Implementation of a Wireless Underwater Video Link

by

James Paul Morash

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology
January 2008

Author ____
Department of Electrical Engineering and Computer Science
January 31, 2008

Certified by ____
Chryssostomos Chryssostomidis
Professor of Mechanical Engineering
Director, MIT Sea Grant
Thesis Supervisor

Certified by ____
Milica Stojanovic
Principal Research Scientist, MIT Sea Grant
Thesis Co-Supervisor

Accepted by ____
Arthur C. Smith
Professor of Electrical Engineering
Chairman, Department Committee on Graduate Theses

Implementation of a Wireless Underwater Video Link

by

James Paul Morash

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

January 2008

Sponsored by the Chevron-MIT University Partnership Program

Abstract

Autonomous underwater vehicles (AUVs) are increasingly being considered for remotely supervised missions, primarily for routine subsea inspection tasks currently performed by tethered remotely operated vehicles (ROVs). This project is a step in the development of a high speed, networked wireless communication capability for AUVs using MIT Sea Grant's software-defined Reconfigurable Modem (R-Modem) acoustic communications platform. We have demonstrated a test implementation of live streaming video with a digital camera connected to an R-Modem transceiver; and explored a range of tuning parameters for the video link.

Thesis Supervisor:	Professor Chryssostomos Chryssostomidis Department of Mechanical Engineering Director, MIT Sea Grant Program
Thesis Co-Supervisor:	Dr. Milica Stojanovic Principal Research Scientist MIT Sea Grant Program

Acknowledgments

First I have to thank Milica Stojanovic, for her enthusiasm, her sense of humor, and for showing me that acoustic signal processing can be fun. Thanks to Chrys Chryssostomidis, for finding me a great thesis project on short notice. Another big thank-you to our Turkish trio of communications engineers, Mehmet Aydinlik, Ahmet Turan Ozdemir, and Ethem Mutlu Sözer, for their hard work on the R-Modem testbed. Thanks also to the AUV Lab engineering team, Victor Polidoro, Justin Eskesen, and Dylan Owens, for last-minute machining and good times on Oahu. And thanks especially to my wife Melanie, who has helped me to succeed in this project, as in so many other endeavors, through her love and encouragement.

This project was generously funded by the Chevron-MIT University Partnership Program.

Table of Contents

1 Introduction

1.1 The need for acoustic communication underwater

The rapid advance of autonomous underwater vehicle (AUV) technology in the past two decades has led to a renewed push to replace subsea-industry-standard remotely operated vehicles (ROVs) with 'wireless' AUVs under remote supervision. In a 'supervisory control' [1] operating scenario, the AUV user would be able to examine real-time feedback from on-board sensors, decide on an appropriate course of action, and re-task the vehicle as necessary [2]. This mode of operation would combine the ease of use of an AUV with the advantages of human intelligence applied to data analysis and adaptive mission planning.

Early deployments of this technology will be limited to light-duty tasks such as routine inspections of subsea structures; the primary motivation is the cost savings realizable by elimination of the ROV tether. The surface assets required for successful deep-water deployment of a work-class ROV (even for simple non-intervention tasks like visual inspection) incur operating costs on the order of $100,000 per day. The high cost of ROV operations is primarily due to the engineering and logistical demands of tether management, including a dedicated ship with dynamic positioning capability. In complex environments or heavy seas, ROV operators must be wary of entanglement hazards, cable-induced disturbance of vehicle motion, and cable breakage, all of which are exacerbated by the increasing operating depth requirements of new oil and gas fields. [3]

AUVs lack a cabled connection to the sea surface and, as a result, must navigate without human aid or intervention. To reduce the complexity of this task, an AUV is typically deployed with a pre-programmed dive plan: a list of tasks to perform and navigation waypoints to follow. The AUV then relies on its limited intelligence to guide it safely along its course. Early AUVs were designed for open-water environments and wide-area surveys, where demands on perception (e.g. obstacle avoidance) and decision-making are low [4]. Modern AUVs are capable of more sophisticated autonomous behaviors, such as detecting, tracking and imaging man-made structures automatically [5], but are still confined to the state of the art in machine vision and artificial intelligence.

Regardless of the sophistication of the AUV itself, its fundamental purpose is to serve as a proxy for human presence underwater. While ROV technology can provide human operators with a real-time 'telepresence', thanks to the high-bandwidth communications link provided by the tether, AUVs are designed to collect and store a full set of data, then return to the surface for data download and post-processing. The non-real-time nature of standard AUV operations makes it difficult to perform a dynamic, adaptive survey; an ROV pilot may re-task his vehicle at any time to more closely examine a feature of particular interest. If something that merits further examination is found in the course of post-processing AUV data, the AUV must be sent on another dive to re-examine the target. In the case of underwater inspection (of hulls, pipelines, and other structures), real-time feedback from the remote platform may inform time-critical repairs in a way that post-processed data cannot.

The operational ideal is a robotic vehicle that combines the strength of these two platforms: the low deployment costs of an AUV, and the real-time feedback and control of an ROV. Such a vehicle will require a wireless communications link as an effective replacement for the ROV cable. The wireless link must support high-speed uplink data transfer, to provide a live feed of sensor data, while downlink needs will remain limited to transmission of intermittent high-level commands (specific tasks or navigation maneuvers) from the human operator to the AUV's on-board controller. An AUV with this type of high speed wireless connection to the surface may be used in a 'supervisory' mode: the vehicle will perform semi-independent inspection tasks, while connected to an operator interface that shows live updates of the resulting data, and provides for remotely re-tasking the vehicle on demand.

In the most typical application, where the AUV is intended to replace an ROV for visual inspection tasks, the uplink budget must be primarily dedicated to transmitting live images from an on-board camera to the operator. The ability to view images of the underwater environment in real time would demonstrate the viability of the proposed wireless concept of operations. Since any autonomous vehicle operating independently in the oilfield environment presents a safety hazard to other equipment (ROVs, pipelines, wellhead structures), our wireless supervisory link will also be critically important for real-time tracking of AUV position and speed, and remote emergency stop. Thus, whatever solution we develop for the wireless communication problem should enable our AUV to benefit from existing navigation infrastructure (e.g. acoustic beacons) as well as existing communication infrastructure, as it moves through the oilfield.

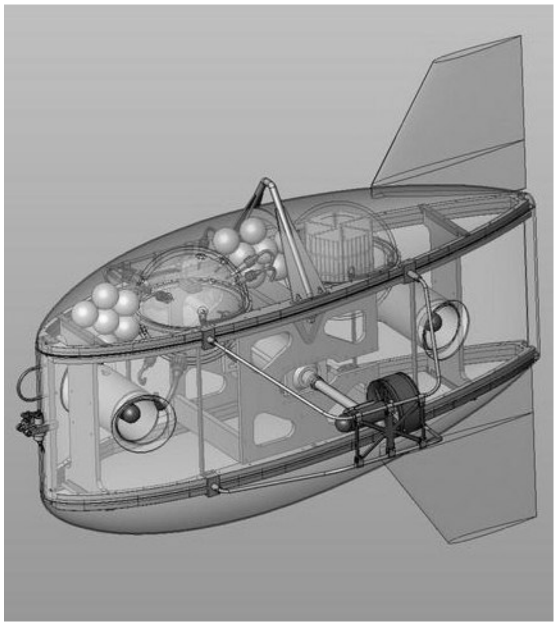
The implementation of a high-speed wireless underwater communications link is an area of active research. Seawater is essentially opaque to electromagnetic waves [6], precluding the use of radio frequency (RF) transceivers. Experiments with laser-based transceivers in the blue-green range of the visible spectrum have shown some promise, but thus far have proven unsuitable for connecting mobile platforms due to the need for precise optical alignment between transmitter and receiver. However, sound waves travel exceptionally well through seawater; low frequency sounds (~20 Hz) may even be detected across an entire ocean basin [7], while ultrasonic frequencies (>20kHz) are usable at ranges of hundreds to thousands of meters. Industry practice has settled on acoustic transmission as the key to effective real-time communication with sub-sea assets.

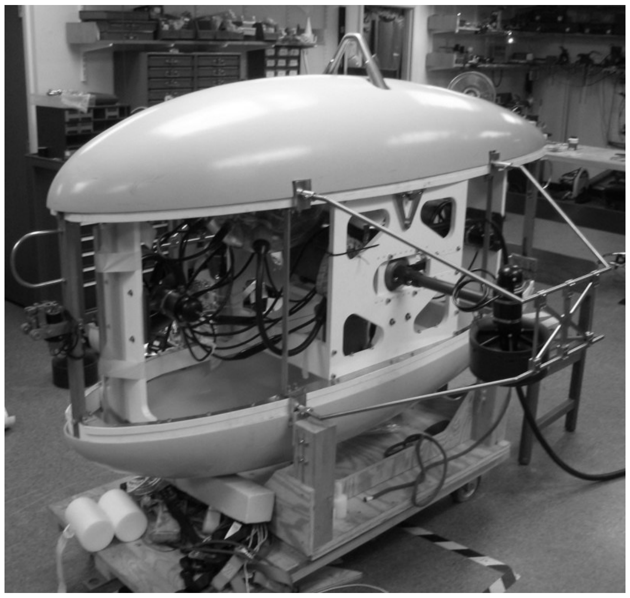
Because the field of underwater acoustic communications is still evolving rapidly, MIT Sea Grant (MITSG) researchers have chosen to develop a flexible new hardware platform to support communications experiments. This platform, the Reconfigurable Modem or “R-Modem” [8] is essentially a software-defined radio for the underwater realm. The R-Modem has several modular components: an acoustic physical layer (piezoelectric transducer and power amplifier), a fast, programmable digital signal processor (DSP), and a software-defined system model developed in MATLAB Simulink. The Simulink model is converted into DSP code that controls the entire digital communications signal chain. In this project, we have used the current version of the R-Modem as one component in a prototype wireless underwater communications system.

This project is part of a larger research effort focused on the particular application of automated AUV-based inspection of underwater structures. Our work was funded by the Chevron-MIT University Partnership Program, as part of a pilot study on the general subject of reducing production costs through increasing use of autonomous vehicles in offshore oil and gas operations.

The concept of operations is founded on wireless supervisory control of a high-maneuverability vehicle, allowing the operator to remotely inspect complex structures with real-time feedback from the sensor platform. Inspection targets of interest include nearly all offshore oil equipment designed for long-term deployments, including floating production, storage and offloading vessels (FPSOs), wellhead equipment, sea-floor pipelines, and vertical risers. Initial testing of the AUV-based inspection concept will focus on risers and riser flex-joints. These components are especially subject to wear and fatigue, caused external current forcing and vortex-induced vibration (VIV); thus they require the most frequent inspections. Applying AUV technology to the problem of riser maintenance not only provides an opportunity for substantial cost savings over ROV operations, but also an opportunity to increase the frequency and effectiveness of inspections. A reliable autonomous inspection platform could perform a cost-effective, detailed examination of a variety of subsea equipment on a daily basis, increasing operational safety and decreasing the likelihood of a catastrophic leak.

The author has spent the past six years as a member of the MIT Sea Grant research engineering team. Our team has recently designed and constructed a prototype hovering AUV, named *Odyssey* IV, which will serve as the AUV testbed for the offshore oil equipment inspection program. The *Odyssey* IV platform is deep-water capable, supports four-degree-of-freedom (4DOF) active control, and can carry large payloads (up to 100 liters of neutrally buoyant volume) [9], making it easy to integrate new sensors for inspection, as well as new equipment for acoustic telemetry and supervisory control. The design and testing of a demonstration version of this acoustic telemetry system is the subject of this thesis.





SolidWorks model *Odyssey IV* under construction

Figure 1. *Odyssey IV* deep-water hovering AUV.

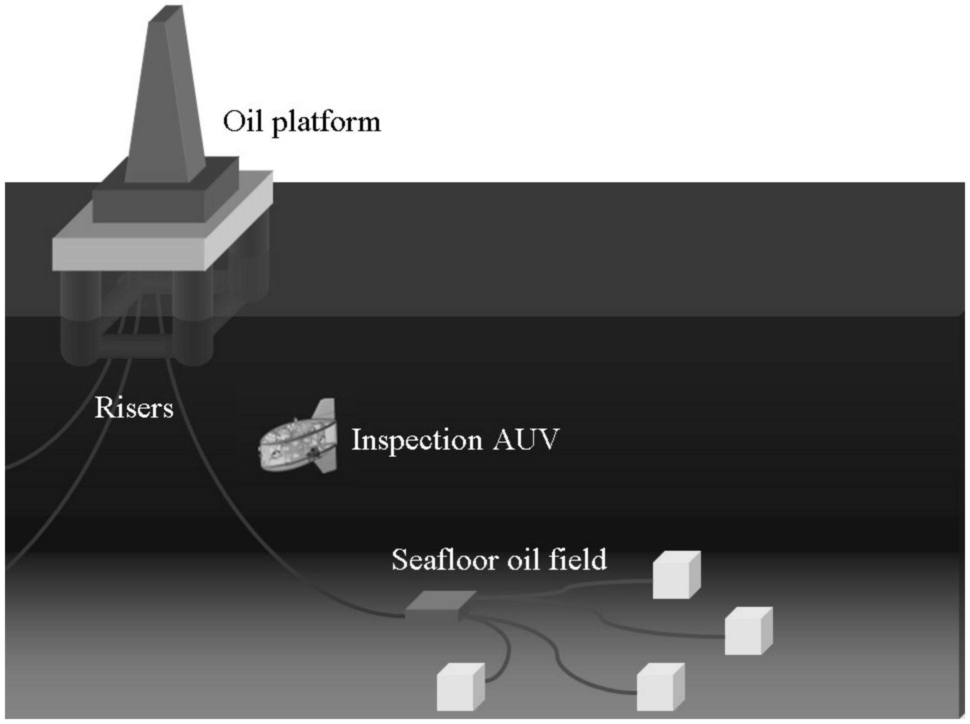


Figure 2. Autonomous riser inspection concept-of-operations.

