token = strtok (NULL,delimiter);
        Vid.Begin = atof(token);
        token = strtok (NULL,delimiter);
        Vid.End = atof(token);
        token = strtok (NULL,delimiter);
        Vid.NavDir = token;
        token = strtok (NULL,delimiter);
        Vid.NavLog = token;

        if((Vid.Begin) < startTime && (Vid.End) > startTime) {

                fclose(vidinfo);
                printf("Video File Found = %s\n",Vid.Name);
                strcat(sysCall,Vid.Name);
                strcat(sysCall,".VOB --no_split -y ppm,null -z -K -o ./TC_Working/ --
nav_seek ./");

                strcat(sysCall,Vid.NavDir);
                strcat(sysCall,Vid.NavLog);

                printf("%s\n", sysCall);

                return(0);
                }
        printf("%s.VOB\t%.2f\t%.6d\t%.2f\n", Vid.Name, Vid.Begin, startTime,
Vid.End);

        }
        printf("Video File Not Found, exiting..\n");
        exit(0);
}/* eo loadVidInfo */
```

```
int option_page () { /*system("clear\n");*/
        printf("\n\t\t\t  TOWCAM Video Mosaic Processor\n");
        printf("\t\t\t\tJ. Bradford 2004\n\n");
        printf("\tSelection Menu:\n");
        printf("\n\t\t1\t-Merge Shipboard and Towbody Navigation and Telemetry
Data\n");
        printf("\t\t2\t- Select Video Clip Start and Length:\n\t\t\tStart Time=
%.6d\tLength= %d sec \n",startTime, mosLen);
        printf("\t\t3\t- Export Video Frames\n");
        printf("\t\t4\t- View Extracted Frame Animation\n");
        printf("\t\t5\t- Process Nav and link to Frames\n");
        printf("\t\t6\t- Project Frames and Insert Nav to Header\n");
        printf("\n\t\t0\t-Exit Program;\n\n\t\t");

        scanf("%d", &option_select);

        switch (option_select) {

        case 0 : exit (0);
        break;

        case 1 : prepNav();
        break;

        case 2 : printf("Enter New Start Time HHMMSS\n\t");
                  scanf("%d", &startTime);


         printf("Enter New Mosaic Length in Seconds\n\t");
                  scanf("%d", &mosLen);


        option_page();

        case 3 : loadVidInf();
                  getTelem();
                 getTimeBounds();
                 export_frame();
;
        break;

        case 4 :system ("echo Loading Sub-set of Frames standby...\n echo right click for
menu!\nanimate TC_Working/0*[1,6].pgm\n");

        break;
```

```
        case 5: system("waz2");

        break;
        case 6 : bulk_Proj();

        break;

        } /* eo switch */
printf("\n\tOption %d completed!", option_select);


        option_page();
        } /* eo option_page*/

int export_frame() {

system("rm -f ./TC_Working/*");
outfile = fopen("TC_Working/tCode","wb");

fprintf(outfile,"%s.log --frame_interval %d -c %d-%d \n",sysCall,
frameInt,startFrame,startFrame+numFrames);
fclose(outfile);
system("chmod +x TC_Working/tCode");
system("./TC_Working/tCode");

outfile = fopen("TC_Working/pgm.inf","wb");
fprintf(outfile,"%.2f\n",mosStartSec);
fprintf(outfile,"%d\n",numFrames);
fclose(outfile);

return(0);
}

int getTimeBounds() {


        printf("\n Video File StartTime (HHMMSS.ss)= %.2f\n",Vid.Begin);

        printf("\n Mosaic StartTime (HHMMSS.ss)= %d\n", startTime);

        vidStartHr = floor(Vid.Begin/10000);
        vidStartMin = floor((Vid.Begin-vidStartHr*10000)/100);
        vidStartSec = Vid.Begin-vidStartHr*10000-vidStartMin*100;
        vidStartSec = vidStartSec+vidStartMin*60+vidStartHr*3600;

        mosStartHr = floor(startTime/10000);
```

```
        mosStartMin = floor((startTime-mosStartHr*10000)/100);
        mosStartSec = startTime-mosStartHr*10000-mosStartMin*100;
        mosStartSec = mosStartSec+mosStartMin*60+mosStartHr*3600;

        vidOffSec = mosStartSec - vidStartSec;

        printf("Vstart %f\t mStart %f\t %f\n", vidStartSec, mosStartSec, vidOffSec);

        printf("\n Enter Time Length to be Processed (seconds)\n");

        while(mosLen <=0) {
                        scanf("%f", &mosLen);
                        }
        startFrame = floor(vidOffSec*29.97/frameInt);
        numFrames = ceil(mosLen*29.97/frameInt);
        printf("Start Frame %d\tNum Frames %d\n", startFrame, numFrames);
        return(0);
        }

int bulk_Proj() {

FILE *infilelist;
char *innamelist, *inname;
char *linelist = NULL;
        innamelist ="./TC_Working/pgmlist.txt";
        infilelist = fopen(innamelist,"rb");
        system("rm -f ./TC_Working/curProj");
        system("rm -f ./MOS_Working/*.*");
        while ((read = getline(&linelist, &len, infilelist)) != -1)     {
        inname = strsep(&linelist,"m");/* gets rid of linefeed return */


outfile = fopen("TC_Working/curProj","wb");
fprintf(outfile,"proj -in TC_Working/%sm -out MOS_Working/%sm\n",inname,inname);
fprintf(outfile,"proj -in TC_Working/%sm -out MOS_Working/%sm -wts
MOS_Working/wts_%sm\n",inname,inname,inname);
fclose(outfile);
system("chmod +x TC_Working/curProj");
system("./TC_Working/curProj");
linelist = NULL;
        }

return(0);
}
int prepNav(){
```

```
system("ls -R */*.03e > TC_Working/03eList.txt");
infile = fopen("TC_Working/03eList.txt","rb");

line = NULL;

char dircom[256];
char *syscom, *path;


while((read = getline(&line, &len, infile)) != -1){



strcpy(dircom,"cd ");
path = strsep(&line,"/");
strcat(dircom, path);
strcat(dircom,"\n../Merge_Data\n");/*shell script is below*/
system(dircom);
printf("%s\n",dircom);

        }/* EO while */
fclose(infile);
return(0);
}
```

A2.2 filtNav.c

```
/**********/
/* filtNav.c */
/* j bradford 29 Mar 2004 */
/* performs a five point median filter on ORE Azimuth */
/* called as part of Merge_Data script which in turn is called mosProc.c*/

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#include <string.h>

typedef struct mos_telem{
                int Time;
                double North;
                double East;
                double Lat;
                double Lon;
                float Hdg;
                float Smg;
                float Cmg;
                int Yr;
                int Mo;
                int Day;
                double Sec;
                float Pitch;
                float Roll;
                double Yaw;
                float Alt;
                float Depth;
                } mos_telem;

        mos_telem Nav0;
        mos_telem Nav1;
        mos_telem Nav2;
        mos_telem Nav3;
        mos_telem Nav4;
        mos_telem Nav5;



typedef struct ore_dat{
        int Time;
        double Az;
        double Rge;
```

```
        float x;
        float y;
        float z;
        int err;

        } ore_dat;

        ore_dat Ore0;
        ore_dat Ore1;
        ore_dat Ore2;
        ore_dat Ore3;
        ore_dat Ore4;
        ore_dat Ore5;

const char delimiter[] = ",";
char *token;
char * line = NULL;
 size_t len = 0;
 ssize_t read;

int counter =0;
int i, j;
int filterPoints =5;
double  a, temp;
double *data, *data1;

FILE *infile, *outfile;
char *inname, *outname;

main (){

inname = "ORE_Data.tmp";
outname= "ORE_Data.flt";
do_ore();

/*inname = "mosTrack.txt";
outname = "mosTrack.flt";
do_att();*/

exit(0);
}/*eo main*/



int do_ore(){
infile= fopen(inname,"rb");
```

```
outfile = fopen(outname,"wb");

data = (double *) malloc(filterPoints* sizeof(double));
while  (((read = getline(&line, &len, infile)) != -1)){

token = strtok (line,delimiter);
Ore0.Time = atoi(token);
token = strtok (NULL,delimiter);
Ore0.Az = atof(token);
token = strtok (NULL,delimiter);
Ore0.Rge =atof(token);
token = strtok (NULL,delimiter);
Ore0.x =atof(token);
token = strtok (NULL,delimiter);
Ore0.y =atof(token);
token = strtok (NULL,delimiter);
Ore0.z =atof(token);
token = strtok (NULL,delimiter);
Ore0.err = atoi(token);

counter++;

Ore5=Ore4;
Ore4=Ore3;
Ore3=Ore2;
Ore2=Ore1;
Ore1=Ore0;

if (counter > 3){
data[0] = Ore1.Az; data[1] = Ore2.Az; data[2] = Ore3.Az; data[3] = Ore4.Az;
 data[4] = Ore5.Az;

for (i= filterPoints-1; i >=0; i--){
        for (j=1; j<=i; j++){
                if (data[j-1] > data[j]){
                        temp = data[j-1];
                        data[j-1] = data[j];
                        data[j] = temp;
                                }
                        }
                }
```

```
fprintf(outfile,"%.6d,%.1f,%.1f,%.1f,%.1f,%.1f,%d\n",Ore3.Time,data[2],Ore3.Rge,Ore3
.x,Ore3.y,Ore3.z,Ore3.err);
 }
else if (counter < 2 )
{fprintf(outfile,"%.6d,%.1f,%.1f,%.1f,%.1f,%.1f,%d\n",Ore0.Time,Ore0.Az,Ore0.Rge,O
re0.x,Ore0.y,Ore0.z,Ore0.err);
 }
}/*eo while read*/
fprintf(outfile,"%.6d,%.1f,%.1f,%.1f,%.1f,%.1f,%d\n",Ore0.Time,Ore0.Az,Ore0.Rge,Ore
0.x,Ore0.y,Ore0.z,Ore0.err);
free(data);
fclose(infile);
fclose(outfile);
counter=0;
return(0);
}


int do_att(){
infile= fopen(inname,"rb");
outfile = fopen(outname,"wb");

data = (double *) malloc(filterPoints* sizeof(double));
data1 = (double *) malloc(filterPoints* sizeof(double));

while  (((read = getline(&line, &len, infile)) != -1)){

                token = strtok (line,delimiter);/* time */
                Nav0.Time = atoi(token);
                token = strtok (NULL,delimiter);/*Decimal Degrees Lat Lon (North
East)*/
                Nav0.North = atof(token);
                token = strtok (NULL,delimiter);
                Nav0.East = atof(token);
                token = strtok (NULL,delimiter);/*Geo Lat Lon DDMM.mmmm*/
                Nav0.Lat = atof(token);
                token = strtok (NULL,delimiter);
                Nav0.Lon = atof(token);
                token = strtok (NULL,delimiter);/* Heading Speed Cmg */
                Nav0.Hdg = atof(token);
                token = strtok (NULL,delimiter);
                Nav0.Smg = atof(token);
                token = strtok (NULL,delimiter);
                Nav0.Cmg = atof(token);
                token = strtok (NULL,delimiter);/* Y M D */
```