2 Related Work

2.1 Acoustic modem design for video transmission

The concept of an underwater acoustic modem is not new. Early experiments in SONAR-based data transmission (as opposed to SONAR target detection) using modulated sound began in the 1950s. The earliest 'underwater telephones' transmitted analog voice modulated on a low-ultrasonic carrier; this technology is still used today in naval and commercial diving operations. Since those early days, the development of underwater acoustic technology for faster, more reliable transport of information has been an area of active research. Underwater acoustic modems are now in regular use in the field, primarily as low-bandwidth links to remote sensor platforms. Long-range acoustic links provide hourly or daily updates from semi-permanent sea floor deployments of ocean-bottom seismometers or acoustic Doppler current profilers, with no need for installation of an expensive cable [10]. Shorter-range acoustic communication is regularly used for high level command and control of commercial AUVs, but uplink telemetry is generally limited to sparse updates on vehicle position and heading [11].

The design of a new acoustic data link is driven by the requirements of the application at hand and by inevitable engineering trade-offs in an effort to balance three design metrics: complexity (is it practically achievable?), robustness (how often will the link fail?), and performance (throughput and bit error rate). An engineer wishing to make a contribution to the state of the art must decide which parameters are most in need of improvement. In the case of remote supervision of underwater vehicles, the overall throughput required to support effective supervisory control is dependent on the needs of the uplink (from subsea to surface) end of the system; downlink needs are limited to simple, specific control commands.

The most common request from potential AUV users is to provide an effective replacement for the high-quality real-time video stream available over the cabled connection of an ROV. Video signals inevitably have a much larger information content per unit time (effectively, bandwidth) than any other feedback signal that might be of interest to the AUV operator. For comparison, a typical acoustic communication link might operate at a carrier frequency of some tens of kiloHertz, and support a bandwidth of five kiloHertz; while North American standard (NTSC) analog broadcast television occupies channels more than four megaHertz wide. In the more bandwidth-efficient digital realm, we might consider the recent popularity of streaming video on the World Wide Web. The success of YouTube and similar video-dependent web applications was only possible after widespread deployment of high-speed (hundreds of kilobits per second) Internet connections in the homes of average users; in the days of dial-up Internet service, the act of downloading a single, brief video clip would saturate one's connection for hours. Demand for digital video continues to increase; the newest digital cellular telephone networks are able to take advantage of recent advances in video encoding (H.263, MPEG4 part 2, MPEG4 AVC / H.264) to provide low-resolution streaming video on mobile, wireless handsets at minimum bit rates of only 64 kilobits per second (kbps). While 64 kbps is a typical 'baseline' operating point for MPEG4, the underlying compression algorithms are designed for usable performance at much lower bit rates (down to approximately 20 kbps) [12]. More sophisticated encoding methods based onwavelet transforms have shown some promise for very low bit rate underwater applications [13].

For the first several decades of acoustic modem research, it was thought to be impossible to successfully transmit digital data underwater at such high bit-rates; it appeared that the higher-order modulation schemes necessary to achieve high throughput would exceed certain basic limits imposed by the typical underwater channel impulse response, the relatively low carrier frequencies available, and the long propagation delays associated with underwater sound. Breakthroughs in the 1990s, achieved by applying lessons learned in the field of digital radio communications, have led to new techniques for solving some of these apparently fundamental problems. In the past ten years, several commercial vendors have developed off-the-shelf acoustic modems with claimed peak throughput comparable to the early days of dial-up Internet service:

Table 1. Commercial acoustic modems, 2007.

Manufacturer	Model	Advertised Minimum Bit-Rate	Advertised Maximum Bit-Rate
Teledyne Benthos	TeleSonar	360 bps	15,360 bps
LinkQuest	UWM series	80 bps	38,400 bps
Sercel	MATS	20 bps	26,500 bps
WHOI / Hydroid	MicroModem	80 bps	5,400 bps
EvoLogics	S2C series	unspecified	6,500 - 33,300 bps

The wide range of operating bit-rates quoted for each product line [14] deserves particular attention. Commercial acoustic modems are offered in a number of application-specific hardware and firmware configurations, selected by the customer with guidance from the vendor's application engineering staff. Field testing of commercial acoustic modems has shown that the high throughput versions showcased in product literature can provide a reliable link only under the most benign acoustic conditions. Higher background noise levels, longer point-to-point ranges, and increased multipath (echo) effects in shallow water all tend to lead the customer toward the low-bit-rate, high-reliability end of a given product line. Once a particular modem is selected for a given application, the acoustic link must be thoroughly field-tested to ensure that the system behaves as expected. Commercial acoustic modems do not typically offer an in-depth interface for performance tuning, due to the proprietary nature of the design; as a result, a modem that works perfectly under benign channel conditions may be completely unusable in another environment. Even at their performance peak, none of the commercial offerings presented can match the minimum bit-rate of modern wireless video streams (64 kbps MPEG4, as described above). To understand where this limitation comes from, and how the proposed work may be able to improve upon the commercial state of the art, requires a more detailed examination of the behavior of underwater sound.

The underwater acoustic communication channel is very different from the more familiar terrestrial radio channel. Foremost is the variability of the underwater channel, both from moment to moment (coherence times under one second are typical in the open sea) and from place to place. Point-to-point transmission of an acoustic signal is subject to severe distortion in time and frequency [15]; without some form of compensation for channel effects, received signals appear indistinguishable from random noise (Figure 3 (a)):

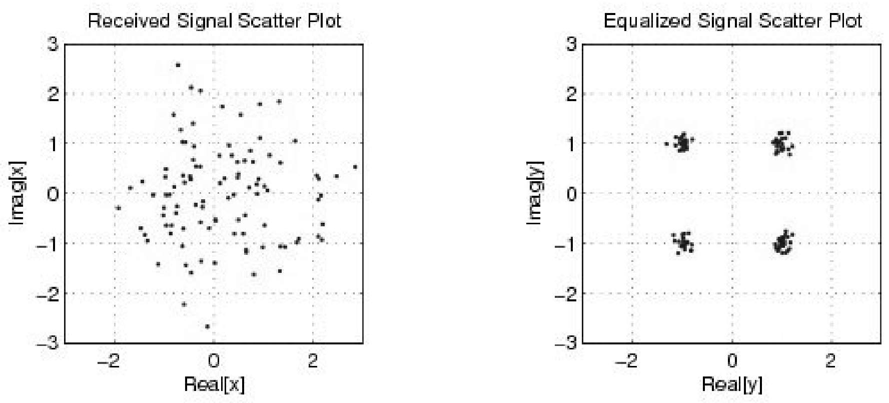


Figure 3: Example QPSK constellation (a) before and (b) after equalization.

An acoustic modem may compensate for the negative effects of the channel by estimating the channel impulse response function on the fly, then deconvolving the received signal using an adaptive equalizer (see Figure 3 (b)). The major channel effects that must be countered include phase distortion, induced by platform motion, volume scattering and impedance variations in the water column; severe attenuation (a typical acoustic modem receiver must cover over 120dB of dynamic range) that exhibits non-linear variation with frequency and path length; and multipath: reflections from the sea surface, the sea floor, rocks, man-made structures, etc., resulting in multiple (anywhere from two to fifty) overlapping copies of the transmitted signal with similar amplitudes, and total delay spreads ranging from 1 to 100 milliseconds. In mobile systems, Doppler shift may also be significant; a vehicle speed of some few meters per second produces a detectable shift on the order of tens of Hertz.

2.2 Prior work at MIT Sea Grant

Although the work described in this thesis was largely dedicated to the refinement of a point to point communications capability, the overall program of research at MITSG has the development of robust algorithms for acoustic networking as a primary goal. The further development of the R-Modem as a platform for high-speed telemetry in a complex acoustic environment will benefit from ongoing efforts in faster point to point acoustic communication (physical layer), and in more efficient use of the acoustic channel by multiple users (network layer).

2.2.1 Point to point communication

By far the most benign environment for underwater communication is the so-called "deep sea vertical channel", where transceivers are vertically aligned, but separated (in depth) by some distance (meters to kilometers). The actual water depth is unimportant, so long as it is large relative to the separation between the transceivers (thus reducing the influence of surface and bottom reflections). Each transceiver uses a narrow-beam acoustic transducer, in order to transmit and receive sound with high signal to noise ratio (SNR) and to minimize the effects of multipath. In this relatively open environment, spurious echoes are few, and thermal layering in the water column has a minimal effect on transverse propagation.

Recent experimental work at MITSG explored the performance limits of an acoustic communication system in a near-ideal vertical channel [16]. The experiment in question used a pre-recorded video clip as a data source, and a pair of laptop PCs running MATLAB for pre-processing, playback, recording, and post-processing. Through-water transmission of MPEG2 video was demonstrated at rates up to 150 kbps, at 10m vertical separation. The received data stream could only be decoded into a readable format after careful post-processing with a hand-tuned equalizer.

The equalizer technology in question, originally described in [17], has been refined and tested over the past decade in a variety of channel conditions at many different carrier frequencies and bit-rates. This method, which combines a fractionally spaced recursive least-squares decision feedback equalizer (RLS-DFE) with an integrated digital phase-locked loop (PLL), has been shown to yield performance close to that of an optimal maximum likelihood detector at much reduced computational complexity [18]. The fractionally spaced DFE corrects for small symbol timing misalignments, decodes individual symbols, then uses the result of each symbol decision to subtract predicted multipath interference from subsequent symbols prior to decoding. Use of the RLS algorithm for error minimization yields predictably fast equalizer convergence in a variety of channel conditions, while the integrated PLL compensates for high-speed phase fluctuations induced by the environment and, simultaneously, for carrier frequency offsets induced by Doppler distortion. Through the use of these techniques, our research group has been able to repeatedly demonstrate coherent post-processing of phase-modulated signals recorded in quickly varying, multipath rich environments.

In the current work, we have implemented the RLS-DFE with PLL in a modular, reusable Simulink block. This has made it possible to move beyond the realm of post-processed experiments and to test the performance of the equalizer in real time on the R-Modem embedded DSP platform. In the last two years, two different optimized, embedded implementations of this type of equalizer [19, 20] have been demonstrated in the field. Each of those projects relied on dedicated engineering teams to produce specifically targeted DSP code for specific physical layer parameters, while our Simulink implementation is intended to be reusable and reconfigurable. The use of Simulink in developing R-Modem firmware is described in greater detail in Chapter 3. The new system is capable of real-time performance, using live images captured from a digital video source. Chapter 4 describes our preliminary steps taken to investigate the acoustic video transmission design parameter space.

2.2.2 Underwater acoustic networks

In the deep-water oilfield environment, we cannot rely on idealized assumptions of undistorted vertical acoustic propagation. In the Gulf of Mexico, for example, sharp thermoclines often cause near total reflection in the vertical channel, forming an acoustic barrier that prevents long-range communication with the surface [21]. Below, on the sea floor, acoustic communication is limited by ambient noise from heavy industrial operations (drilling, completion, production) and competing transmissions (imaging sonar, Doppler velocimetry, navigation beacons). In the oilfield, as in other equipment-intensive subsea deployment scenarios proposed for scientific expeditions and naval exercises, we believe reliable, high speed acoustic communication will ultimately require a multi-hop network architecture and short-range, high frequency horizontal transmission.

Oilfield acoustic networks will depend on one or more stationary 'sink' nodes to forward data packets to the surface over a cabled connection. The higher bandwidth available and higher effective SNR due to the nonlinear increase in attenuation at high carrier frequencies will inevitably drive down maximum effective transmission ranges. Ease-of-deployment constraints, and the integration of mobile nodes into the acoustic network, will drive the use of horizontally omni-directional (toroidal beam-pattern) transducers (thereby increasing multipath interference). It may be desirable to integrate an acoustic positioning capability into the data network, in order to efficiently combine real-time tracking and navigation (for AUVs, ROVs, and other assets) with data transmissions, optimize routing on a geographic basis, etc. Thus, the acoustic environment becomes more demanding.

Future work with the R-Modem platform will increasingly require additional layers of functionality to manage channel contention (medium access control or MAC) and application layer data transport demands, in addition to faster point to point transmission. The current work includes only a preliminary look at adapting routing and MAC layer controls to the R-Modem platform; full integration into a networked, multi-vehicle environment (in the oilfield or elsewhere) will demand these higher levels of functionality.

3 System Design

This chapter describes the design and implementation of a hardware testbed for experiments in live, real-time acoustic video transmission, followed by the results of experimental investigation of the design parameter space (encoding, modulation, carrier frequency, etc.) in Chapter 4. The current work was distinguished from prior work on this topic primarily by the intent to produce a complete, end-to-end system that would be field-ready. We have successfully integrated off-the-shelf parts with custom-created control software, and produced a wireless telemetry link that can capture live digital video images from the AUV's point of view, encode them in real time for acoustic transmission, and display decoded streaming images at the 'topside' end of the acoustic link. Due to speed constraints of the existing hardware testbed, we have been unable to demonstrate true video-rate (multiple frames per second) image transfer, but our real-time implementation does provide a foundation for future work.

Having demonstrated a minimally working wireless video connection, we have taken care to examine several future directions for this project while continuing to work within the limitations of the existing hardware testbed. The remainder of Chapter 3 describes the demonstration prototype system architecture. Our original goal of varying video encoding and acoustic modulation parameters to determine their effects on video stream quality has been constrained by current capabilities, as detailed in Chapter 4. Several options for high-level system redesign intended to support more complex experiments are described in Chapter 5.

3.1 Digital camera

The hardware building blocks of this project are divided into two sub-systems: the digital video camera, and the acoustic modem. Camera choice is guided in this case by ease of integration both with the AUV control system and with the digital interface to the acoustic link. The demonstration prototype platform is based on the Axis 207 Network Camera from Axis Communications, Inc. Several key features of the Axis 200 series product family have contributed to a successful proof-of-concept demonstration. All Axis 200 cameras have a direct Ethernet interface, with a simple built-in Web (HTTP) server for remote configuration and live display of digital video. Users may simply point their browser to the Internet Protocol (IP) address assigned to the Axis 207 and retrieve continuous video streams or individual captured images on demand. The camera may be configured on-the-fly to output individually encoded JPEG images, Motion-JPEG video, or MPEG4 video. Each encoding format may be adjusted by the user to trade off image quality against bandwidth demand, by altering color depth, image resolution, compression quality, and (in the case of MPEG4) by limiting the overall bit-rate of the video stream with a fixed or variable cap. The range of available tuning parameters is summarized in Table 2.

Table 2. Axis 207 video output options.

Data format	**Image size**	**Image / Video complexity**
JPEG	4:3 aspect ratio	JPEG quality factor, 0 to 100
single frame capture		(compression loss)
Motion JPEG	Pixel sizes:	
series of frames,	160 x 120	Color or grayscale
individually encoded	176 x 144	
	240 x 180	Frame rate:
MPEG4	320 x 240	1 to 30 frames per second
series of frames	352 x 288	
high efficiency interframe encoding	480 x 360	MPEG4 bit rate:
	640 x 480	set maximum limit, and/or
		average target. 1 to 1000 kbps.

Recent test deployments of a new AUV carrying an Axis network camera have shown that this technology is simple to integrate and performs well underwater [22]. Figure 4 (a) shows sample output from a wide-angle Axis 212 camera deployed on a Hawaiian coral reef aboard a small AUV. As in our acoustic wireless video system, the camera itself is responsible for real-time encoding of high-quality video into a standard digital format, making it easy to connect other Ethernet-capable devices to the output stream for storage or further processing. In the case of the *Reef Explorer* AUV pictured in Figure 4 (b), the Axis 212 is connected directly to an Ethernet-interface 900MHz radio modem, which is towed on a small antenna float permanently attached to the AUV. In the prototype acoustic wireless video system, the Axis 207 camera is instead connected to a custom-engineered acoustic modem based on an off-the-shelf digital signal processor (DSP) platform.

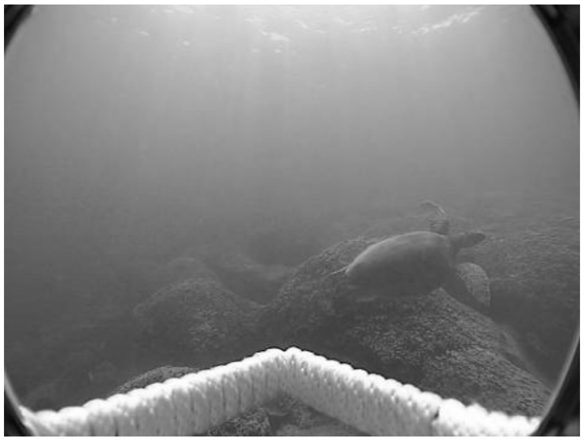
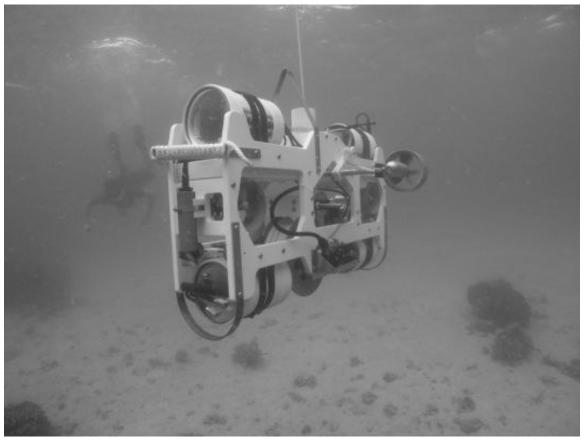



Figure 4. (a) Underwater video frame capture from Axis 212 camera.
(b) Axis camera installed in *Reef Explorer* AUV.

3.2 Acoustic modem

The acoustic modem used in the current work is the MIT Sea Grant-developed Reconfigurable Modem or 'R-Modem' [8]. The R-Modem platform provides a complete acoustic modem solution: off-the-shelf DSP hardware (Spectrum Digital DSK6713), custom-built analog front end (MITSG designed), and custom-built firmware – designed on a standard PC in MATLAB and Simulink, and converted to real-time code for operation of the DSK6713.

The R-Modem testbed hardware is currently installed in a pair of NEMA 6P rated steel enclosures. Each enclosure is connected to local (DC) ground for shielding against electromagnetic interference; the acoustic transducers are wired using balanced, shielded cables to studio-microphone style 3-pin XLR connectors, which meet the continuous shielded enclosure at each steel bulkhead. For further isolation, each R-Modem is powered by a transformer-isolated DC supply from 120VAC utility power, and connected via USB to separate laptop PCs for debugging and software updates.

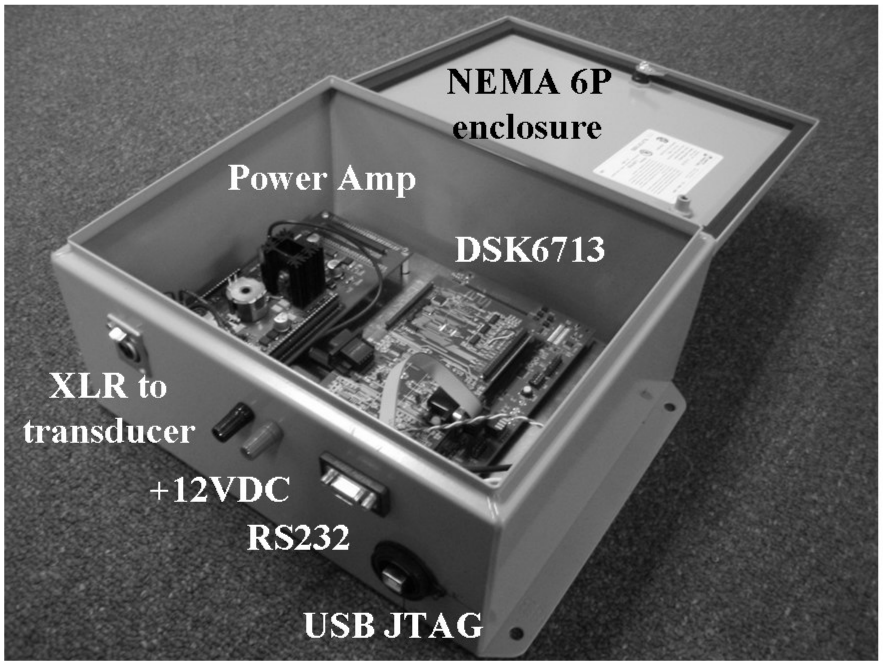


Figure 5. Acoustic modem prototype hardware.

3.2.1 DSP hardware

A typical laptop PC running MATLAB and Simulink would be difficult to integrate into an AUV; the R-Modem is designed to be deployable on mobile platforms in the field, underwater, with no hardware changes required. The R-Modem is based on an embedded DSP processor, the Texas Instruments (TI) TMS320C6713, which has made it possible to build a physically compact, low-power system (~2W typ. @ 5V supply). The 6713 is a general purpose 32-bit floating-point signal processing engine, capable of up to 1800 million floating point operations (MFLOPs) or 600 million multiply-and-accumulate (MMACs) per second. It is designed to perform real-time operations on digital signal streams – filtering, decimation, detection – far more efficiently than a general purpose PC processor. We have chosen a floating-point DSP for ease of software development, despite the much greater speed (in MMACs) available from a fixed-point processor. Due to the high dynamic range of underwater signals and the numerical requirements of nonlinear equalization techniques, it is quite difficult to choose appropriate fixed-point scaling at each link in our signal processing chain.

The Spectrum Digital DSK6713 is a developer's kit, designed in cooperation with Texas Instruments, that is intended to allow low-volume users to develop and test high-speed signal processing systems without needing to design a fully customized set of hardware. The DSK6713 has two stereo audio inputs and two stereo outputs, connected to a TI TLV320AIC23 audio codec (sampling up to 96 ksps / 16-24 bit delta-sigma); 8 MB SDRAM and 512 KB Flash; an on-board JTAG interface (via USB) for target programming and debugging; and a +5V input power supply that produces regulated low-voltage power for the DSP core and peripherals. As with other Spectrum Digital DSKs, the DSK6713 has a standardized stack-through daughtercard interface that makes it relatively easy to add new hardware components. Our R-Modem design uses an RS-232 UART add-on card to provide the DSK6713 with a serial port.

3.2.2 DSP firmware

The 6713 DSP is designed by TI to be programmed in assembly or C code by trained developers, using the Code Composer Studio (CCS) integrated development environment. To support rapid application development, and testing of new ideas by students unfamiliar with the intricacies of DSP architecture, the 6713 may alternatively be programmed using a code-generation environment built into MATLAB and Simulink. Our R-Modem project relies on MathWorks'

Link for Code Composer Studio and Simulink Real-Time Workshop. Together, these software packages make it possible to design every layer of the embedded DSP software in the Simulink graphical programming environment. Simulink blocks control memory allocation, task switching, application-level (serial port) I/O, analog-to-digital (ADC) and digital-to-analog (DAC) conversion, quadrature modulation and demodulation, filtering, decimation, interpolation, equalization, encoding, and decoding. Each block has a set of tunable parameters for user customization.

The R-Modem Simulink model currently contains more than 100 separate blocks that comprise a complete software-defined physical layer. In principle, any one block (e.g. the equalizer) may be changed at will without affecting the rest of the model. A finished Simulink model is converted to C code by Real-Time Workshop Target for TI C6000, then compiled and downloaded to the DSP by Link for Code Composer Studio, all in a matter of minutes.

The DSP communicates with MATLAB and CCS on the host PC through a USB JTAG emulator. The JTAG emulator is directly connected volatile RAM and nonvolatile Flash memory on the target board, allowing the user to download compiled programs and debug running code. Through JTAG Real-Time Data Exchange (RTDX) we are able to watch variables, perform stack traces, etc., then adjust Simulink parameters, recompile the model, and reprogram the target for another run. Unfortunately, the use of RTDX creates a significant 'observer effect'; in a real-time system, any CPU time dedicated to data transfer between the embedded target and the host PC is CPU time withheld from other, perhaps more critical tasks. At the rates demanded of the 6713 by our Simulink model, practical RTDX transfers are limited to updates of a few hundred bytes at the end of each round of packet decoding, when the processor is relatively idle.

3.2.3 Analog front end

The transmit power amplifier and the receive preamplifier are built on a single custom PCB designed to the Spectrum Digital daughtercard specification. Transmitter power-enable, and switching between transmit and receive modes, are controlled by logic-level outputs from the DSP, while the transmitter input and preamp output are connected to the DSK mini-stereo jacks via separate shielded cables. In transmit mode, the output DAC stage is fed into a Class B linear power amplifier, constructed as a push-pull pair of MOSFETs with split (±12V) supply. The Class B output drives the primary of a step-up transformer, which boosts the output voltage by a factor of 4 in order to drive the acoustic transducer at up to 15 Watts. Transmit power control is achieved by adjusting a rotary potentiometer connected as an adjustable voltage divider after the transformer secondary. The secondary side of the step-up transformer is connected to a double-pole double-throw electromechanical relay for transmit/receive switching. In receive mode, low-level signals from the transducer are fed into a two-stage fixed-gain preamplifier (~55 dB) built from discrete op-amps. Anti-alias filtering is handled on the DSK6713 at each audio input.