```
                Nav0.Yr = atoi(token);
                token = strtok (NULL,delimiter);
                Nav0.Mo = atoi(token);
                token = strtok (NULL,delimiter);
                Nav0.Day = atoi(token);
                token = strtok (NULL,delimiter);/* Seconds */
                Nav0.Sec = atof(token);
                token = strtok (NULL,delimiter);/* Pitch */
                Nav0.Pitch = atof(token);
                token = strtok (NULL,delimiter);/* roll */
                Nav0.Roll = atof(token);
                token = strtok (NULL,delimiter);/* alt */
                Nav0.Alt = atof(token);
                token = strtok (NULL,delimiter);/* depth */
                Nav0.Depth = atof(token);
counter++;

Nav5=Nav4;
Nav4=Nav3;
Nav3=Nav2;
Nav2=Nav1;
Nav1=Nav0;

if (counter > 3){
data[0] = Nav1.Pitch; data[1] = Nav2.Pitch; data[2] = Nav3.Pitch; data[3] = Nav4.Pitch;
 data[4] = Nav5.Pitch;
data1[0] = Nav1.Roll; data1[1] = Nav2.Roll; data1[2] = Nav3.Roll; data1[3] = Nav4.Roll;
 data1[4] = Nav5.Roll;

for (i= filterPoints-1; i >=0; i--){
        for (j=1; j<=i; j++){
                if (data[j-1] > data[j]){
                        temp = data[j-1];
                        data[j-1] = data[j];
                        data[j] = temp;
                                }
                        }
                }


for (i= filterPoints-1; i >=0; i--){
        for (j=1; j<=i; j++){
                if (data1[j-1] > data1[j]){
                        temp = data1[j-1];
                        data1[j-1] = data1[j];
                        data1[j] = temp;
```

```
                        }
                    }
                }


fprintf(outfile,"%.6d,%.10f,%.10f,%.4f,%.4f,%.1f,%.1f,%.1f,%d,%d,%d,%.0f,%.3f,%.3f,
%.3f,%.3f\n",
Nav3.Time,Nav3.North,Nav3.East,Nav3.Lat,Nav3.Lon,Nav3.Hdg,Nav3.Smg,Nav3.Cmg,
Nav3.Yr,Nav3.Mo,Nav3.Day, Nav3.Sec,data[2],data1[2],Nav3.Alt,Nav3.Depth);

 }
else if (counter < 2 )
{fprintf(outfile,"%.6d,%.10f,%.10f,%.4f,%.4f,%.1f,%.1f,%.1f,%d,%d,%d,%.0f,%.3f,%.3
f,%.3f,%.3f\n",
Nav0.Time,Nav0.North,Nav0.East,Nav0.Lat,Nav0.Lon,Nav0.Hdg,Nav0.Smg,Nav0.Cmg,
Nav0.Yr,Nav0.Mo,Nav0.Day, Nav0.Sec,Nav0.Pitch,Nav0.Roll,Nav0.Alt,Nav0.Depth);
 }
}/*eo while read*/

fprintf(outfile,"%.6d,%.10f,%.10f,%.4f,%.4f,%.1f,%.1f,%.1f,%d,%d,%d,%.0f,%.3f,%.3f,
%.3f,%.3f\n",
Nav0.Time,Nav0.North,Nav0.East,Nav0.Lat,Nav0.Lon,Nav0.Hdg,Nav0.Smg,Nav0.Cmg,
Nav0.Yr,Nav0.Mo,Nav0.Day, Nav0.Sec,Nav0.Pitch,Nav0.Roll,Nav0.Alt,Nav0.Depth);

free(data);
fclose(infile);
fclose(outfile);
counter=0;
return(0);
}
```

A2.3 newNav.c

```
/*-------------------------------------------*/
/*        newNav.c*/
/*       J bradford 2004                     */
/* Interpolator and Nav Process
  function development module for Towcam
 newNav.c
 2 Apr 04
 16 May 04 incorporated altitude delay algorithm
            and camera yaw algorithm          */
/*-------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#include <string.h>

int startTime,  timeLen, frameInt, x;
char x_char;
char *nav, *pmwname, *path;
double vidOffSec, ore_xoff, ore_yoff, ore_zoff, alt_delay;
float mosLen, frameTime;
int startFrame, numFrames, c, d, j, k, delay;
double north_old, east_old, time_old, fxDist, fyDist, fxOld, fyOld;
float xrotateYPR(double xi, double yi, double zi, double yaw, double pitch, double roll);
float yrotateYPR(double xi, double yi, double zi, double yaw, double pitch, double roll);
float zrotateYPR(double xi, double yi, double zi, double yaw, double pitch, double roll);
double LenLatMin(double lat), LenLonMin(double lat);
double Lin_interp (double x0, double x1, double x2, double y0, double y1) ;
int stdtime_days_in_month[12] = {31,28,31,30,    /* j f m a */
                                 31,30,31,31,    /* m j j a */
                                30,31,30,31};    /* s o n d */
int stdtime_dmy_to_jul_day (short day, short month, short year);
short jday;
void write_pmw();
typedef struct mos_telem{
                    int Time;
                    double North;
                    double East;
                    double Lat;
                    double Lon;
                    float Hdg;
                    float Smg;
                    float Cmg;
                    int Yr;
```

```
                int Mo;
                int Day;
                double Sec;
                float Pitch;
                float Roll;
                double Yaw;
                float Alt;
                float Depth;
                } mos_telem;


        mos_telem ship[64];
        mos_telem shipAvg;
        mos_telem orePos;
        mos_telem camPos;

typedef struct ore_dat{
        int Time;
        double Az;
        double Rge;
        float x;
        float y;
        float z;
        int err;
        double Sec;
        } ore_dat;

        ore_dat Ore[64];
        ore_dat OreTemp;
        ore_dat Interp_Ore;/*linearly interpolated data*/
double fish_xyRge, fish_z, oreAz;

        float i ;/*time increment variable*/
const char delimiter[] = ",";
const char workpath[]= "./TC_Working/";
char *token;
char * line = NULL;
 size_t len = 0;
 ssize_t read;
 char *tokenlist;
char * linelist = NULL;
const char listdelim[] =".";
 size_t lenlist = 0;
 ssize_t readlist;

char pmwnew[128];
```

```
FILE   *infile,  *infilelist, *pmwfile;

char *inname, *innamelist, *pmwout;


/*--------------Begin Main ---------------------*/
/*--------------Begin Main ---------------------*/
/*--------------Begin Main ---------------------*/
/*--------------Begin Main ---------------------*/

main() {

delay=2;/*delay to use later altitude (ie future alt)for towcam flat earth re-Projection*/
set_initial_values();/* loads start time,paths,gets data, sets fixed offsets, does some
averaging*/
j=0;k=0;
system("cd ./TC_Working\nls *.pgm > pgmlist.txt\ncd ..\n");
innamelist ="./TC_Working/pgmlist.txt";
infilelist = fopen(innamelist,"r");
jday = stdtime_dmy_to_jul_day (ship[j].Day, ship[j].Mo, ship[j].Yr);

for (i=0;i<mosLen/30;i=i+1/29.97){

frameTime = vidOffSec+i;
if(frameTime == vidOffSec) {ore_pos();camPos.Depth = orePos.Depth;}
if(frameTime >= ship[j+1].Sec) j++;
if(frameTime >= Ore[k+1].Sec) k++;
cam_pos();



write_pmw();

}/*eo for i<mosLen/30*/


fclose(infilelist);

printf("Normal Exit\n");
exit(0);

} /* eo Main */
/* eo Main *//* eo Main *//* eo Main **********************/
/*------------------ eo Main *//* eo Main *//* eo Main ----------------------*/
```

```
int set_initial_values(){
        inname ="./TC_Working/pgm.inf";
        infile = fopen(inname,"rb");
        if(!infile) {printf("%s not found ... I Quit!\n",inname); exit(0);}
        (read = getline(&line, &len, infile));



        vidOffSec = atof(line); /* sets the start time that will be passed to the function */

        (read = getline(&line, &len, infile));
        mosLen = atof(line);/* # of frames to be mosaiced */
        timeLen = (mosLen/30);
      printf("%.3f sec %.0f frames \n", vidOffSec,mosLen);
        fclose(infile);

        load_merged_Nav();
        load_Ore();
        load_offsets();
        ship_avg();

        return(0);
        }/* eo set_initial_values */

int load_merged_Nav(){
      readVidInf(); /* locates path to data*/
        inname = "";
        inname = nav;

        inname = strcat(inname,"mosTrack.txt"); /*    */

        infile = fopen (inname,"rb");

        if (!infile) {

                printf (" Could not open %s for reading, bummer.... man :(\n", inname);
                exit(-1);
        }/* eo if !infile*/



        while  (((read = getline(&line, &len, infile)) != -1) && (ship[0].Sec<=vidOffSec-
2)) {

                parse_Nav(0);
                }/* eo while read*/
```

```
                parse_Nav(1);
        {
                for(c=2;c<=timeLen+delay+1;c++){
                        (read = getline(&line, &len, infile));
                        parse_Nav(c);

                        }
                }
        for(c=0;c<=timeLen+delay+1;c++)
        printf("ship[%d]: %d,%.10f,%.10f,%.1f\n", c, ship[c].Time,ship[c].North,
ship[c].East, ship[c].Alt);
        fclose(infile);
        return(0);
        }/* eo load_merged_Nav*/

int parse_Nav(int i){


/*   Parse a  delimited text file */
                token = strtok (line,delimiter);/* time */
                ship[i].Time = atoi(token);
                token = strtok (NULL,delimiter);/*Decimal Degrees Lat Lon (North
East)*/
                ship[i].North = atof(token);
                token = strtok (NULL,delimiter);
                ship[i].East = atof(token);
                token = strtok (NULL,delimiter);/*Geo Lat Lon DDMM.mmmm*/
                ship[i].Lat = atof(token);
                token = strtok (NULL,delimiter);
                ship[i].Lon = atof(token);
                token = strtok (NULL,delimiter);/* Heading Speed Cmg */
                ship[i].Hdg = atof(token);
                token = strtok (NULL,delimiter);
                ship[i].Smg = atof(token);
                token = strtok (NULL,delimiter);
                ship[i].Cmg = atof(token);
                token = strtok (NULL,delimiter);/* Y M D */
                ship[i].Yr = atoi(token);
                token = strtok (NULL,delimiter);
                ship[i].Mo = atoi(token);
                token = strtok (NULL,delimiter);
                ship[i].Day = atoi(token);
                token = strtok (NULL,delimiter);/* Seconds */
                ship[i].Sec = atof(token);
                token = strtok (NULL,delimiter);/* Towfish Pitch */
                ship[i].Pitch = atof(token);
```