3.2.4 Serial interface

The R-Modem hardware platform is capable of high-speed serial port communication (RS-232), up to 250 kbps. While this data rate is more than adequate to guarantee that the application interface can keep up with the acoustic physical layer, this serial interconnect cannot be directly connected to the Ethernet-only Axis camera. While there is an Ethernet daughterboard on the market specifically designed for Spectrum Digital DSKs, the addition of an Ethernet controller would also require the addition of new Simulink blocks to our R-Modem model, in order to manage data transport correctly. The Simulink block-set designed for Ethernet integration on the DSK is costly (~$8000), and incompatible with the version of Code Composer Studio shipped with the DSK6713.

In the final application, we can solve this problem through full AUV integration of the subsea end of the acoustic link. Once integrated, the R-Modem will be connected to the MITSG standard AUV control computer, which is an industrial PC/104 stack. The R-Modem will communicate with the PC/104 stack over an on-board RS-232 port, and the Axis camera will communicate with the PC/104 stack through an on-board Ethernet switch.

In the current work we have temporarily substituted a laptop PC running MATLAB for the AUV control computer. Running a complete MATLAB environment on the 'subsea' node enables rapid development and full debug support of the R-Modem through the USB JTAG interface. In future experiments, a debugged, field-ready R-Modem is expected to operate with no host JTAG connection, running firmware stored in on-board Flash. Similarly, we have connected the 'topside' R-Modem to a second laptop PC running MATLAB.

The complete system architecture is pictured in Figure 6. The addition of the AUV host processor to the system will ultimately allow for more sophisticated, adaptive use of the acoustic link, as well as providing an opportunity to test additional video encodings that are not provided by the Axis hardware but are readily implemented in software. As explained in Section 3.3, we are currently depending on the 'subsea' laptop processor to re-encode digital video frames into acoustic packets suitable for DSP processing. This general concept of combining a dedicated signal processing engine with an embedded general-purpose PC processor, in order to provide a more robust and flexible communications transceiver, is explored in more detail in section 5.2.

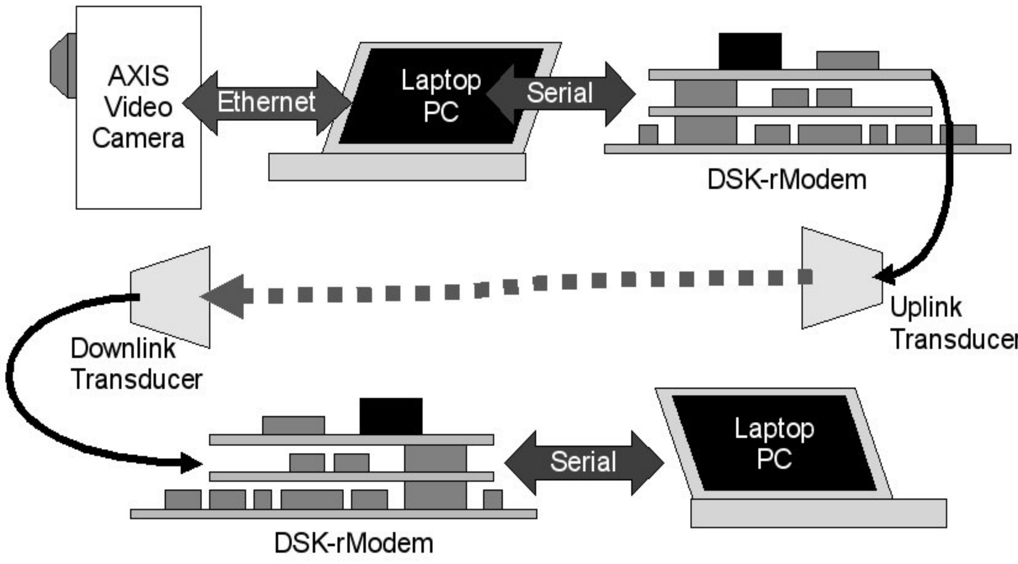
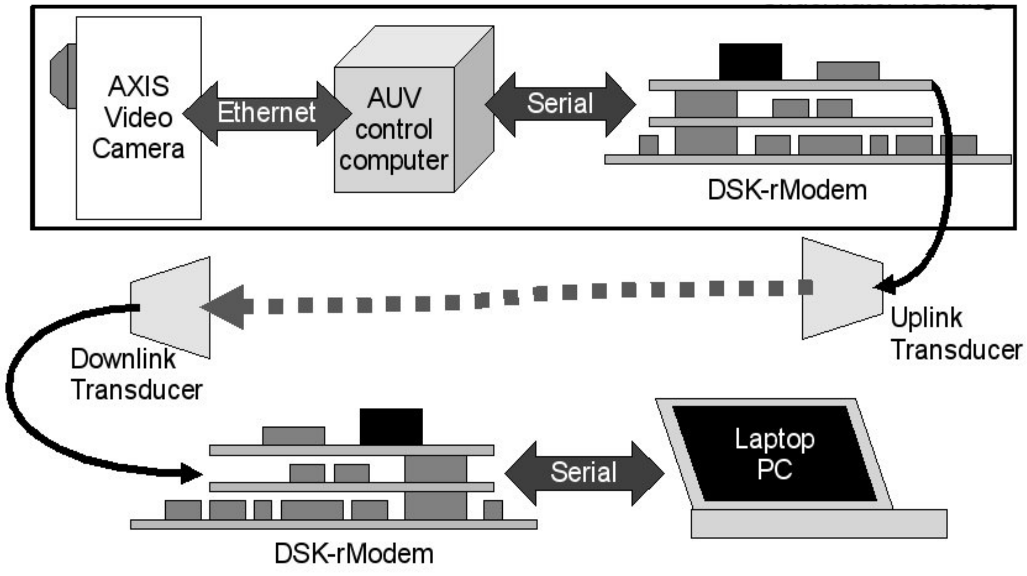


Figure 6: (a) Wireless video link, as built



(b) Wireless video link, future AUV integration

3.3 Image encoding for acoustic transmission

We have taken advantage of the MATLAB rapid development environment available at both ends of the prototype acoustic link to adapt our data payload strategy to the evolving capabilities of the existing hardware. Our initial plans for data encoding focused on the state of the art in high efficiency video compression, as described in the Introduction. In the course of this project we have adjusted our immediate throughput target from “very low bit-rate” (kpbs) to “extremely low bit-rate” (tens to hundreds of bps) video compression regimes. As a result, the decision was made to concentrate on transmission of individually compressed images, instead of implementing a more complex encoding that would provide true video (frame-to-frame) compression. This decision was partly driven by the observation of very high bit error rates (> 5%) in initial R-Modem tests, as individually encoded frames were thought to be better suited to error recovery.

3.3.1 Packetization constraints

Our initial plans assumed that the acoustic data link would be fast and effectively transparent – bits in, bits out, with the occasional dropped packet. Transferring individual images would be no more than a simple file-transfer operation, breaking a binary JPEG image into packet-sized chunks and reassembling it at the receiver. A video stream would require a more sophisticated approach, with careful tuning of the video encoder to work within the constraints of our chosen packet size and maximum sustained bit-rate.

These plans were changed to accommodate the level of performance realistically attainable given our short development time-frame. In the current implementation, the R-Modem serial interface is configured to wait for input in blocks of 48 bytes (which we can also think of as ASCII 8-bit characters, or 8-bit integers). A timer starts upon receipt of the first byte and the unfilled portion of each 48-byte block is zero-padded when the timer expires. Each 48 byte block is then transmitted as an acoustic packet. At the receiver, as each packet is decoded, 48 bytes at a time are sent to the host PC serial port.

This extremely simple interface design has no commands, no control characters, no start or end delimiters for packets, no parity bits, no cyclic redundancy checks. That said, it does act as a near-transparent data pipe – as long as the transmitter pauses every 48 bytes for an interval sufficient to receive and decode the most recent packet. Suggested improvements to the R-Modem serial interface are reserved for Chapter 5. For the time being, we have chosen to work around the existing implementation.

3.3.2 Image compression strategies

The question, therefore, is: given 48 bytes to work with, and a relatively high bit error rate, what is the best way to transmit digital images? In the prototype implementation, we have used two strategies: reduce the size of the captured image as much as is practical, then divide it into small self-contained blocks, which allows some measure of error recovery without needing to re-transmit bad packets.

By default, the Axis 207 camera captures VGA resolution (640x480) full-color images.

Rather than trying to transmit up to 7.3 megabytes of data (640 x 480 x 24 bits per pixel) in 48-byte fragments, we have instead selected a much lower resolution (96x72) gray-scale output image using the camera's built-in web-based configuration tool. In bench tests, we found this to be an effective minimum intelligible size and color depth, sufficient to demonstrate the system concept. Each frame is 96 x 72 x 8 bits per pixel = 6912 bytes uncompressed.

Working with such low-resolution images is unusual. Algorithms found in the literature that are specifically designed for underwater images are meant to take advantage of low contrast, blurring, etc. in order to perform lossy compression without noticeable loss of image quality [13]; at our chosen resolution there is little to be gained by such strategies. We have examined several different very simple encoding schemes in an effort to make the best of our limited throughput. After some unsatisfying attempts at coarse compression of the entire image at once (using singular value decomposition, and whole-image discrete cosine transformation) we have settled on re-implementation of the core JPEG compression algorithm, with some important modifications.

JPEG is well known as the *de facto* standard for compression of real-world images (digital photos). The core JPEG algorithm operates as follows:

(1) break the image into blocks, typically 8x8 pixels

(2) take the discrete cosine transform (DCT) of each block

(3) quantize each DCT block to eight-bit signed integers

(4) compress each DCT block using entropy coding

(5) write the entire image to a compressed file, including Huffman dictionary

Steps 3 and 4 are the steps that make JPEG compression so efficient. The components of any given DCT block may range over several orders of magnitude, depending on the image in question. In step 3, each DCT coefficient is divided by its corresponding quantization factor, based on an 8x8 matrix of values which is used for the entire image, in compression and decompression. The quantization step allows us simultaneously to reduce the number of bytes required to store each coefficient (reducing 32-bit floating point numbers to 8-bit signed integers) and to discard unimportant data (lossy compression). There are several empirically derived JPEG quantization matrices in the literature [23], with the magnitude of each of the 64 factors determined by averaging the DCTs of a number of real-world images and weighing the dynamic range of each DCT coefficient accordingly. After this step, only the most significant high-frequency components of the DCT block remain, all others being rounded to zero. Thus, after quantization, the DCT block is reduced to a sparse matrix, with nonzero elements clustered towards the top left.

In step 4, the quantized DCT blocks are losslessly compressed using entropy coding – run-length encoding ordered by spatial frequency (recording repeating values only once), followed by Huffman coding. This step takes advantage of both the sparsity of the quantized blocks and the reduced range of possible values (8 bit signed integers, vs. floats) to create an efficient variable-length code. JPEG files may be Huffman coded sequentially (block by block) or progressively (coefficients of all the blocks in spatial-frequency order), with progressive encoding giving somewhat better compression.

In the current application we have chosen to eliminate the entropy coding step and as such could not work directly with binary JPEG files. Bit errors in received packets are unpredictable. If we rely on variable-length codewords and the packet(s) containing the codeword dictionary are corrupted, even a single bit error could result in the entire image being decoded incorrectly. At our low image resolution (and low block count) the savings realizable by entropy coding are less compelling in any case. We are primarily concerned, for the time being, with transmitting an intelligible image in the minimum possible time. Our prototype implementation simply tolerates occasional corrupted packets with no effort to detect or re-transmit them.

Instead we have adopted a simpler strategy. Our prototype MATLAB code, running on a laptop PC, downloads an individual JPEG-compressed video frame from the Axis camera over HTTP, and reads it into a MATLAB array (effectively decoding the JPEG image into an uncompressed bitmap). The resulting bitmap image is then resized (to 96x72 pixels), and split into 108 8x8 blocks. Each block is discrete-cosine transformed and quantized, effectively re-implementing the first half of traditional JPEG compression. The quantized blocks then undergo a further lossy compression step by simple truncation: only the upper-leftmost 15 values (of 64) in each block are kept. The higher-frequency components are simply discarded. We arrived at the number 15 after averaging a large number of quantized DCT blocks. A plot of the average values has a distinct upper-left-triangular shape, with the highest value at the DC component and a sharp fall-off in magnitude towards the diagonal (see Figure 7).

Each block is thus reduced to a set of fifteen eight-bit signed integers. Acoustic modem packets are formed as a string of 46 bytes: a packet number, indicating each packet's place in the transmission sequence, followed by 45 quantized DCT coefficients. We have observed that the first byte of a packet is rarely corrupted, making simple packet numbering a safe method of organizing incoming data. Images are currently transmitted at a fixed resolution known to transmitter and receiver, so no negotiation or other overhead is required. Each three-block packet is self contained and decoded independently at the receiver. A second host PC receives each 48-byte packet as it is decoded. The three blocks in the new packet are marshaled into 8x8 arrays, element-wise multiplied by the quantization matrix, and inverse discrete cosine transformed. The reconstructed image blocks are then placed, according to packet number, in their assigned positions within the slowly assembled 'topside' image. This encoding scheme is readily adaptable to a variety of image sizes as long as the image is evenly divisible into sets of three 8x8 blocks.

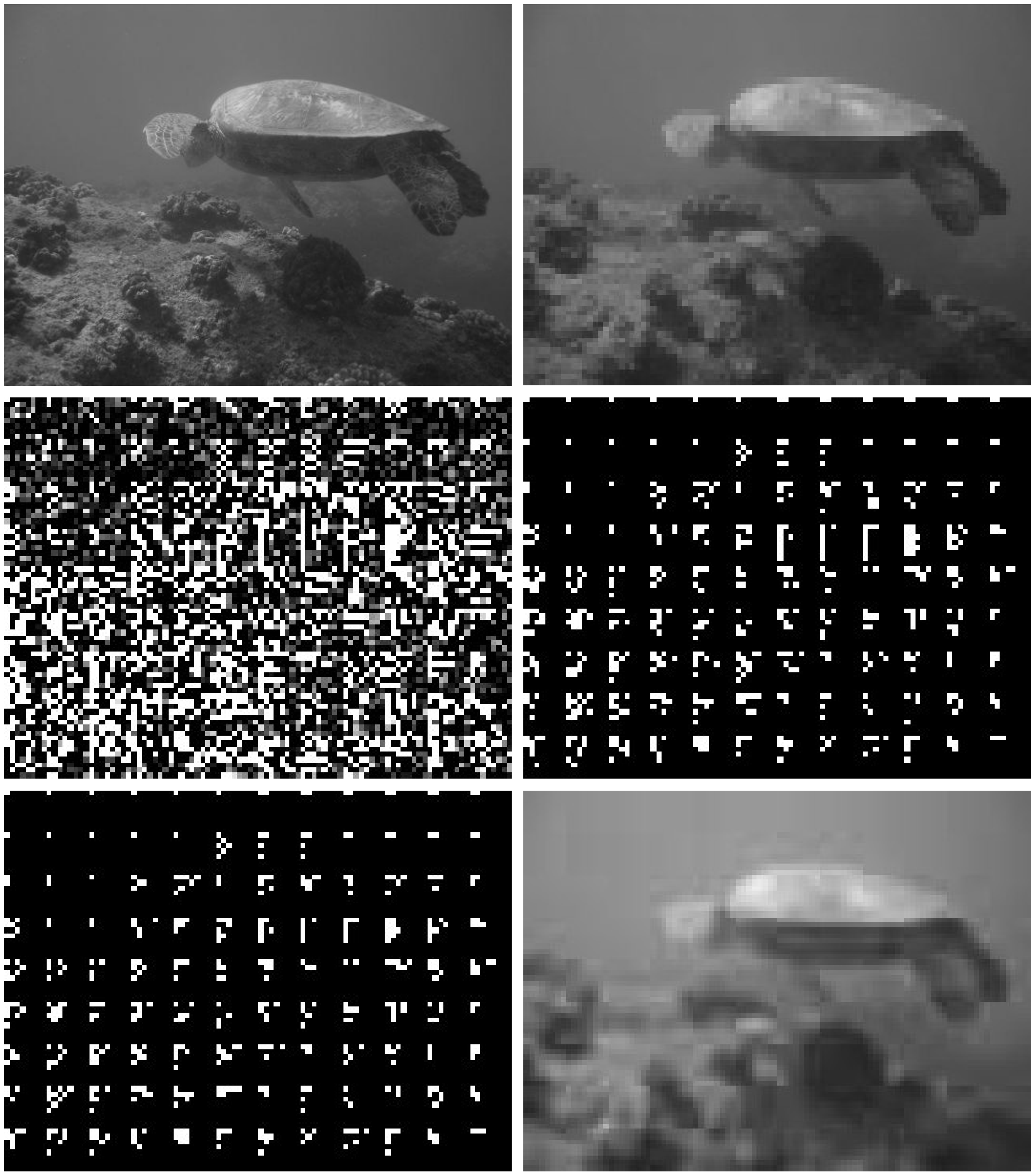


Figure 7. Encoding an image for acoustic transmission.

(a) Original image, (b) resized to 96x72, (c) 8x8 DCT2, (d) quantized DCT2,

(e) upper-left masked to 15 coefficients per block, (f) reconstructed image.

4 Acoustic Link Testing

The R-Modem based acoustic telemetry system is not yet ready for deployment in a true field experiment. The main restriction on field testing is the requirement to have both ends of the acoustic link (transmitter and receiver) connected to host PCs running MATLAB. Until we are satisfied with a 'final' working combination of physical layer parameters, we must keep the R-Modem nodes connected (via JTAG) to Code Composer Studio, Simulink and Real-Time Workshop for debugging. Without the JTAG connection we cannot observe the performance of the system (e.g. to observe decoded symbols on the output of the equalizer) or adjust the symbol rate, equalizer depth, packet size, etc. as needed. As a result, the R-Modems cannot go anywhere that a laptop PC cannot go; options for modifying the existing equipment into a system that could be deployed at sea are presented in Chapter 5. In principle, it is possible to develop R-Modem firmware that is remotely reconfigurable on-the-fly; for example, it might accept tuning parameters (packet size, even constellation size) over the serial interface. In practice, this sort of no-recompile parameter adjustment is difficult to build into a running Simulink model.

The acoustic experiments performed in this project were executed indoors, in two different environments, both of which provided the benefits of simplified logistics and protection from inclement weather. Our first set of experiments was conducted at the MIT Towing Tank; the second set in a small (30 gallon) bench-top aquarium. In the latter case, our goal (in part) was to create a portable test environment suitable for small-group demonstrations of the prototype hardware and of the overall acoustic video transmission concept.

4.1 Acoustic link parameter tuning

As described in Chapter 3, this project began with system integration: acquiring new hardware, writing new software to connect the hardware components, and initial experiments with encoding digital video frames into acoustic modem packets. Following this initial testing stage, our work proceeded to wet testing of the full system with all acoustic hardware in the loop. We set out to perform a coarse-grained parameter sweep of the acoustic link design space.

4.1.1 Initial goals for link tuning

We began this project with the intention of testing several different video encoding, modulation, and equalization schemes in a challenging underwater channel, in order to try and determine which combination would provide the best video quality. This portion of the project had two interrelated goals: design of the physical layer for maximum (reliable) throughput, and design of the input data stream to best take advantage of the available bandwidth, following a procedure generally known as cross-layer optimization. Such attempts at optimization require recursive adjustments of the physical layer and the application layer of the overall data link.

4.1.2 Tuning limitations

Application layer tuning consisted of video encoding choices on the digital camera, as described in Chapter 3. Tuning of the R-Modem physical layer proved to be more challenging. While attempting to increase throughput, we encountered many processing errors. Receiver processing time was originally predicted to increase gradually with increasing modulation complexity.

Ideally, the receiver would transition (with increasing work-load) from real-time processing, to near-real-time, to running well behind the transmitter – but still decoding packets correctly.

We attempted to change many parameters, one by one – increased packet length, larger symbol constellation (e.g. 8PSK), lower transmitter interpolation rate – and consistently caused the receiver to lock up, presumably due to overrun errors. As of this writing we have been unable to fully trace the root cause of these crashes. After a long process of trial-and-error, we have pushed the system to perform as well as it can, against limits which were much lower than we anticipated. The working Simulink model uses quadrature phase-shift keying (QPSK) modulation at a 12 kHz carrier. We use rate 1/3 convolutional coding for forward error correction. Each packet contains 100 pseudo-random equalizer training symbols (200 bits) followed by 200 data symbols (400 bits, or 50 bytes). Two data bytes are reserved for meta-data (parity or cyclic redundancy check, for example) but are not currently used. The transmitter DAC operates at a 48 kHz sample rate; transmitted symbols are interpolated at a rate of 48 to one.

At the receiver, signals are sampled at 48 kHz and digitally down-converted to baseband. Low-pass filtering, decimation, and preamble detection are followed by the RLS-DFE block described in Section 2.2.1. The equalizer output is fed into a hard-decision Viterbi decoder for convolutional code processing. The decoded bit-stream is converted to 8-bit characters, and printed to the serial port. Transmission of one 48-byte packet takes 0.65 seconds. The resulting packet data rate is approximately 600 bits per second, but aggregate throughput is considerably lower, as described in Section 4.3.

4.2 Towing Tank tests

Our first round of acoustic testing took place in the MIT Towing Tank. The Towing Tank is conveniently located on campus near the MITSG facilities, frees the user from depending on good weather for testing, and provides an extremely challenging acoustic environment. The Towing Tank is a long, narrow corridor of water with rigid bottom and sides, 108 feet long, 8.5 feet wide, and 4.5 feet deep. Directional (30 degree beam) R-Modem transducers were placed at mid-depth in the tank, facing each other, for horizontal transmission of sound. It was our hope that, with an equalizer of adequate sophistication, it would be possible to compensate for the severe multipath effects of this acoustic channel and transmit high-quality data at moderate bit rates. Given a successful demonstration of moderate to high bit rate communications in this difficult environment, it might be fair to assume that a system which worked in the Towing Tank would work just as well in a wide variety of real subsea conditions.

Our directional transducers were mounted face to face at approximately two feet underwater, one fixed to the mobile towing carriage of the Towing Tank, the other fixed to a rolling cart that spans the width of the tank. In this way, the transmission range could be varied at will. Acoustic link quality was primarily tested using a simple numbered packet scheme, repeated transmission of the character string “MIT Sea Grant Test #N” (where N = packet number), padded out to 48 bytes with asterisks ('*').

It quickly became clear that link quality was very sensitive to the horizontal position of the transducers within the narrow corridor of the tank. Swinging the transducer mounting arm ten degrees to either side made the difference between 90% packet reception and 90% packet loss. Each time we changed the distance between the two transducers, it was necessary to readjust the horizontal position of the transmitter and receiver as well. This effect is likely due to a standing-wave interference pattern created by reflections from the walls, floor, and free surface of the narrow (approximately 20λ) tank. This type of multipath interference can also appear in the field, especially in horizontal channels. In future work, problems with transducer placement sensitivity will be readily solvable by the use of a significantly higher carrier frequency (making it less likely that the transducer will end up acoustically isolated in a local null), and of a multichannel receiver (to take advantage of spatial diversity) [25].

While the USB-JTAG connection for each R-Modem DSK board made it easy to recompile and download new DSP code as needed, we were less successful in using it for real-time observation of the detection, equalization and decoding process on the receiver node. The limited data throughput available via RTDX (see Section 3.2.2) meant that, for example, we were unable to observe the output of the preamble detector (correlator) at full sample rate. The preamble correlator feeds a peak detector with binary output (packet detected / not detected); after decimating and truncating the output data for plotting (in Code Composer Studio) on our host PC, it was difficult to assess peak detector performance with any precision. We found that it was possible to configure the firmware model for reliable connection (via RTDX) to a Simulink Scope block – presenting us with a scatter plot of QPSK symbols, post-equalization – at the expense of a factor-of-2 increase in packet decoding time.

4.3 Small-scale tests

Later in the fall term, the Towing Tank was unavailable for our work, as it was increasingly occupied by other experimental studies. To support further testing, we assembled a bench-scale facility for acoustic transmission experiments. A thirty-gallon aquarium (1 x 1 x 4 feet) was lined with half-inch sheets of soft, heavy neoprene rubber (Shore 10A hardness, 96 lb/ft^3). This lining was observed to reduce the number of repeat detections of individual short (0.65 sec) packets from a typical value of five, down to one or two. Our directional acoustic transducers were mounted at six inches depth on opposite ends of the tank. We installed the tank in the base of a mobile cart, with the transmit and receive R-Modems and their respective host PCs on the top shelf. While the small scale of this test environment makes it an unrealistic model for the channel effects of the open ocean, it is perfectly adequate to perform full system check-out tests, with real acoustic propagation in the loop.

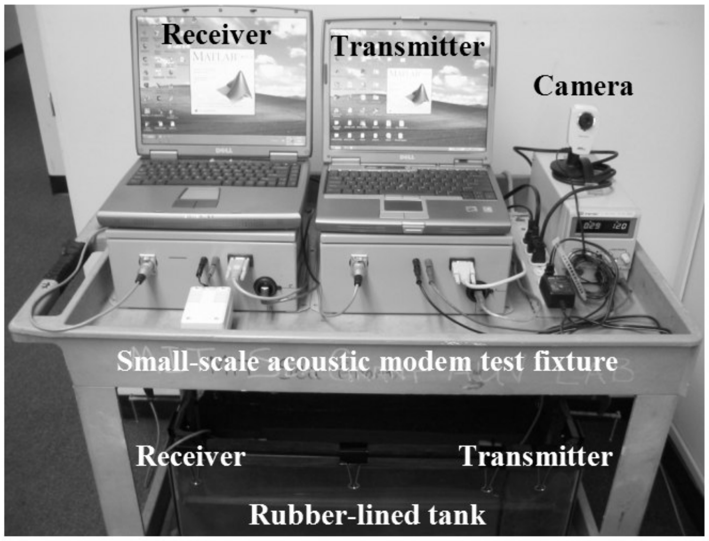


Figure 8. Acoustic modem testing in 30 gallon tank.