```
		token = strtok (NULL,delimiter);/* Towfish roll */
		ship[i].Roll = atof(token);
		token = strtok (NULL,delimiter);/* towfish alt */
		ship[i].Alt = atof(token);
		token = strtok (NULL,delimiter);/* towfish depth */
		ship[i].Depth = atof(token);

		return(0);

		}/*eo parse_Nav */

int load_Ore(){
	readVidInf();
	inname = "";
	inname = nav;
	printf("%s\n", nav);
	inname = strcat(inname,"ORE_Data.flt");
	infile = fopen (inname,"r");

	if (!infile) {
		printf (" Could not open %s for reading, bummer.... man :(\n", inname);
		exit(-1);
		}


	while  (((read = getline(&line, &len, infile)) != -1) &&  OreTemp.Sec <
vidOffSec-1){

				parse_Ore(0);
		}/*eo while read < vidOffSec*/

	parse_Ore(1);

	for (d=1;d<=floor(timeLen/2);d++){
		read=getline(&line, &len, infile);
		parse_Ore(d+1);
		}
		/*{printf("%s\n", line); /* spit out the input line*/


	fclose(infile);
	average_Ore_Az();
	return(0);
	}/*eo load_Ore */
```

```
int parse_Ore(int i){

                token = strtok (line,delimiter);/* time */
                Ore[i].Time = atoi(token);
                token = strtok (NULL,delimiter);
                Ore[i].Az = atof(token);
                token = strtok (NULL,delimiter);
                Ore[i].Rge = atof(token);
                token = strtok (NULL,delimiter);
                Ore[i].x = atof(token);
                token = strtok (NULL,delimiter);
                Ore[i].y = atof(token);
                token = strtok (NULL,delimiter);
                Ore[i].z = atof(token);
                token = strtok (NULL,delimiter);
                Ore[i].err = atoi(token);
                Ore[i].Sec = calc_sec(Ore[i].Time);
        OreTemp = Ore[i];
        return(0);
        }/*eo parse_Ore*/

int average_Ore_Az(){
                oreAz=0.0;
                for (j=0; j<d+1;j++){ oreAz=oreAz+Ore[j].Az;
        printf("Ore[%d]: %d,%.1f,%.2f,%.3f sec\n",j, Ore[j].Time,Ore[j].Az, Ore[j].Rge,
Ore[j].Sec);
                }
                oreAz=oreAz/(d+1);
printf("OreAz %d pts %.2f deg\n", d+1, oreAz);

        }/*eo average_Ore_Az*/


int load_offsets(){
                ore_xoff =-25.0; /* ore offset values for CCGS HUDSON*/
                ore_yoff =2.9;
                ore_zoff =6.0;


        return(0);
        }/*eo load_offsets()*/

int ship_avg(){

        shipAvg = ship[0];
```

```
        double yDist, xDist;
        yDist= (ship[timeLen].North-ship[0].North)*60*LenLatMin(ship[0].North);
        xDist= (ship[timeLen].East-ship[0].East)*60*LenLonMin(ship[0].North);
        shipAvg.Smg= pow(yDist,2) + pow(xDist,2);
        shipAvg.Smg= sqrt(shipAvg.Smg) / (ship[timeLen].Sec - ship[0].Sec) ; /* meters
per sec */
        shipAvg.Cmg= atan2(xDist,yDist)*180/M_PI;
        printf("shipAvg.Smg %.3f m/s shipAvg.Cmg %.3f deg\n",shipAvg.Smg,
shipAvg.Cmg);
        printf("%.3f my %.3f mx\n",yDist, xDist);
        return(0);
        } /* eo ship_avg*/


int ore_pos(){

 orePos = ship[j];

/* apply shipboard offsets antenna to transducer */

orePos.North = ship[j].North + xrotateYPR(ore_xoff, ore_yoff, ore_zoff, ship[j].Hdg, 0.0,
0.0)/(60*LenLatMin(ship[j].North));
orePos.East = ship[j].East + yrotateYPR(ore_xoff, ore_yoff, ore_zoff, ship[j].Hdg, 0.0,
0.0)/(60*LenLonMin(ship[j].North));
orePos.Depth = ship[j].Depth - zrotateYPR(ore_xoff, ore_yoff, ore_zoff, ship[j].Hdg, 0.0,
0.0);
printf("orePos.North %.10f orePos.East %.10f orePos.Depth %.3f\n", orePos.North,
orePos.East, orePos.Depth);
        }/* eo ore_pos */


int cam_pos() {
        double xDist, yDist;
Interp_Ore.Rge= Lin_interp(Ore[k].Sec,Ore[k+1].Sec, frameTime,
Ore[k].Rge,Ore[k+1].Rge);
fish_z = Lin_interp(ship[j].Sec,ship[j+1].Sec, frameTime,
ship[j].Depth,ship[j+1].Depth)-orePos.Depth;
fish_xyRge = sqrt((Interp_Ore.Rge*Interp_Ore.Rge) - (fish_z*fish_z));
/*printf("Interp_Ore.Rge %.3f %.3f\n", Interp_Ore.Rge, fish_z);*/
fxDist =(cos((oreAz+orePos.Hdg)*M_PI/180)*fish_xyRge) /
(60*LenLatMin(north_old));
fyDist =(sin((oreAz+orePos.Hdg)*M_PI/180)*fish_xyRge) /
(60*LenLonMin(north_old));


                        east_old = camPos.East;
```

```
                    north_old= camPos.North;
                    time_old = camPos.Sec;
                    camPos.Sec = frameTime;

/* generate interpolated ship to towcam bearing*/
orePos.Hdg=Lin_interp(ship[j].Sec,ship[j+1].Sec, frameTime, ship[j].Hdg,ship[j+1].Hdg);

/*interp ship lat */
camPos.North =Lin_interp(ship[0].Sec,ship[c-1].Sec, frameTime, ship[0].North,ship[c-
1].North);
/* offset to trackpoint transducer */
camPos.North = camPos.North + xrotateYPR(ore_xoff, ore_yoff, ore_zoff, ship[j].Hdg,
0.0, 0.0)/(60*LenLatMin(ship[j].North));
/* apply trackpoint avg bearing and depth adjusted range*/
camPos.North = camPos.North + ((cos((oreAz+orePos.Hdg)*M_PI/180)*fish_xyRge) /
(60*LenLatMin(north_old)));
/*apply tracpoint responder to camera Tow vehicle rotation*/
camPos.North = camPos.North + xrotateYPR(-0.20, -0.08, 0.49, ship[j].Hdg+ oreAz +
180 , camPos.Pitch, camPos.Roll)/(60*LenLatMin(ship[j].North));

/*interp ship lon */
camPos.East =Lin_interp(ship[0].Sec,ship[c-1].Sec, frameTime, ship[0].East,ship[c-
1].East);
/* offset to trackpoint transducer */
camPos.East = camPos.East + yrotateYPR(ore_xoff, ore_yoff, ore_zoff, ship[j].Hdg, 0.0,
0.0)/(60*LenLonMin(ship[j].North));
/* apply trackpoint avg bearing and depth adjusted range*/
camPos.East = camPos.East + ((sin((oreAz+orePos.Hdg)*M_PI/180)*fish_xyRge) /
(60*LenLonMin(north_old)));
/*apply tracpoint responder to camera Tow vehicle rotation*/
camPos.East = camPos.East + yrotateYPR(-0.2, -0.08, 0.49, ship[j].Hdg+ oreAz + 180,
camPos.Pitch, camPos.Roll)/(60*LenLonMin(ship[j].North));

camPos.Pitch =Lin_interp(ship[j].Sec,ship[j+1].Sec, frameTime,
ship[j].Pitch,ship[j+1].Pitch);
camPos.Roll =Lin_interp(ship[j].Sec,ship[j+1].Sec, frameTime,
ship[j].Roll,ship[j+1].Roll);

/* get the future altitude of the camera when over image footprint */
if(!camPos.Alt) camPos.Alt =2;
alt_delay = camPos.Alt * tan((camPos.Pitch +25)*M_PI/180) ;
alt_delay = alt_delay-22.25*0.0254;/* offset lense to altimeter*/
alt_delay = alt_delay/shipAvg.Smg; /*convert distance to time */
int delay1=floor(alt_delay);
int delay2=ceil(alt_delay);
if(ship[j+ delay2].Sec < frameTime + alt_delay){++delay1;++delay2;}
```

```
camPos.Alt =Lin_interp(ship[j+ delay1].Sec, ship[j+ delay2].Sec, frameTime + alt_delay,
ship[j+ delay1].Alt,ship[j+ delay2].Alt);
/*apply vertical offset from altimeter to lense*/
camPos.Alt = zrotateYPR(13.5*0.0254, 0.0, camPos.Alt-4.25*0.0254, 0.0, camPos.Pitch,
camPos.Roll);
camPos.Depth =Lin_interp(ship[j].Sec,ship[j+1].Sec, frameTime,
ship[j].Depth,ship[j+1].Depth);


yDist= (camPos.North- north_old)*(60*LenLatMin(north_old)) ; /*meters*/
xDist= (camPos.East - east_old)*(60*LenLonMin(north_old)) ;/*meters*/
camPos.Smg=sqrt(pow(yDist,2)+ pow(xDist,2));
camPos.Smg= camPos.Smg/(frameTime-time_old) ; /*m/s */

camPos.Cmg = atan2(xDist,yDist)*180/M_PI;


printf("cam North %.10f East %.10f Pitch %.3f Roll %.3f Alt %.3f Depth %.3f
\n",camPos.North,camPos.East, camPos.Pitch,camPos.Roll,camPos.Alt,camPos.Depth);
printf("SMG %.3f m/s CMG %.3f Deg %.3f sec\n", camPos.Smg,camPos.Cmg,
frameTime);
printf("shipAvg.Smg %.3f m/s shipAvg.Cmg %.3f deg %d %d\n",shipAvg.Smg,
shipAvg.Cmg, j,k);
printf("alt_delay %f, delay1 %d, delay2 %d, %f, %f \n",alt_delay ,delay1,delay2,ship[j+
delay1].Sec, ship[j+ delay2].Sec );
return(0);
        }/*eo cam_pos */

void write_pmw(){printf("got here! j %d, k %d\n", j, k);
        if(readlist = getline(&linelist, &lenlist, infilelist) !=-1){
        pmwname = strsep(&linelist,".");
        strcpy (pmwnew, workpath);
        strcat(pmwname,".pmw");
        strcat(pmwnew,pmwname);
        printf("Opening %s ..\n",pmwnew);
        pmwfile = fopen(pmwnew,"wb");
printf("%d,%d,%.3f,%.9f,%.9f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f\n", ship[j].Yr,jday,
frameTime,camPos.North,camPos.East, camPos.Pitch,camPos.Roll,camPos.Alt,
camPos.Depth, shipAvg.Smg, shipAvg.Cmg);

fclose(pmwfile);
}


return;
}/*eo write_pmw*/
```

```
int calc_sec (int hhmmss) {
        int Hr, Min;
        float Sec;
        Hr = floor(hhmmss/10000);
        Min = floor((hhmmss-Hr*10000)/100);
        Sec = hhmmss-Hr*10000-Min*100;
        Sec = Sec+Min*60+Hr*3600;

return(Sec);
}/*eo calc_sec*/

int readVidInf() {
        float BeginTime, EndTime;

        inname="Vid_Info.txt";
        line = NULL;
        infile = fopen(inname,"rb");

while  ((read = getline(&line, &len, infile)) != -1) {
        token = strtok (line,delimiter);
        token = strtok (NULL,delimiter);
        BeginTime = calc_sec(atof(token));

        token = strtok (NULL,delimiter);
        EndTime = calc_sec(atoi(token));
        token = strtok (NULL,delimiter);
        nav = token;

        token = strtok (NULL,delimiter);


        if(BeginTime < vidOffSec && EndTime > vidOffSec) {
/*printf("Vid Info was matched!! %f %f %f\n", BeginTime, vidOffSec, EndTime);*/
                fclose(infile);
                return(0);
                }
        }
printf("Vid Info Not matched %f %f %f\n", BeginTime, vidOffSec, EndTime);
exit(-1);
}/*eo load vid info*/


/*The rotateYPR function performs vector rotations in SIMRAD Space */
/* the x axis is positive towards the bow, the Y axis is */
/* positive to Starboard and the Z - axis is positive downwards*/
/* Yaw is positive clockwise, pitch is positive bow up */
```

```
/* roll is postive down to starboard */
/* originally coded by J Bradford for GGE6023 FA 2002*/
float xrotateYPR(double xi, double yi, double zi, double yaw, double pitch, double
roll){ /* Yaw Pitch Roll Matrices Times input vector */
inname = path;

        double xrot;
        double PI = M_PI;


xrot = xi * cos(PI/180*yaw) * cos(PI/180*pitch) - yi * sin(PI/180*yaw) * cos(PI/180*roll)
+ yi * cos(PI/180*yaw) * sin(PI/180*pitch) * sin(PI/180*roll) + zi * sin(PI/180*yaw) *
sin(PI/180*roll) + zi * cos(PI/180*yaw) * sin(PI/180*pitch) * cos(PI/180*roll);
        return(xrot);
        }


float yrotateYPR(double xi, double yi, double zi, double yaw, double pitch, double roll){

        double yrot;
        double PI = M_PI;

yrot = xi*sin(PI/180*yaw)*cos(PI/180*pitch) + yi*cos(PI/180*yaw)*cos(PI/180*roll) +
yi*sin(PI/180*yaw)*sin(PI/180*pitch)*sin(PI/180*roll) -
zi*cos(PI/180*yaw)*sin(PI/180*roll) +
zi*sin(PI/180*yaw)*sin(PI/180*pitch)*cos(PI/180*roll);
        return(yrot);
        }

float zrotateYPR(double xi, double yi, double zi, double yaw, double pitch, double roll){

        double zrot;
        double PI = M_PI;
        zrot= (-xi*sin(PI/180*pitch) + yi*cos(PI/180*pitch)*sin(PI/180*roll) +
zi*cos(PI/180*pitch)*cos(PI/180*roll));
         return(zrot);

        }
/* eo rotateYPR functions */


/*LenLatMin and LenLonMinfunction based on Admiralty Manual of Navigation Vol1
page 44/45 */
/*WGS params from NIMA TR8350.2 dated 4 July 1997 */
/*created by J Bradford Feb 2004 */
```

```
double LenLatMin(double lat) { /* Length in metres of a minute of Lat */

double a, f, e, Len, Q, x;

        x = lat*M_PI/180; /* convert to Radians */
        a =  6378137.00; /* WGS 84 Semi Major Axis */
        f = 1/298.257223563; /* WGS84 Flattening */
        e = sqrt((2*f-f*f)); /* eccentricity */

        Q=(1-e*e*sin(x)*sin(x));
        Len = a*(1-e*e)/sqrt(Q*Q*Q)*sin(M_PI/10800);/* length in metres */

        return(Len);

        }

double LenLonMin(double lat) { /* Length in metres of a minute of Long at a given Lat*/

double a, f, e, Len, Q, x;

        x = lat*M_PI/180; /* convert to Radians */
        a =  6378137.00; /*WGS84 Semi Major Axis */
        f = 1/298.257223563; /* WGS84 Flattening */
        e = sqrt((2*f-f*f)); /* eccentricity */

        Q=(1-e*e*sin(x)*sin(x));
        Len = a*cos(x)/sqrt(Q)*sin(M_PI/10800);/*length in metres*/

        return(Len);

        }

double Lin_interp (double x0, double x1, double x2, double y0, double y1) {
        /*y=mx+b interpolator j bradford 2004*/
        double y2, b, m;
        m = (y1-y0)/(x1-x0);
        b = y0-m*x0;
        y2 = m*x2+b;
/*printf("y2 %f, m %f, x2 %f b %f x0 %f x1 %f y0 %f y1 %f\n", y2, m, x2, b,
x0,x1,y0,y1);*/
return(y2);
} /*eo Lin_interp */

int IS_LEAP_YEAR(int year)
{
      if(year%4 != 0) return(0);
```

```
        else
        if(year%400 == 0) return(1);
        else
        if(year%100 == 0) return(0);
        else
        return(1);
}
int stdtime_dmy_to_jul_day (short day, short month, short year)
{
        short jday;
           jday = day;
        if (IS_LEAP_YEAR(year) && month>2) jday++;
        while (--month) jday+=stdtime_days_in_month[month-1];
        return (jday);
}
```

A2.4 projNew.c

```
/* ---------------------------------------
 * projects extracted video frame to
   a plane, applies radial and radiometric
          correction
 * projNew.c   Bradford 2004
 * 09 Mar 04 - encoded cropping of image to output only rows and
                columns containing data
 * 26 Mar 04 - added nocrop switch for use in making animated gifs
 * 14 May 04 - reduced size of proj frame to use only 30 rows
          (sweet spot)
* 16 May 04 - added switch to control number of rows in sweet spot
 *--------------------------------------------- */

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#include <string.h>


#define BYTE unsigned char

typedef struct telem{
                double Time;
                double North;/*decimal Degrees*/
                double East;/*decimal degrees*/
                double Lat;
                double Lon;
                float Hdg;
                float Smg;
                float Cmg;
                int Yr;
                int Mo;
                int Day;
                double Sec;
                float Pitch;
                float Roll;
                float Alt;
                float Depth;
                float OreAz;
                float OreRge;
                } telem;
telem Cam;

int COL, ROW, outRow, outCol, height, width, topClip=219, botClip=269;
```

```
float pixAng, dFOV, rayElev, rayRoll, camPitch, camRoll, camYaw, mountPitch, lensAlt,
rayRange;
float uDist, vDist, pixRes, k1, k2, f;
float imRotX(double xi,double yi, double zi, double yaw, double pitch, double roll);
float imRotY(double xi,double yi, double zi, double yaw, double pitch, double roll);
long i, j;
char *line;
const char delimiter[] =",";/* for use in strtok */
void fill_point( int x, int y, int color, unsigned char weight);

        BYTE *outim, *out2im;;


FILE    *infile, *outfile, *out2file, *pmwfile, *erroutfile;
int wts_flag=0;
int man_flag=0;
int no_crop=0;
int option_select;

main (int argc, char ** argv) {

        char     *inname=0l,  *outname=0l, *out2name=0l;

        BYTE *inim;
        int *outXcoord, *outYcoord;
        BYTE *weights;

        while (*(++argv)) {
if (!strcmp (*argv,"-in")) {

                if (!*(++argv)) {

                        printf ("-in requires input binary filename ");
                        err_exit();
                }
                inname = *argv;

        } else if (!strcmp (*argv,"-out")) {
                if (!*(++argv)) {

                        printf ("-out requires output file name ");
                        err_exit();
                }

                outname = *argv ;
```

```
        } else if (!strcmp (*argv,"-wts")) {
                if (!*(++argv)) {

                        printf ("-wts requires output file name ");
                        err_exit();
                }

                wts_flag++;
                out2name = *argv ;
        } else if (!strcmp (*argv,"-manual")) {


                man_flag++;
                 ;
        } else if (!strcmp (*argv,"-no_crop")) {


                no_crop++;
                 ;
        } else if (!strcmp (*argv,"-top")) {

                ++argv;
                topClip= atoi(*argv);
                 ;
        }  else if (!strcmp (*argv,"-bot")) {

                ++argv;
                botClip= atoi(*argv);

                 ;
        } else {
                printf(" garbage on command line %s\n", *argv);
                exit(0);

        }
  }

        if (!inname || !outname) err_exit();
        else
        printf("\n %s   --> -->  %s\n  ", inname, outname );
/*-------------------------------------------------------------*/

        infile = fopen (inname,"rb");
        if (!infile) {
                printf (" Could not open %s for reading, bummer \n", inname);
```

```
                err_exit();
        }

        outfile = fopen (outname,"wb");
        if (!outfile) {
                printf (" Could not open %s for writing, bummer \n", outname);
                err_exit();
        }
        if(wts_flag) {
        out2file = fopen (out2name,"wb");
        if (!out2file) {
                printf (" Could not open %s for writing, bummer \n", out2name);
                err_exit();
        }
        }

        erroutfile = fopen ("error_dump","wb");
        if (!erroutfile) {
                printf (" Could not open error_dump for writing, bummer \n");
                err_exit();
        }

/*-----------------------------------------------------------------*/


GetPGMrc();
read_pmw(inname);

if (topClip>ROW/2-1||topClip<0) topClip=0;/*test for valid image row clip values ->*/
if (botClip<ROW/2||botClip>480) botClip=480;/*-> if not valid set to max*/
printf("topClip %d, botClip %d\n", topClip, botClip);
dFOV = 96.24*M_PI/180;
pixAng=dFOV/sqrt(ROW*ROW+COL*COL);
k1=-2.8e-6;/* radial distortion constant estimated value from rough calibration*/
k2=0.0;/* estimated value from rough calibration*/
set_params();

if(man_flag>0) { getManualAtt();}

inim = (BYTE *) malloc(COL * ROW * sizeof(BYTE));
weights = (BYTE *) malloc(COL * ROW * sizeof(BYTE));
memset(weights, COL*ROW,255);
outXcoord = (int *) malloc(COL * ROW * sizeof(int));
outYcoord = (int *) malloc(COL * ROW * sizeof(int));
```