We began a second round of experiments. While adjusting the transmitter delay between packets, we discovered a flaw in the receiver firmware. As built, the receiver Simulink model appears to stop decoding the most recent received packet when a new preamble is detected. If the transmitter is configured to pause for the full packet processing delay (nearly five seconds) required by the receiver, packets nearly always come through uncorrupted (95% success rate). As this delay is reduced, more and more of each packet (starting with the last few symbols and working backward) is consistently corrupted, as if the decoding process is either terminating prematurely, or is interfered with by the parallel process of detecting and buffering a new packet.

For the purposes of full system demonstration, we chose to simply delay each packet by the required processing latency. Even though packets are transmitted at a symbol rate equivalent to 600 bps, the receiver processing delay currently reduces our aggregate throughput to approximately 80 bps. A 96x72 test image (as described in Section 3.3), broken into 36 packets, takes three minutes to transmit. Despite waiting five seconds between transmissions, we still experience intermittent corruption at the end of three-block packets.

In future experiments, an improved version of the R-Modem firmware will enable us to use much longer packets. Extending the length of a packet well beyond the reverberation time of the channel will eliminate the problem of repeated preamble detection; at the moment, our packets are significantly shorter than the total reverberation time of the 30 gallon tank. In open water testing (see Chapter 5), reflecting surfaces will be much farther apart, which means we will have fewer echoes that are more heavily attenuated. Longer packets will be necessary in any case, as part of our effort to achieve higher throughput – more data per packet means less time dedicated to overhead (preamble, training sequence) on average.

5 Conclusions

5.1 Project accomplishments

This project began with a concept of operations, and a largely untested set of hardware and software. We have produced a reliable demonstration prototype of an acoustic video transmission system. The system includes a working near-real-time implementation of a 12 kHz (carrier frequency) QPSK receiver, based on a decision-feedback equalizer with integral PLL, that is entirely built in MATLAB and Simulink. We have laid the groundwork for AUV integration of acoustic video telemetry in the next phase of this research program, by demonstrating continuous live acquisition of digital video frames and streaming image transmission through the underwater channel. Finally, this project has served as a successful proof-of-concept for student involvement in the development of our software-defined acoustic communications platform, beginning with introductory training in Simulink and Real-Time Workshop, and ending with management of the entire R-Modem project handed off to the author prior to full system demonstration.

5.2 Recommendations for future work

5.2.1 Oilfield integration

In order to design a reliable communications system for oilfield use, we must accurately characterize the background acoustic environment of oilfield operations. In the next phase of this project, we plan to perform a thorough literature search and follow-up analysis on the subject of acoustic noise in the oilfield. To supplement this research, we plan to make our own broadband recordings of oilfield ambient noise using a rugged, deep-rated digital data acquisition system. Using this information, we will design communication signals and processing algorithms for operation in the oilfield acoustic environment.

In future versions of the R-Modem, we expect to achieve significantly higher throughput on the acoustic link (bit rate improvements of 10-100x). This still represents a significant constraint, as compared to cabled data transport. In the final implementation, we may expect to adjust trade-offs between video frame rate and image quality on demand. For example, during a routine inspection, the AUV supervisor might request high-resolution snapshots of problem areas, interspersed with low-resolution, high-speed streaming video.

In Section 2.2.2 we described the need for networked communications in the oilfield environment. Sections 5.2.4 and 5.2.5 address the evolution of our acoustic communications platform from a basic point-to-point transmitter (or receiver) into a multipoint-capable network node.

5.2.2 Signal design

In order to increase the throughput of our acoustic link to the point where we can achieve true video-rate image transmission, we must attack the problem on several fronts. Putting aside (for the moment) the problem of processing speed, we may consider proposed changes at the physical layer.

Building on the results of [16], higher-order constellations (more bits per symbol) are the primary tool envisioned for increasing throughput. Increasing the constellation size while keeping the symbol rate relatively low will make it easier for the equalizer to compensate for inter-symbol interference. Hardware upgrades pending, it will also be possible to test higher carrier frequencies. The R-Modem prototype is limited to a 12 kHz carrier frequency and 2 kHz of bandwidth. As a next step, moving to (e.g.) 24 kHz center frequency and 4 kHz of bandwidth will make it easier to increase the number of bits per symbol transmitted. The non-linear increase in attenuation at higher carriers will provide an additional benefit by decreasing the interfering effects of multipath reception.

Early results are promising for a different physical layer design altogether [25]. Orthogonal frequency-division multiplexing (OFDM) is a technique in widespread use for high-speed data transmission in complex channels; well-known applications include Digital Subscriber Line (DSL) Internet service, 802.11g wireless radio networking, and broadcast high-definition television (HDTV). By dividing a broad transmission band into many orthogonal sub-carriers, many data bits may be transmitted in parallel while holding each sub-carrier to a low symbol rate. For example, BPSK modulation over each of 1024 tightly space sub-carriers could provide kilobit-per-second throughput at only one symbol per second, effectively masking out inter-symbol interference entirely.

Packing so many modulated sub-carriers into a single signal requires a combination of wider transmit/receive bandwidth and careful management of sub-carrier spacing. We have successfully tested broadband omni-directional transducers (model #1042) from ITC with our R-Modem hardware. These transducers have over 60 kHz of usable bandwidth (~15-75 kHz range). Taking full advantage of their capabilities will require increasing our transmitter DAC sample rate, as well as the operating bandwidth of our transmit power amplifier.

In the underwater channel, an OFDM-based physical layer design will be particularly susceptible to Doppler distortion. At separations of only tens to hundreds of Hz, continuous, precise Doppler tracking and compensation will be required in order to keep sub-carriers distinct and OFDM symbols intelligible. Nonetheless, we anticipate a significant overall reduction in computational complexity at the receiver – the core OFDM demodulator consists of a simple FFT – and a corresponding leap forward in real-time performance.

5.2.3 Video encoding

The first step in improving video transmission – as opposed to simply improving throughput of arbitrary data – is to go beyond our simple adaptation of the JPEG algorithm and to experiment with other encodings for single frames. After considering JPEG2000, and other, more domain-specific image encoding methods [13], we have settled on the ICER format [26] as the best candidate for improved single-frame encoding. ICER is another wavelet-based method, originally developed by NASA-JPL, and currently used for image telemetry from the Mars Exploration Rovers (MER). ICER seems particularly well suited to our application because it is designed to accommodate error-prone wireless connections, with automatic “image context error correction” for recovery from single bit errors mid-frame. In the MER program, ICER has been used to compress large gray scale images to an average of 1.13 bits per pixel, and small (64x64) thumbnails to 1.27 bits per pixel. For comparison, our current low-fidelity monochrome image encoding averages 1.87 bits per pixel.

We expect to continue using single-frame encoding until physical layer performance has improved by approximately two orders of magnitude. Empirical results suggest that 5 kbps is the practical minimum for transmission of an intelligible (very low-resolution) H.264-encoded video stream. Tests with the Reef Explorer AUV (see Section 3.1) have shown that a one-Hz frame rate is adequate for very low-speed inspection, while a true 'video' user experience requires at least ten frames per second. As higher bit-rates become practical, this project may make use of a variety of free implementations of modern, high efficiency streaming video codecs [27].

The integration of the wireless video system into a sophisticated autonomous vehicle presents yet another possibility. As AUV technology continues to improve, we may expect computer-aided detection and classification (CAD/CAC) methods originally developed for military applications (mine countermeasures) to be applied to equipment inspection. An embedded system dedicated to riser inspection, for example, might be designed to fuse high-resolution video and sonar data and produce a sparse three-dimensional model of its immediate environment. A system sophisticated enough to detect and classify flaws (e.g. cracks in pipes) could present the AUV supervisor with a simple 3-D diagram of the inspection target, reserving bandwidth-intensive high resolution images for automatically identified points of failure.

5.2.4 The R-Modem platform

The next phase of this project must address a number of flaws in our prototype acoustic communications platform: DSP overruns, power amplifier bugs, lack of network- and application-layer software, and close coupling with a host PC programming environment. In this section we consider these flaws and suggest ways to eliminate them.

DSP overruns

One of the challenges of developing a high-speed signal processing system is the management of multiple concurrent computing tasks on the DSP processor. While DSPs are optimized for real-time manipulation of streams of data, the receiver in our application must do a combination of stream-like processing (down-conversion, decimation, preamble detection, decoding) and block processing (packet-level operations). When the beginning of a packet is detected, it must be buffered, then demodulated and decoded in the background while a second software component (task) listens for new packets. As the DSP is the only processor on the DSK6713, it must also handle higher-level tasks. At the moment, this consists only of communication with the topside computer via RS-232, while future experiments are planned which will also require network-layer (MAC and routing) operations.

Texas Instruments provides a built-in facility for multitasking via the lightweight DSP-BIOS real time operating system. The multitasking behavior of DSP-BIOS may be tuned to accommodate tasks that must run at different rates and with different input/output conditions, but this is difficult to do from MATLAB. Real-Time Workshop was originally designed for hardware-in-the-loop simulation of physical systems, for control theory applications, and for sampling rates no higher than a few kilohertz.

Simulink blocks from MATLAB's built-in library are designed for rapid prototyping on a PC and are not optimized for embedded processor architectures. For a number of signal processing applications, TI provides optimized blocks specifically developed for their DSPs; some engineering effort dedicated to converting the R-Modem Simulink model to use a greater proportion of TI-optimized blocks may be worthwhile. It has been our experience that simply requesting that the Real-Time Workshop compiler optimize its generated code, either on a block level or on a full-model level, consistently results in code that crashes. It is possible that the attendant code generation facilities have only been thoroughly tested for control applications and are not designed to produce tightly optimized code suitable for a real-time communications transceiver. Perhaps it is simply difficult to generate code that can gracefully share processor time between direct-conversion sampling (48 ksps and above) and baseband-rate equalization and decoding (closer to 1 ksps).

In an effort to reduce the real-time workload of the DSP, MITSG has developed a custom hardware alternative to the DSK6713-based R-Modem. The R-Modem custom board has faster ADCs and DACs (240 ksps, 16 bit, four channels each), active filters for finer control of anti-aliasing, more RAM, more Flash, and a variety of other feature upgrades as compared to the DSK6713. The primary difference, however, is that the custom R-Modem board is designed around a large programmable logic device, an Altera Cyclone FPGA. The Cyclone serves as a reconfigurable interconnect between all other devices on the R-Modem custom board, controlling the flow of data between ADCs, DACs, RAM, Flash, digital I/O, RS-232 UART, and the DSP.

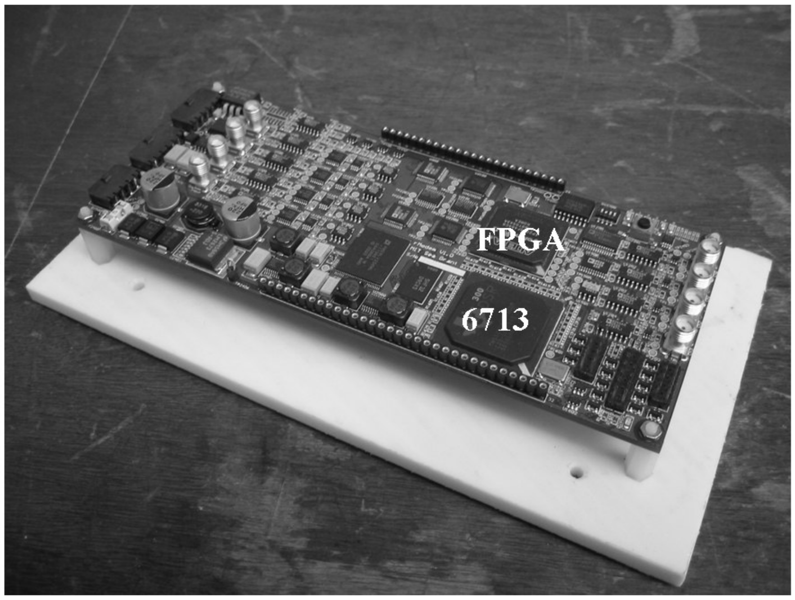


Figure 10. R-Modem custom board.

This sort of flexibility should make it possible to offload several important signal processing tasks from the DSP to the FPGA. First, we plan to use the FPGA for digital down-conversion and up-conversion (DDC/DUC) as well as decimation and low-pass filtering. These few changes will reduce the DSP workload enormously, as the 6713 will then be free to operate exclusively at baseband. Later revisions to the firmware design may include FPGA implementation of further processing stages, such as the preamble detector (correlator).

The R-Modem custom board is not yet ready for integration with the rest of our wireless video system. The down-side of using an FPGA as master device is our limited ability to make use of the hardware without complete, debugged VHDL code. While the basic logic necessary to pass data from ADC to RAM to DSP (for example) is complete, the existing direct-conversion front end implementation does not work correctly. The custom board requires a significant investment of engineering time in order to reach a fully usable state. We may be able to continue with our current model for rapid development by incorporating Altera's DSP Builder software tool into the existing R-Modem Simulink model. DSP Builder is a code generation 'bridge' similar to Real-Time Workshop's Link for Code Composer Studio.