```
unsigned char imageByte;
long byteCount = 0;
height = 1500+(botClip-topClip)*5;
width =2000+(botClip-topClip)*5;
printf("pixAng= %f\t pixRes=%f\t ht Val = %d\n", pixAng,pixRes, height);
double value;
outim = (BYTE *) malloc(height * width * sizeof(BYTE));
memset(outim, 255, height*width);

if(wts_flag) {
out2im = (BYTE *) malloc(height * width * sizeof(BYTE));
memset(out2im, 255, height*width);
}



fseek(infile,-(ROW * COL),SEEK_END);

/* read in the whole input image */
        fread (inim,ROW*COL,1,infile);
for (j=topClip;j<botClip; j++) { for (i=0; i < COL; i++) {
        if(*(inim+(j*COL)+i) ==255) *(inim+(j*COL)+i) = 254; /* restrict values
output*/
}}
/* ***BEGIN radial,project,radiometric corrections */
for (j=topClip;j<botClip; j++) { for (i=0; i < COL; i++) {


        rayElev = (((ROW/2-j)*pixAng)+camPitch+mountPitch)*180/M_PI;
        rayRoll = ((((COL/-2)+i)*pixAng)+camRoll)*180/M_PI;

        float r, dr;/*radial distortion variables*/



        if(rayElev < 80 && rayRoll < 80 && rayRoll > -80 && rayElev > -80) {

        uDist=((ROW/2)+6-j);/*de-centering pixel shift +6 rows -2 cols
                                from cam calib*/
        vDist=((COL/-2)-2+i);
        r = sqrt(uDist*uDist + vDist*vDist);/*Correcting for Radial Distortion*/
        dr = k1*r*r + k2*r*r*r*r;
```

```
uDist= uDist*(1-dr)*pixRes*1.07/2;/* undistort and scale */
vDist = vDist*(1-dr)*pixRes/2;/* watch for problems with scaling values*/

double x;
double y;
        /*** rotation and reproject after Derenyi 1996 eqn.(4-7)*/
x = -lensAlt*imRotX(uDist, vDist, -f, 0.0, - (camPitch + mountPitch), camRoll);
y = -lensAlt*imRotY(uDist, vDist, -f, 0.0, - (camPitch + mountPitch), camRoll);


outRow = floor(x/pixRes); /*output result by row and col */
outCol =floor(y/pixRes);

                *(outXcoord+j*COL+i) = outCol;
                  *(outYcoord+j*COL+i) = outRow;


                  if(wts_flag) {

                        if (j < ROW/2) /*weight by row from middle to top*/
                  *(weights+j*COL+i) = 255* ((ROW/2)-j)/(ROW/2);
                        else /*weight by row from middle to bottom*/
                  *(weights+j*COL+i) = 255* (j-(ROW/2))/(ROW/2)*.5;


                  }/* eo if wts_flg*/

value =*(inim+j*COL+i);
value = 0.9*value + r/30;
r = sqrt(lensAlt*lensAlt/pixRes+r*r);
r=r/90;
value = value + pow(r,3) ;
*(inim+j*COL+i) = value;
if (value > 255) value = 255;
if (value < 0) value = 0;

*(outim + (height-1000-outRow)*width+outCol+width/2) = *(inim+j*COL+i);

} else {
                *(outXcoord+j*COL+i) = -999.0;
                  *(outYcoord+j*COL+i) = -999.0;
                  if(wts_flag) *(weights+j*COL+i) = 0;
}

++byteCount;
}
```

```
} /* eo radial,project,radiometri */

/* now go back through and paint in between pixel quads */

for (j=topClip;j<botClip-1; j++) { for (i=0; i < COL-1; i++) {

        fill_quad2 (
          *(outXcoord+j*COL+i), *(outYcoord+j*COL+i),
                                (int)(*(inim+j*COL+i)),
          *(outXcoord+(j+1)*COL+i), *(outYcoord+(j+1)*COL+i),
                                (int)(*(inim+(j+1)*COL+i)),
          *(outXcoord+(j+1)*COL+(i+1)), *(outYcoord+(j+1)*COL+(i+1)),
                                (int)(*(inim+(j+1)*COL+(i+1))),
          *(outXcoord+j*COL+(i+1)), *(outYcoord+j*COL+(i+1)),
                                (int)(*(inim+j*COL+(i+1))),
                                (int)(*(weights+j*COL+i))    );


        }
}



int maxCol, minCol, topRow, botRow;
unsigned char pixval;
/* locate row and col limits of image for use in cropping the output */
        for(j=0;j<height;j++){for(i=0;i<width;i++) { pixval=  *(outim+j*width+i);
                                if (pixval!=255){ printf("%d \t", pixval);
                                topRow=j; j=height; i=width;
                                        }
                                }
                        }

        for(j=height-1;j>topRow;j--){for(i=0;i<width;i++) { pixval=*(outim+j*width+i);
                                if (pixval!=255){ printf("%d \t", pixval);
                                botRow=j; j=topRow; i=width;
                                        }
                                }
                        }
        for(i=0;i<width;i++){for(j=0;j<height;j++){ pixval= *(outim+j*width+i);
                                if (pixval!=255){ printf("%d \t", pixval);
                                minCol=i;j=height; i=width;
                                        }
                                }
                        }
```

```
        for(i=width-1;i>minCol;i--){for(j=0;j<height;j++) { pixval= *(outim+j*width+i);
                                if (pixval!=255){ printf("%d \t", pixval);
                                maxCol=i; j=height; i=minCol;
                                        }
                                }
                        }


printf("Top %d Left %d Bot %d Right %d\n", topRow, minCol, botRow, maxCol);

if(no_crop){topRow=0;minCol=0;botRow=height;maxCol=width;} /* reset the crop
bounds to max extent */


        fprintf(outfile,"P5\n#PGM reprojection utility\n#%f %f
%f\n#$PMW,%d,%d,%.3f,%.9f,%.9f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f\n%d %d 255\n",
pixRes, (minCol+width/-2)*pixRes, (height-topRow-1000)*pixRes,  Cam.Yr, Cam.Day,
Cam.Time, Cam.North, Cam.East, Cam.Pitch, Cam.Roll, Cam.Alt, Cam.Depth,Cam.Smg,
Cam.Cmg,(maxCol-minCol), (botRow-topRow));
        for (j=topRow;j<botRow;j++){
        fwrite (outim+j*width+minCol,(maxCol-minCol),1,outfile);} /* write to the pgm
file */
        export_track(Cam.Yr,Cam.Day,Cam.Time,Cam.North,Cam.East);
        if(wts_flag) {
        fprintf(out2file,"P5\n#PGM reprojection utility\n#%f %f
%f\n#$PMW,%d,%d,%.3f,%.9f,%.9f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f\n%d %d 255\n",
pixRes,(minCol+width/-2)*pixRes, (height-topRow-1000)*pixRes,  Cam.Yr, Cam.Day,
Cam.Time, Cam.North, Cam.East, Cam.Pitch, Cam.Roll, Cam.Alt, Cam.Depth,Cam.Smg,
Cam.Cmg,(maxCol-minCol), (botRow-topRow));
        for (j=topRow;j<botRow;j++){
        fwrite (out2im+j*width+minCol,(maxCol-minCol),1,out2file);}/* write to the
pgm file */
        }

        /* Tidy up and Exit */


free(inim);
free(outXcoord);
free(outYcoord);
free(outim);
free(weights);
if(wts_flag) free(out2im);
fclose(infile);
fclose(outfile);
if(wts_flag) fclose(out2file);
```

```
printf ("\t%d\tImage Bytes Projected\n\n Done!!\n", byteCount);

exit(0);


 }

/* ------------- eo main program -------------*/

int GetPGMrc () {

        int  OAcount = 0;
        BYTE inbyte;
        BYTE r[6];
        BYTE c[6];
         i=0;
         j=0;
        ROW=0;
        COL=0;

        while(!feof(infile)) {

        fread (&inbyte,1,1,infile);

                if (inbyte==0x0A) {OAcount ++;}

                if (OAcount == 2 && COL == 0 ) {  c[i] = inbyte; i++;

                                                }
                if (OAcount == 2 && inbyte == 0x20)         COL= atoi(c);

                if (COL > 0 && inbyte > 0x29 )        {r[j] = inbyte; j++;}

                if (COL > 0 && inbyte == 0x20 ) ROW = atoi(r);

                if (OAcount == 3)       { break;}

        }

        printf("Rows= %d \tCols= %d \n", ROW, COL);

        return(0);
} /* eo GetPGMrc */

int ExportFrames () {
```

```
        return(0);
} /* eo ExportFrames */

int err_exit()
{
      fprintf(stderr, "Oooopps I need switches Usage: %s\n", "Description");
         fprintf(stderr, "-in input binary filename \n");
      fprintf(stderr, "-out outfilename \n");
      exit (-1);
}
/* ---------------------------------------- */
/* ---------------------------------------- */
void fill_point(int x, int y, int color, unsigned char weight) {
int theX, theY;
                /* to avoid overflowing map sheet */
        theX = x+width/2;
        theY = height-1000-y;

        if(theX>0 && theX < width &&
          theY>=0 && theY < height) {
         *(outim + theY*width+theX)=color;

        if(wts_flag)
                *(out2im + theY*width+theX)=weight;
        }
}

/* ---------------------------------- */

/*   Open, Parse and Close a  comma delimited text file */
int read_pmw(char *inname) {

        char  *token, *innamepmw;
        line = NULL;
        size_t len = 0;
        ssize_t read;

        innamepmw = strsep(&inname,"p");
        strcat (innamepmw,"pmw");
        printf("%s\n", innamepmw);

        pmwfile = fopen(innamepmw, "rb");
if (!pmwfile) {
                printf (" Could not open %s for reading, selecting manual input
\n",innamepmw);
                man_flag++;
```

```
                return(0);
                }

        (read = getline(&line, &len, pmwfile));
        token = strtok(line,delimiter);/*Year*/
        Cam.Yr = atoi(token);
        token = strtok (NULL,delimiter);/*Year*/
        Cam.Day = atoi(token);
        token = strtok (NULL,delimiter);/* time sec*/
        Cam.Time = atof(token);
        token = strtok (NULL,delimiter); /* northing */
        Cam.North = atof(token);
        token = strtok (NULL,delimiter); /*easting*/
        Cam.East = atof(token);
        token = strtok (NULL,delimiter); /*pitch*/
        Cam.Pitch= atof(token);
        token = strtok (NULL,delimiter); /*roll*/
        Cam.Roll = atof(token);
        token = strtok (NULL,delimiter); /*Alt*/
        Cam.Alt = atof(token);
        token = strtok (NULL,delimiter); /*depth*/
        Cam.Depth = atof(token);
        token = strtok (NULL,delimiter); /* ore az */
        Cam.Smg = atof(token);
        token = strtok (NULL,delimiter); /* ore range*/
        Cam.Cmg = atof(token);
printf("$PMW,%d,%d,%.3f,%.9f,%.9f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f
\n",Cam.Yr,Cam.Day,Cam.Time, Cam.North, Cam.East,Cam.Pitch,Cam.Roll,Cam.Alt,
Cam.Depth, Cam.Smg, Cam.Cmg);

        fclose(pmwfile);


return(0);
}
int export_track(int yr, int jday, double time, double north, double east){
FILE *expfile; /* creates a txt file for import to ArcView*/
expfile = fopen("MOS_Working/sitesout.txt","ab");
fprintf(expfile,"%.9f,%.9f,%d,%d,%.3f\n",north,east,yr,jday,time);
fclose(expfile);
return(0);
}
int getManualAtt(){
printf("Select Toggle Num to Adjust Navigation &/or Attitude Parameters\n(1)North=
%.9f\t(2)East=%.9f\t(3)Alt=%.3f\t\n(4)Pitch=%.2f\t\t(5)Roll=%.2f\t(6)Yaw=%.2f\t(7)En
ter Rows and Cols \n(8) to adjust camera Diagonal FOV (%.2f)\n(0) When Done to
```

```
Project Frame!\nPix Res is Set To: %.3f mm\tRows= %d Cols=
%d\n\n",Cam.North,Cam.East,Cam.Alt,Cam.Pitch+mountPitch*180/M_PI,Cam.Roll,Ca
m.Cmg,dFOV*180/M_PI,pixRes*1000, ROW, COL);



        printf("Make Selection (0-7):  \n");
                scanf("%d \n", &option_select);

        switch(option_select){
        case 0 : set_params();return(0);
        break;
        case 1 : printf("Enter New Northing Value:  \n");
         scanf("%f", &Cam.North);
         set_params();
        break;
        case 2 : printf("Enter New Easting Value:  \n");
         scanf("%f", &Cam.East);
         set_params();
        break;
        case 3 : printf("Enter New Altitude Value:  \n");
        scanf("%f", &Cam.Alt);
        set_params();
        break;
        case 4 : printf("Enter New Pitch Value:  \n");
        scanf("%f", &Cam.Pitch);
        set_params();
        break;
        case 5 : printf("Enter New Roll Value:  \n");
         scanf("%f", &Cam.Roll);
         set_params();
        break;
        case 6 : printf("Enter New Yaw Value:  \n");
        scanf("%f", &Cam.Cmg);
        set_params();
        break;
        case 7 : printf("Enter New Row Value: \n");
                  scanf("%d", &ROW);
                 printf("Enter New Col Value: \n");
                 scanf("%d", &COL);
                 set_params();
        break;
        case 8 :printf("Enter Diagonal FOV Value: \n");
                  scanf("%f", &dFOV);
                 dFOV = dFOV*M_PI/180;
                 set_params();
```

```
        break;
        }/*eo switch*/

getManualAtt();
}

int set_params() {

pixAng=dFOV/sqrt(ROW*ROW+COL*COL);
mountPitch = 25*M_PI/180; /*measured mount angle for camera on towbody*/
camPitch = Cam.Pitch*M_PI/180;
camRoll = Cam.Roll*M_PI/180;
camYaw = Cam.Cmg*M_PI/180;
lensAlt = (Cam.Alt)*cos(camPitch);
pixRes=0.005;/*arbitrary resolution value*/
/*pixRes= 2*lensAlt*tan(pixAng/2);/* ifov x (secant squared view angle)*/
        f = (sqrt(ROW/2*ROW/2+COL/2*COL/2))/tan(dFOV/2)*pixRes; /* calculate
focal length in pixel and convert to length*/
        printf("f = %f\n", f);
return(0);
}


float imRotX(double xi,double yi, double zi, double yaw, double pitch, double roll) {
        /* after Derenyi 1996 eqn (2-20) and (4-7)*/
        double xrot;

        xrot= xi * cos(yaw) * cos(pitch) - yi*sin(yaw)*cos(pitch) + zi*sin(pitch) ;

        xrot = xrot / (xi* (sin(yaw) * sin(roll) - cos(yaw) * sin(pitch) * cos(roll)) +
yi*( cos(yaw)*sin(roll) + sin(yaw)*sin(pitch)*cos(roll)) +  zi*cos(pitch)*cos(roll));

        return(xrot);
        }

float imRotY(double xi,double yi, double zi, double yaw, double pitch, double roll) {
        /* after Derenyi 1996 eqn (2-20) and (4-7)*/
        double yrot;

        yrot = xi * (sin(yaw) * cos(roll) + cos(yaw) * sin(pitch) * sin(roll)) +
yi*(cos(yaw)*cos(roll) - sin(yaw)*sin(pitch)*sin(roll)) - zi*cos(pitch)*sin(roll);
        yrot = yrot/(xi* (sin(yaw) * sin(roll) - cos(yaw) * sin(pitch) * cos(roll)) +
yi*( cos(yaw)*sin(roll) + sin(yaw)*sin(pitch)*cos(roll)) +  zi*cos(pitch)*cos(roll));
        return(yrot);
        }
```

A2.5 Merge_Data

```
# shell script to
#strip and reformat pertinent nav and telemetry data from ".03e and .03T" files
#script is called by mosProc.c
#j bradford 29 Mar 04
#an OFS with a trailing space ", " is used because later
#processing employs the function strtok
#which does not behave well with NULL fields
rm -f *.txt *.tmp *.flt
echo .txt .tmp and .flt files removed ... writing NMEA_Data.tmp
gawk -F, 'BEGIN {OFS=", "} /GPZDA/{gpYr=$5; gpDay=$3; gpMon=$4} /GPVTG/
{gpCmg=$2; gpSmgKm=$8}  /HDT/ {if ($2!="")shipHdg=$2}  /GPGGA/ {gpTime=$2;
gpLat=$3; gpLon=$5; hr=int($2/10000);min=int(($2-int($2/10000)*10000)/100);
sec=$2-int($2/100)*100;  print
gpTime,gpLat,gpLon,shipHdg,gpSmgKm,gpCmg,gpYr,gpMon,gpDay,hr*3600+min*60
+sec}' NMEA_Data.03e > NMEA_Data.tmp
tail -n 5 NMEA_Data.tmp
echo writing ORE_Data.tmp
gawk -F, '/GPGGA/ {OFS=", "; gpTime=$2} /POREB/ {oreAz=$5; oreRge=$6;
oreDx=$7; oreDy=$8; oreDz=$9; oreErr=$11; oreRoll=$12; orePit=$13; print
gpTime,oreAz,oreRge,oreDx,oreDy,oreDz,oreErr,oreRoll,orePit}' NMEA_Data.03e >
ORE_Data.tmp
tail -n 5 ORE_Data.tmp
echo extract Towcam data and comma delimit it placing gps time in first field
gawk '{FS=", "; OFS=", ";if (NR>2 && timeOld <$7) print $7,$3,$4,$5,$6;timeOld=$7}'
Towcam_Data.03T > Towcam_Data.tmp
echo combine ship and towcam 1Hz data
join -t , NMEA_Data.tmp Towcam_Data.tmp > Merge_Data.tmp
tail -n 5 Merge_Data.tmp
echo convert ddmm.mm Lat and Lon to dd.ddddd and join to 1Hz data
gawk -F, '{printf("%.6d, %.10f, %.10f\n"), $1,  (int($2/100)+($2-
(int($2/100)*100))/60),(int($3/100)+($3-(int($3/100)*100))/60)*-1 }' Merge_Data.tmp >
lola.tmp
join -t , lola.tmp Merge_Data.tmp > mosTrack.txt
echo run median filter on Trackpoint Az
#filtNav is a five point median filter run on Trackpoint Az
../filtNav
tail -n 5 *.flt
```

A2.6 to_jhc

```
#script to generate mosaic images from re-projected pgm files
#j bradford Mar 2004
 PCItoJHC -pgm -in $1.pgm -out $1.rot -bradford -towcam_azi $2
```

```
cp closeup.blank $1.mos
patchArea -interp -prime -back 255 -out $1.mos -ignore 255 -import $1.rot.band1
```

A2.7 Corners.sh:

```
#corners.sh
#shell script to identify the max and min Lat Lon of selected reprojected pgm files
#it calculates and prints the Upper Left and Lower Right corners
#of a bounding box 5 meters outside the max and mins
# these coords may be cut and pasted into the OMG "make_blank" utility
#j bradford Mar 2004

rm -f *000000*
# pass first 4 lines of each pgm file through gawk to identify max/min lats then add/sub
# 5 meters writing out the result to screen in decimal degrees
head -n4 0*.pgm | gawk 'BEGIN {FS=","; PI=3.14159265359;  OFS=",";maxN=-
90;minN=90;minE=180;maxE = -180} /PMW/ { if (maxN < $5) maxN=$5; if (minN >
$5) minN=$5; if (minE > $6) minE=$6; if (maxE < $6) maxE=$6; course=$12} END
{printf("UL Corner= %.10f %.10f\nLR Corner= %.10f %.10f\n", maxN+5.0/1852/60,
minE-5.0/(1852*cos(maxN*PI/180))/60, minN-5.0/1852/60,
maxE+5.0/(1852*cos(maxN*PI/180))/60)}'

rm -f pmw.txt

# pass first 4 lines of each pgm file through gawk and
#output Frame Number with PMW data
# in comma delimited format for use as a camera track file for input to ARCGIS
head -n4 0*.pgm | gawk 'BEGIN {FS=","; PI=3.14159265359;  OFS=",";maxN=-
90;minN=90;minE=180;maxE =-180;
print("Frame,Yr,Day,Sec,Lat,Lon,Pitch,Roll,Alt,Depth,SMG,CMG");i=1} /PMW/ { if
(maxN < $5) maxN=$5; if (minN > $5) minN=$5; if (minE > $6) minE=$6; if (maxE <
$6) maxE=$6; course=$12;
printf("%.6d,%d,%d,%.3f,%.10f,%.10f,%.3f,%.3f,%.3f,%.3f,%.3f\n",i,$2, $3, $4,
$5, $6, $7, $8, $9, $10, $11, $12) ;i++}' > pmw.txt
echo pmw.txt written