Power amplifier

The analog front end design of the R-Modem, while adequate for bench-top experiments, needs further revision before the next stage of our work. The power amplifier and preamplifier might be improved in several ways. The Class B power stage is linear, but inefficient. Large passive convection heat sinks were barely adequate to keep the power MOSFETs cool during bench testing and would not perform as well in a partially-evacuated underwater housing. There is no means of software control over transmit power, aside from simple scaling down of the DAC output; the dynamic range achievable in this way is limited, and the resolution at low DAC voltages quickly becomes very poor. In field testing it is both impractical and imprecise to rely on a rotary potentiometer 'volume knob' at the power amplifier output. Transmit/receive switching via electromechanical relay is slow and noise-prone. The hand-wound output transformer is difficult to manufacture, and seem to have a high leakage inductance which further interferes with switching from receive to transmit mode. The fixed-gain preamplifier has worked well at our 12 kHz carrier but has not been characterized for broadband operation, and cannot be used to implement automatic gain control (AGC).

Many of these problems might be solved by the design of a new analog front end daughterboard for the DSK R-Modem and/or the R-Modem custom board. We propose a transformerless switching power amplifier design. The new power amplifier would combine a digitally controlled fixed-frequency boost converter first stage with an analog-input Class D H-bridge second stage. The first stage would be used to control transmit power by varying the supply voltage for the Class D bridge. The Class D, operating directly at the transducer drive voltage (tens to hundreds of volts, depending on the output power setting) would operate at a fixed frequency of 1 MHz, driven at varying pulse-width modulation (PWM) duty cycles by a high-speed comparator at the analog input. The design concept is based on high-efficiency industrial motor drive circuits. The Class D output may or may not require a filter inductor to reduce switching harmonics, or to better match the piezoelectric transducer load.

At the output of the Class D switching amplifier – that is, at the transducer port – we would connect a passive transmit/receive (T/R) switching network (snubbing diodes and resistors). The passive T/R network would allow faster mode switching, and might also improve reliability as compared to the electromechanical relay. On the other side of the T/R switch we would connect a wide-band programmable gain amplifier to allow software control of receiver gain (either on-demand or through an AGC servo loop).

We have experimented with off-the-shelf amplifier components as a temporary alternative to the custom-built analog front end. Power amplifiers and preamplifiers are commonly available in the audio frequency band only (20 Hz to 20 kHz), so this approach cannot be used for high carrier frequencies. That said, we have had some success with a combination of a 12VDC powered Class D audio amplifier, a 4:1 step-up transformer (designed for high-voltage audio distribution in public-address systems), and a 12VDC powered studio-quality preamplifier for dynamic microphones (up to 50dB manually adjustable gain). Part numbers and ordering information are available in Appendix 6.1.

Networking and field testing

As built, the R-Modem has a simple interface (described in Section 3.2.4) and very little in the way of internal state. On the receiver, packets are simply buffered, decoded, and sent to the serial port, 48 bytes at a time. This is adequate for a one-way point to point link, though even in this simple case the application layer would benefit from some additional structure: special characters to indicate the end of a packet, or messages back to the host that indicate a corrupted packet (bad cyclic redundancy check). In a multi-point network, the R-Modem will need to deal with meta-data as well: source and destination addresses, message priority, multi-hop routing. The modem will need a medium access control (MAC) layer, which means it will need to detect collisions, repeat failed transmissions, and handle request-to-send / clear-to-send (RTS / CTS) handshakes. All this implies a fairly sophisticated packet-based state machine running on top of the basic physical layer model.

While it is entirely possible to build network-layer software in MATLAB and Simulink, it may prove difficult to integrate the network layer into the existing Simulink model. We have already encountered difficulties in programming the DSP for multiple tasks that run at very different rates. One possible solution to this problem is to split the design, leaving the physical-layer to the R-Modem hardware (DSP and/or FPGA) and passing packet data and diagnostic information to MAC and routing layers running on the host PC processor. These layers might be implemented in MATLAB or in whatever development environment is most suitable. MITSG is currently developing an acoustic network simulator in Python; it may be possible to re-use some simulator MAC and routing code in a Python-based off-board network layer for the R-Modem.

The host processor cannot remain a full-sized laptop PC, however. Future experiments will demand deployment of R-Modem nodes in the field – not just at the Towing Tank. Two or more R-Modems installed in simple buoys, anchored near shore in shallow water, would provide a much more realistic test environment for our signal processing algorithms. With the R-Modems beyond the reach of USB cables, however, it will be impossible to debug running code or to edit and recompile our Simulink models. Installing laptops in buoys presents another problem altogether. We propose adding an embedded PC-compatible host processor running MATLAB to the standard R-Modem hardware installation. The embedded PC will provide a

USB JTAG connection, and a full remote-access development environment (through e.g. VNC, Remote Desktop, or pcAnywhere) so that we may use MATLAB, Simulink and Code Composer from shore, via wireless Ethernet.

In our robotics research at MITSG, we use a similar approach for developing AUV control software. The AUV navigation computer runs Ubuntu Linux, rather than Microsoft Windows, but otherwise the concept is the same: we use a wireless Ethernet connection to change tuning parameters, edit mission scripts, and start experiments (dives). When the AUV comes to the surface, the experiment is complete, and we may download and examine the data it has recorded. It is easy to edit and recompile the AUV control software (C++ in this case, rather than Simulink models) over the wireless connection. A similar approach for R-Modem deployments would help us overcome some of the obstacles to further refinement of our design.

5.2.5 Alternative platforms

As we have seen in the previous section, despite the successes of this project, there is clearly room for improvement in the underlying platform. In addition to the changes suggested above, we have investigated several alternative platforms for software-defined acoustic modem experiments.

Simulink Real-Time Windows Target

The first idea that seemed promising was to eliminate the DSP from the system entirely, and build a new R-Modem based exclusively on a general-purpose PC processor. While this approach would significantly increase power consumption of the R-Modem hardware, it would eliminate the additional layer of complexity introduced by generating DSP-targeted code. The Simulink code generator and compiler includes a Real-Time Windows Target in addition to its various embedded processor targets (DSPs, microcontrollers). We planned to convert the R-Modem Simulink model to run under Real-Time Windows Target (RTWT), using a PC-based data acquisition peripheral (National Instruments DAQCard) for analog I/O. With DAQCard drivers built into Simulink, and off-the-shelf power amplifier and preamplifier already available (see Section 5.2.4, Power Amplifier), we expected to make progress quickly. Simulink, however, and RTWT in particular, are primarily designed for control applications – motion control, temperature control – and the RTWT manual states that real-time models may run at sample rates "as fast as 5 kHz", which is insufficient for our purposes.

GNURadio and the Universal Software Radio Peripheral

GNURadio is a free toolkit for experiments in software-defined radio (SDR). It is an open-source project, distributed under the GNU General Public License and developed by a worldwide community of volunteers and professionals. GNURadio is designed to use a general-purpose PC processor for digital signal processing; PCs running GNURadio can be turned into flexible transmitters or receivers using any type of modulation or data encoding. It is a cross-platform toolkit and is actively developed for Linux, Windows and Mac OS. A GNURadio application is built by connecting modular components (blocks, written in C++) into a 'flow graph', using a simple Python scripting syntax. The GNURadio distribution provides more than 100 modular blocks which implement everything from simple 'add' and 'multiply' functions to a complete broadcast HDTV receiver. GNURadio has been under development since 1998; in that time, it has grown from a hobby-scale project to a professional-quality SDR framework, in active use at several MIT laboratories and at many of our peer institutions. Just in the last year, the GNURadio development community has begun a collaborative effort to build a set of general-purpose blocks for OFDM modulation and demodulation.

GNURadio flow graphs may be directly connected to analog I/O hardware (such as a DAQCard or PC sound card) for direct digital down- and up-conversion. For radio applications, however, direct conversion from a megasample ADC input is very CPU intensive. The GNURadio project has a companion open-source hardware effort that is very similar, in concept, to the role envisioned for the FPGA portion of the R-Modem custom board. The Universal Software Radio Peripheral (USRP) is a small, reconfigurable USB peripheral based on an Altera

Cyclone FPGA. (The next version of the USRP, due to be released in the second quarter of 2008, will have a gigabit Ethernet interface in place of the USB 2.0 connection). The USRP is based on an Analog Devices AD9862 mixed-signal front end, which provides a general-purpose ADC and DAC interface to up to four daughterboards. USRP daughterboards are available for a variety of operating bands, including a low frequency set (LFTX, LFRX) suitable for the acoustic communication frequency range. The Cyclone FPGA is dedicated to daughterboard interface management, USB data transport, digital down- and up-conversion, decimation and interpolation, and nothing else. The VHDL model for the USRP is tested, debugged, and freely modifiable by any GNURadio user.

In the next version of the R-Modem, GNURadio could replace Simulink as the modular foundation of our signal chain. The GNURadio project's signal processing modules are not just for simulation - they have been tested in real-world SDR systems and are “ready to go” for our application. Running the core signal processing code on an embedded PC, rather than a DSP, should make debugging easier: we would have access to familiar tools (development environments, debuggers) and could take advantage of MITSG's in-house expertise in C++ and Python software design.

A PC-based R-Modem would have effectively infinite storage (disk and RAM), as compared with the DSP boards; the new R-Modem would be able to keep detailed logs of its internal state, making it easy to record estimated channel impulse response functions, or equalizer performance metrics, for later review. We can also take advantage of the Python-based GNURadio graphical interface, GNURadio Companion, to display live “virtual oscilloscope” plots from many points in the signal chain simultaneously. Our approach to remote control and supervision of the GNURadio R-Modem in field experiments would be identical to that described for AUV software development in Section 5.2.4.

GNURadio is a soft-real-time architecture. We cannot be sure, at this time, whether a GNURadio implementation of the R-Modem concept would be able to 'keep up' with high-speed acoustic data transport. However, by including the USRP in our redesigned R-Modem, we can allow the PC processor to operate exclusively at baseband. Small-form factor embedded PCs are now available that can attain gigaFLOP performance using the GNURadio libraries, so it seems likely that the system as a whole would be fast enough – and easily upgradeable in the future.

Much like Simulink, GNURadio is designed primarily as a streaming framework. In the current release (3.0), users must develop their own layer of software to manage packet-level operations. Two benefits are seen in this case: PC operating systems are already designed to accommodate multitasking at different rates and priorities; MAC and routing layers developed in Python will be easy to integrate with GNURadio flow graphs. At the application layer, our new R-Modem framework would have direct access to Python libraries (and PC hardware) for serial and Ethernet data transport, without needing to build this type of infrastructure into the real-time model (as in Simulink). Plans for GNURadio release version 3.2 include a new set of tools for packet-level data management, currently known in the developer community as message blocks or 'mblocks'. Development of the GNURadio packet architecture is led by network engineers at BBN Technologies.

6 Appendices

6.1 Part numbers and purchasing information

Spectrum Digital #701895 DSP Starter Kit (DSK) for the TMS320C6713
12502 Exchange Dr., #440
Stafford, TX. 77477
T: (281) 494-4500 x 113
F: (281) 494-5310
http://www.spectrumdigital.com/product_info.php?cPath=31_75&products_id=113

Link Research #DSPG2-IC2-U232 dual RS-232 serial port daughtercard
131 Fairview Street
Providence, RI 02908
Tel: 401-270-4445
FAX: 401-270-5221
www.link-research.com

Axis Communications #207 network camera
100 Apollo Drive
Chelmsford, MA 01824
Tel: +1 978 614 2000
Fax: +1 978 614 2100
http://www.axis.com/products/cam_207/index.htm

Poly-Planar #MZ-100 15W +12VDC Marine Audio Amplifier
http://www.polyplanar.com/productSingle.aspx?prt=MZ-100
Purchased from Crutchfield Corporation, Item #693MZ100
1 Crutchfield Park
Charlottesville, VA 22911-9097
1-888-955-6000
http://www.crutchfield.com/S-fjamYqUfhDl/App/Product/Item/Main.aspx?i=693MZ100

SM Pro Audio XPM1 +12VDC 'Nano' series microphone preamplifier
Purchased from B&H Photo-Video, Item #SMXPM1
420 9th Avenue
New York, N.Y. 10001
800.606.6969
http://www.bhphotovideo.com/c/product/456813-REG/SM_Pro_Audio_XPM1_XPM1_Microphone_Preamp.html

Our Simulink model was developed under MathWorks' MATLAB version R2007a.
No other version is guaranteed to work, including R2007a+.
DSK6713 programmed with Spectrum Digital-bundled Code Composer Studio 3.1.