# call make_blank tool for generation of mosaic header
make_blank closeup
```

Appendix 3 Sample Telemetry, Navigation and Mosaic Data

A3.1 Long-term Telemetry from TOWCAM

Altitude and depth in meters, along with pitch and roll data are recorded from the TOWCAM vehicle at 1 Hertz. Figure A3-1 displays this data for an extended period taken along one survey line. The vehicle pitch varies widely between -15 and 28 degrees. There appears to be a long term signal of about 90 seconds evident in the pitch data and to a lesser extent in the roll and altitude data. The origin of this signal is unknown but may represent a response from the winch feedback control system. There is a shorter term signal also evident which is described in paragraph A3.2. The pitch data has a positive (nose up) bias while the roll data has a negative (down to starboard) bias.

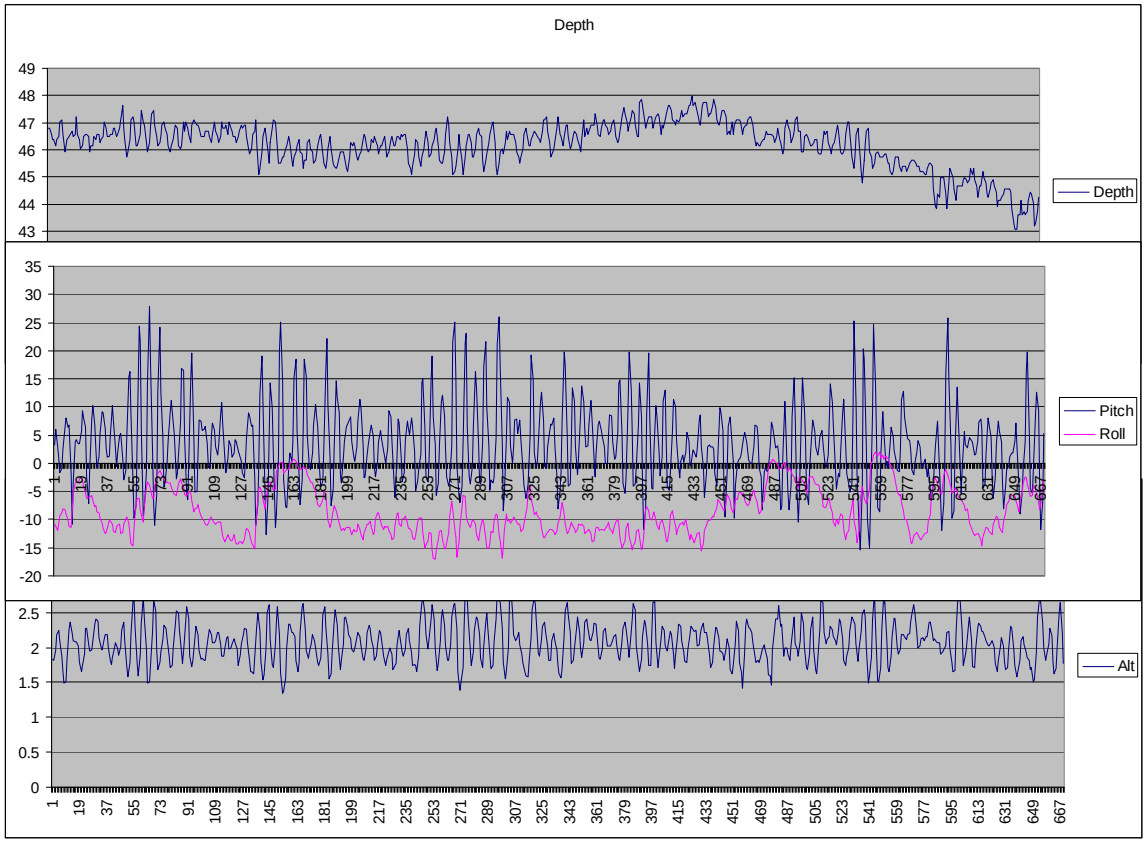

Figure A3-1 Long-term Telemetry Data from TOWCAM 003701-004959, pitch and roll in degrees, altitude and depth in meters.

A3.2 Video Mosaic Composite

This section presents a composite of four mosaic segments covering the epoch from 003930-004058. Graphical presentation of the navigational data from GPS and the Acoustic Positioning system for the timeframe are shown in figure A3-2. Each mosaic in the composite is also presented in the following sub-sections along with the attitude and altitude data from the corresponding period. A 6-8 second signal is evident in the data and is believed to correspond to vessel motion generated by ocean swell. Scattered boulders are clearly evident across the image with a heavy concentrations of boulders towards the eastern end of the mosaic. The shape of current / wave generated bedforms in the surface sediment can also be seen. By comparing the mosaic image with the graphical data the cumulative effect of camera altitude and attitude can be seen. The dark banding along the upper quarter of the mosaic is related to light attenuation in the water column due to range, as the vehicle is rolled down to starboard the video swath covers a further distance to port.

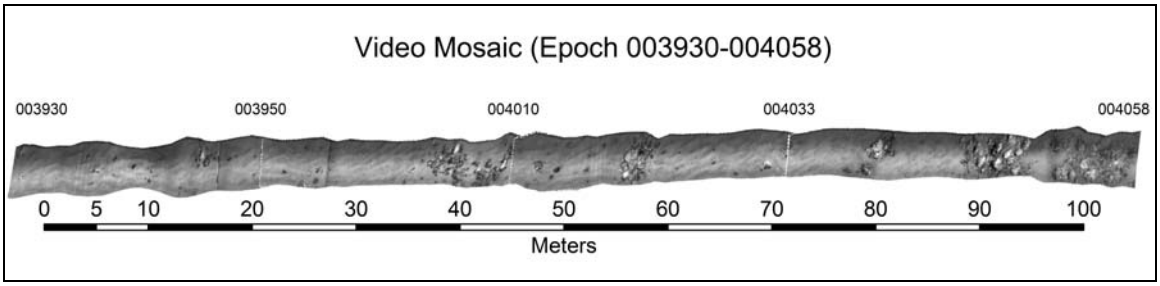

Figure A3-2 Composite Mosaic from TOWCAM Video. The dark and light banding along the image are due to un-modeled range related radiometric attenuation.

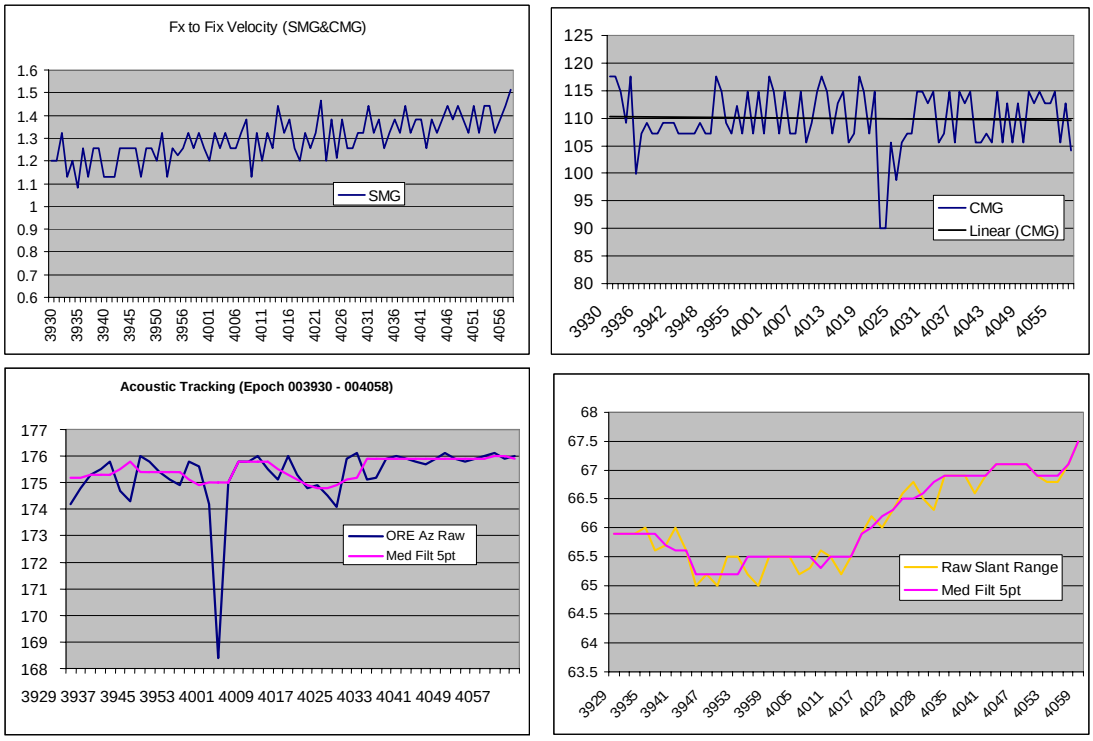

Figure A3-3 Navigation and Acoustic Positioning Data for composite mosaic time period. The top panels show Speed Made Good (SMG) in meters per second and Course Made Good in degrees true calculated between sequential GPS fixes. The bottom panels show Acoustic Tracking Data with a median filter superimposed, azimuth is in degrees while range is in meters.

A3.2.1 Video Mosaic from Epoch 003930-003955

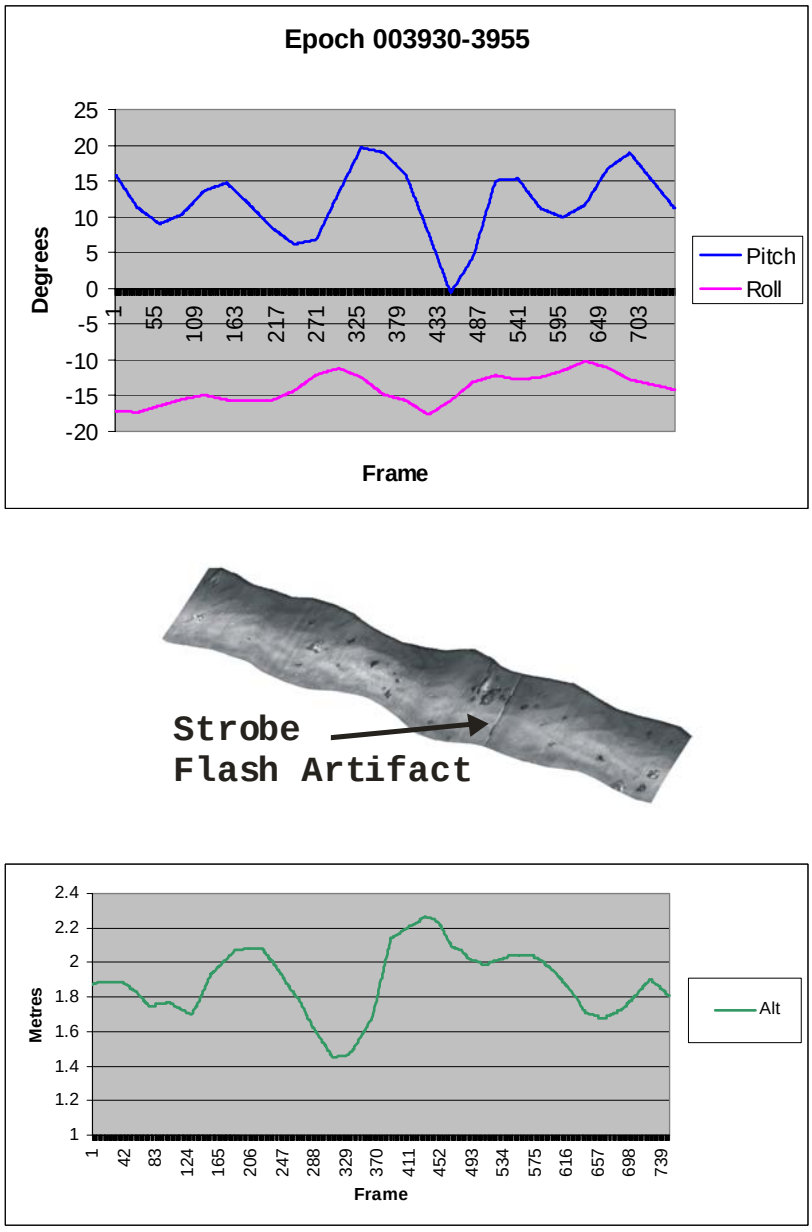

Figure A3-4 Mosaic from Epoch 3930-3955 with Attitude and Altitude Data

A3.2.2 Video Mosaic from Epoch 003950-004015

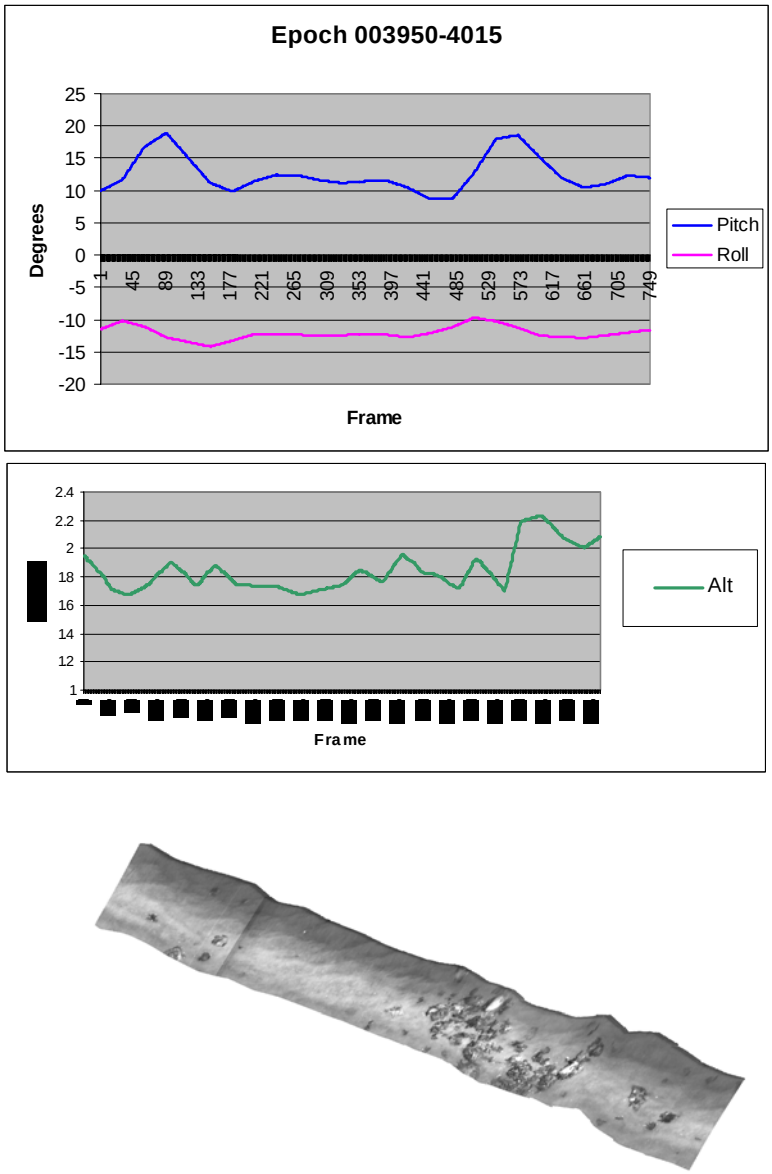

Figure A3-5 Mosaic from Epoch 3950-4015 with Attitude and Altitude Data

A3.2.3 Video Mosaic from Epoch 004010-004035

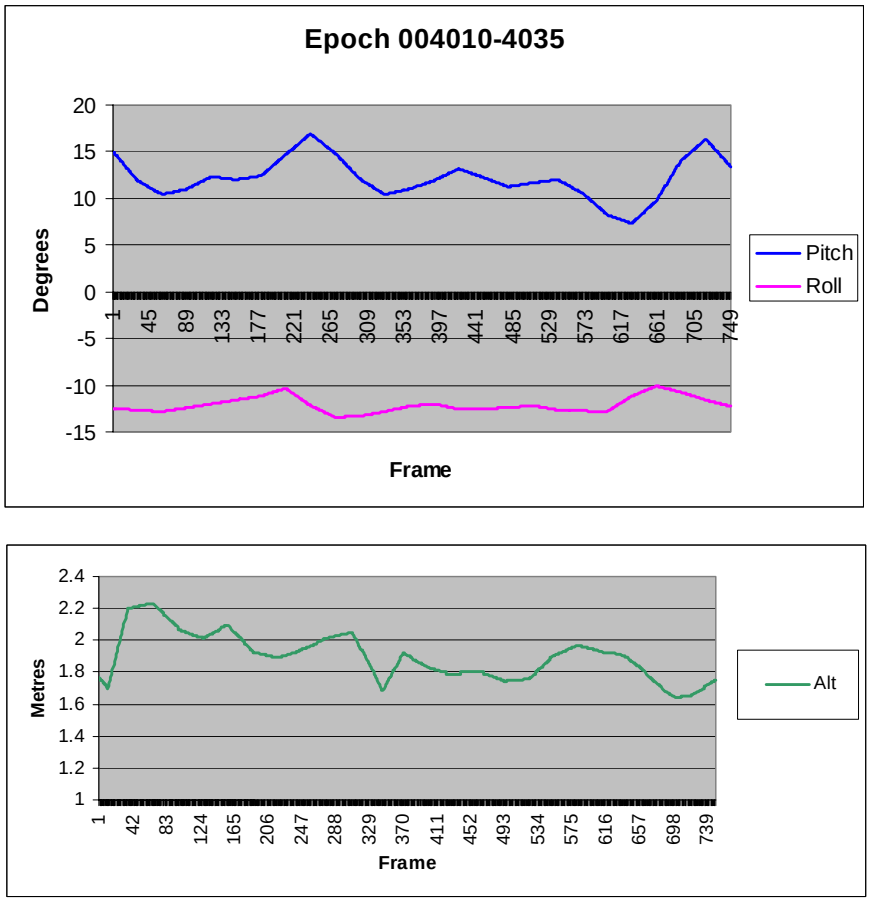

Figure A3-6 Mosaic from Epoch 4010-4035 with Attitude and Altitude Data

A3.2.4 Video Mosaic from Epoch 004033-004058

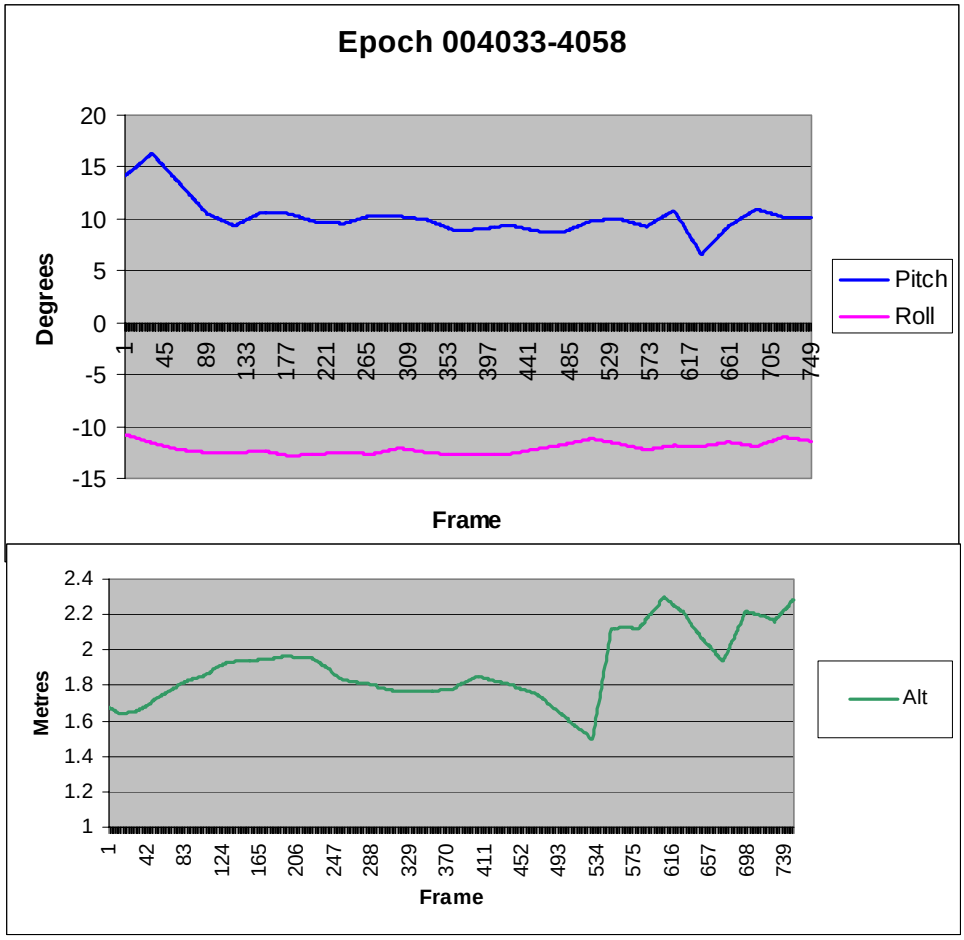

Figure A3-5 Mosaic from Epoch 4033-4058 with Attitude and Altitude Data