Recommended MathWorks training courses:

Simulink for Signal Processing
http://www.mathworks.com/services/training/schedule.html?courseNum=SG02

Real-Time Workshop Fundamentals
http://www.mathworks.com/services/training/schedule.html?courseNum=RT01

Real-Time Workshop Code Generation
http://www.mathworks.com/services/training/courses/RT02_1.html

Acoustic absorbing tank lining:
McMaster-Carr neoprene sheets #9109K25, Shore 10A, 96 pounds per cubic foot

6.2 MATLAB code for image transmission

Transmitter script

```
%%

TRPORT = 'COM4';
s1 = serial(TRPORT, 'BaudRate', 115200);
fopen(s1);

while(1)

A = imread('http://auv1.mit.edu/axis-cgi/jpg/image.cgi?
resolution=160x120&colorlevel=0&compression=50');

blocksize = 8;
blocksh = 12;
blocksw = 16;
height = blocksh*blocksize; width = blocksw*blocksize;
A = imresize(A(:,:,1),[height width]);
C = imresize(A, [height/2 width/2]);
C = imresize(C, [height width]);


blocklayout = ones(blocksh,blocksw);
blockidx = zeros(numel(blocklayout),2);
[blockidx(:,1) blockidx(:,2)] = find(blocklayout);

% quantization matrix
qtz =      [16 11 10 16 24 40 51 61;
            12 12 14 19 26 58 60 55;
            14 13 16 24 40 57 69 56;
            14 17 22 29 51 87 80 62;
            18 22 37 56 68 109 103 77;
            24 35 55 64 81 104 113 92;
            49 64 78 87 103 121 120 101;
            72 92 95 98 112 100 103 99];

txblock = zeros(blocksize);
rxblock = zeros(blocksize);
N = 5;
mask = zeros(blocksize);
mask(1:N,1:N) = fliplr(triu(ones(N)));
[I,J] = find(mask);
% weird little sorting trick using complex values
M = I + sqrt(-1)*J;
M = sort(M);
I = real(M);
J = imag(M);
numsamples = nnz(mask);
```

```
blocksinpkt = 3;

txbuf = zeros(1 + numsamples*blocksinpkt,1);
rxbuf = zeros(1 + numsamples*blocksinpkt,1);

B = zeros(size(A));

txblockctr = 0;
txpktnum = 0;
rxblocknum = 1;
rxpktnum = 0;

for k = 1:length(blockidx)
    i = blockidx(k,1);
    j = blockidx(k,2);

    txblock = A((i-1)*blocksize+1:i*blocksize,(j-1)*blocksize+1:j*blocksize);
      txblock = dct2(txblock);
    txblock = round(txblock ./ qtz);

    for coeffidx = 1:numsamples
        txbuf(1 + coeffidx + (txblockctr*numsamples)) = txblock(I(coeffidx),
J(coeffidx));
    end

    txblockctr = txblockctr + 1;

    if (txblockctr == blocksinpkt)
        txpktnum = txpktnum + 1;
        txbuf(1) = txpktnum;
        txblockctr = 0;
        % transmit data
        tempbuf = zeros(1,48);
        tempbuf(1:length(txbuf)) = txbuf;
        fwrite(s1, tempbuf, 'int8');

        % simulate reception
        rxblock = zeros(blocksize);
        rxbuf = txbuf;
        %rxbuf = round(rxbuf.*(1+randn(size(rxbuf))/10));
        rxpktnum = rxbuf(1);
        for rxblockctr = 0:(blocksinpkt-1)
            rxblocknum = (rxpktnum-1)*blocksinpkt + rxblockctr + 1;
        for coeffidx = 1:numsamples
            rxblock(I(coeffidx),J(coeffidx)) = rxbuf(1 + coeffidx +
(rxblockctr*numsamples));
        end
        rxblock = rxblock .* qtz;
        voffset = (blockidx(rxblocknum,1) - 1)*blocksize + 1;
        hoffset = (blockidx(rxblocknum,2) - 1)*blocksize + 1;
        B(voffset:voffset+blocksize-1, hoffset:hoffset+blocksize-1) =
idct2(rxblock);
        end
        figure(1)
        subplot(2,1,1);
        imshow(A,'InitialMagnification','fit');
        title('Image to be transmitted');
        subplot(2,1,2);
        imshow(uint8(B),'InitialMagnification','fit');
        title('Blocks transmitted so far');
        xl = sprintf('packet #%d',txpktnum);
        xlabel(xl);

        pause(5);
end
end

end

%%
stopasync(s1);
fclose(s1);
delete(s1);
clear s1;
```

Receiver script

```
%%

blocksize = 8;
blocksh = 12;
blocksw = 16;
height = blocksh*blocksize; width = blocksw*blocksize;

RXPORT = 'COM1';
s2 = serial(RXPORT, 'BaudRate', 115200, 'Timeout', 1);
fopen(s2);

blocklayout = ones(blocksh,blocksw);
blockidx = zeros(numel(blocklayout),2);
[blockidx(:,1) blockidx(:,2)] = find(blocklayout);

% quantization matrix
qtz =      [16 11 10 16 24 40 51 61;
            12 12 14 19 26 58 60 55;
            14 13 16 24 40 57 69 56;
            14 17 22 29 51 87 80 62;
            18 22 37 56 68 109 103 77;
            24 35 55 64 81 104 113 92;
            49 64 78 87 103 121 120 101;
            72 92 95 98 112 100 103 99];

txblock = zeros(blocksize);
rxblock = zeros(blocksize);
N = 5;
mask = zeros(blocksize);
mask(1:N,1:N) = fliplr(triu(ones(N)));
[I,J] = find(mask);
% weird little sorting trick using complex values
M = I + sqrt(-1)*J;
M = sort(M);
I = real(M);
J = imag(M);
numsamples = nnz(mask);

blocksinpkt = 3;

rxbuf = zeros(1 + numsamples*blocksinpkt,1);

B = zeros(height,width);
C = zeros(height,width);

rxblocknum = 1;
rxpktnum = 0;

        figure(1)
        subplot(2,1,1)
        imshow(uint8(B),'InitialMagnification','fit');
        s = sprintf('Current image, received packet #%d',rxpktnum);
        title(s);
        subplot(2,1,2)
        imshow(uint8(C),'InitialMagnification','fit');
        title('Previous (completed) image');

while(1)
        % receive a packet
        [rxbuf, count] = fread(s2,48,'int8');
        if (count>0)

        rxblock = zeros(blocksize);

        rxpktnum = rxbuf(1);
        if ((rxpktnum < 1) || (rxpktnum > (blocksh * blocksw / blocksinpkt)))
            continue;
        end
        for rxblockctr = 0:(blocksinpkt-1)
            rxblocknum = (rxpktnum-1)*blocksinpkt + rxblockctr + 1;
        for coeffidx = 1:numsamples
            rxblock(I(coeffidx),J(coeffidx)) = rxbuf(1 + coeffidx +
(rxblockctr*numsamples));
        end

        rxblock = rxblock .* qtz;
        voffset = (blockidx(rxblocknum,1) - 1)*blocksize + 1;
        hoffset = (blockidx(rxblocknum,2) - 1)*blocksize + 1;
        B(voffset:voffset+blocksize-1, hoffset:hoffset+blocksize-1) =
idct2(rxblock);
        end
        end

        if (rxpktnum == (blocksh * blocksw / blocksinpkt))
            C = B;
            B = zeros(height,width);
            rxpktnum = 0;
        end

        figure(1)
        subplot(2,1,1)
        imshow(uint8(B),'InitialMagnification','fit');
        s = sprintf('Current image, received packet #%d',rxpktnum);
        title(s);
        subplot(2,1,2)
        imshow(uint8(C),'InitialMagnification','fit');
        title('Previous (completed) image');

end

%%
stopasync(s2);
fclose(s2);
delete(s2);
clear s2;
```

7 References

[1] Sheridan, T.B. “Space teleoperation through time delay: review and prognosis”; IEEE Transactions on Robotics and Automation, Volume 9, Issue 5, Oct. 1993 Page(s):592 - 606

[2] Sayers, C.P.; Paul, R.P.; Whitcomb, L.L.; Yoerger, D.R.; “Teleprogramming for subsea teleoperation using acoustic communication.” IEEE Journal of Oceanic Engineering 23:1, Jan. 1998 pgs:60 - 71

[3] Chitwood, J. of Chevron Corporation. Personal communication, September 5th 2007.

[4] Bellingham, J. G. and T. R. Consi. “State configured layered control,” in Proc. IARP 1st Workshop on Mobile Robots for Subsea Environments, Monterey, CA, Oct. 1990, pp. 75–80.

[5] Vaganay, J.; Elkins, M.L.; Willcox, S.; Hover, F.S.; Damus, R.S.; Desset, S.; Morash, J.; Polidoro, V.; “Ship hull inspection by hull-relative navigation and control” in Proceedings of MTS/IEEE OCEANS, 2005, 17-23 Sept. 2005 Page(s):761 - 766 Vol. 1

[6] Urick, R.J.: "Principle of Underwater Sound for Engineers", McGraw-Hill Book Co., 1975.

[7] Mikhalevsky, P.N.; Gavrilov, A.N.; Baggeroer, A.B.; “The Transarctic Acoustic Propagation Experiment and climate monitoring in the Arctic.” IEEE Journal of Oceanic Engineering 24:2, April 1999. pgs:183 - 201

[8] Sozer, E. M. and M.Stojanovic, ``Reconfigurable Acoustic Modem for Underwater Sensor Networks," in Proc. First ACM International Workshop on Underwater Networks (WUWNeT'06) / MobiCom 2006, Los Angeles, CA, September 2006.

[9] Desset, S.; Damus, R.; Hover, F.; Morash, J.; Polidoro, V.; “Closer to deep underwater science with ODYSSEY IV class hovering autonomous underwater vehicle (HAUV).” Oceans 2005 – Europe, 20-23 June 2005. Page(s):758 - 762 Vol. 2

[10] Ware, J., Grund, M., Liberatore, S., Koski, P., Faluotico, S., "A Solar Powered Ocean Observatory Using Acoustic and Iridium Links," in Proceedings of the IEEE Oceans Conference, Washington DC, 2005.

[11] Stokey, R.P.; Freitag, L.E.; Grund, M.D. “A Compact Control Language for AUV Acoustic Communication,” Oceans 2005 – Europe Volume 2, 20-23 June 2005 Page(s):1133 - 1137 Vol. 2

[12] Cote, G.; Erol, B.; Gallant, M.; Kossentini, F.; “H.263+: video coding at low bit rates.” IEEE Transactions on Circuits and Systems for Video Technology, 8:7, Nov.1998 pgs:849 - 866

[13] Hoag, D.F.; Ingle, V.K.; Gaudette, R.J.; “Low-bit-rate coding of underwater video using wavelet-based compression algorithms.” IEEE Journal of Oceanic Engineering, 22:2 April 1997 pgs:393 - 400

[14] Datasheets on modem manufacturers' web sites, retrieved 2007-10-15:
http://www.benthos.com/pdf/telesonar2007.pdf
http://www.link-quest.com/html/intro1.htm
ftp://ftp.sercel.com/pdf/brochures/MATS.pdf
http://acomms.whoi.edu/
http://www.evologics.de/documents/S2C%20Modem%20Series%20Datasheets.pdf

[15] Stojanovic, M., J.Proakis and J.Catipovic, "Performance of High-Rate Adaptive Equalization on a Shallow Water Acoustic Channel," Journal of the Acoustical Society of America, vol.100 (4), Pt.1, October 1996, pp.2213-2219.

[16] Pelekanakis, C., M.Stojanovic and L.Freitag, ``High Rate Acoustic Link for Underwater Video Transmission," Proc. IEEE Oceans’03 Conference, San Diego, CA, September 2003.

[17] M.Stojanovic, J.Catipovic and J.Proakis, “Phase coherent digital communications for underwater acoustic communications,” IEEE J. Ocean. Eng. OE-19, 100-111 (1994)

[18] Proakis, J.G.; “Adaptive equalization techniques for acoustic telemetry channels .” IEEE Journal of Oceanic Engineering, Volume 16, Issue 1, Jan. 1991 Page(s):21 - 31

[19] Freitag, L., Grund, M., Singh, S., Partan, J., Koski, P., Ball, K., "The WHOI Micro-Modem: An Acoustic Communcations and Navigation System for Multiple Platforms," in IEEE Oceans Conference, Washington DC, 2005.

[20] Beaujean, Pierre-Philippe J.; “High-Speed High-Frequency Acoustic Modem for Image Transmission in Very Shallow Waters.” OCEANS 2007 – Europe 18-21 June 2007 Page(s):1 - 6

[21] Johnson, M. of Chevron Corporation. Personal communication, December 11th 2007

[22] Morash, J., J. Eskesen, D. Owens, V. Polidoro. “Test deployments of the Reef Explorer AUV,” October 2007. AUV Lab Cruise Report, MIT Sea Grant College Program, Cambridge, MA,.

[23] "Information technology - Digital compression and coding of continuous-tone still images - Requirements and guidelines" in International Telecommunications Union Recommendation T.81. Luminance quantization table in Appendix K. Available at http://www.itu.int/rec/T-REC-T.81-199209-I/en

[24] M.Stojanovic, J.Catipovic and J.Proakis, "Adaptive Multichannel Combining and Equalization for Underwater Acoustic Communications," Journal of the Acoustical Society of America, vol.94 (3), Pt.1, September 1993, pp.1621-1631.

[25] M.Stojanovic, ``Low Complexity OFDM Detector for Underwater Acoustic Channels," in Proc. IEEE Oceans'06 Conference, Boston, MA, September 2006.

[26] Kiely, A. and Klimesh, M. "Preliminary Image Compression Results from the Mars Exploration Rovers." NASA / JPL InterPlanetary Network (IPN) Progress Report 42-156, published February 15, 2004. Available at
http://tmo.jpl.nasa.gov/progress_report/42-156/156I.pdf

[27] "Open source video codecs and containers", in Wikipedia, The Free Encyclopedia.
Cited as a link to an up-to-date list of rapidly evolving free software projects.
http://en.wikipedia.org/wiki/Open_source_codecs_and_containers