Electric Field Imaging

by

Joshua Reynolds Smith

B.A., Williams College (1991)
S.M., Massachusetts Institute of Technology (1995)
M.A., University of Cambridge (1997)

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1999

Author .....
Program in Media Arts and Sciences,
School of Architecture and Planning
November 24, 1998

Certified by .....
Neil Gershenfeld
Associate Professor of Media Technology
Thesis Supervisor

Accepted by .....
Stephan A. Benton
Chairman, Department Committee on Graduate Students

Electric Field Imaging

by

Joshua Reynolds Smith

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
November 24, 1998
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Abstract

The physical user interface is an increasingly significant factor limiting the effectiveness of our interactions with and through technology. This thesis introduces Electric Field Imaging, a new physical channel and inference framework for machine perception of human action. Though electric field sensing is an important sensory modality for several species of fish, it has not been seriously explored as a channel for machine perception. Technological applications of field sensing, from the Theremin to the capacitive elevator button, have been limited to simple proximity detection tasks. This thesis presents a solution to the inverse problem of inferring geometrical information about the configuration and motion of the human body from electric field measurements. It also presents simple, inexpensive hardware and signal processing techniques for making the field measurements, and several new applications of electric field sensing.

The signal processing contribution includes synchronous undersampling, a narrowband, phase sensitive detection technique that is well matched to the capabilities of contemporary microcontrollers. In hardware, the primary contributions are the School of Fish, a scalable network of microcontroller-based transceive electrodes, and the LazyFish, a small footprint integrated sensing board. Connecting n School of Fish electrodes results in an array capable of making heterodyne measurements of any or all $n(n-1)$ off-diagonal entries in the capacitance matrix. The LazyFish uses synchronous undersampling to provide up to 8 high signal-to-noise homodyne measurements in a very small package. The inverse electrostatics portion of the thesis presents a fast, general method for extracting geometrical information about the configuration and motion of the human body from field measurements. The method is based on the Sphere Expansion, a novel fast method for generating approximate solutions to the Laplace equation. Finally, the thesis describes a variety of applications of electric field sensing, many enabled by the small footprint of the LazyFish. To demonstrate the School of Fish hardware and the Sphere Expansion inversion method, the thesis presents 3 dimensional position and orientation tracking of two hands.[1]

Thesis Supervisor: Neil Gershenfeld
Title: Associate Professor of Media Technology

[1] Please see the URL http://www.media.mit.edu/people/jrs/thesis.html for video clips, code, and other information related to this thesis.

Electric Field Imaging

by

Joshua Reynolds Smith

The following people served as readers for this thesis:

Reader: _____

William H. Press
Professor of Astronomy and of Physics
Harvard University

Reader: _____

J. Turner Whitted
Senior Researcher
Microsoft Research

For Dad, who invented the busybox and showed me my first circuit.

Contents

Chapter 1

Introduction

The title of this thesis is misleading. "Machine Electric Field Perception" would be more accurate, but that is quite a mouthful. The reason it is a mouthful is that humans do not (yet?) have a word for electric field perception, because humans do not perceive the world with electric fields. Because of our congenital blindness to low frequency electric fields, we as a species were not even aware until quite recently that several species of fish do perceive the world in this way. The purpose of this work is to demonstrate that what works for fish could work for our machines too, and that endowing them with this sense could make them more useful.

Electric Field Imaging is a new physical channel and inference framework for machine perception of human action. The physical user interface has become a significant factor limiting the effectiveness of our interactions with and through technology, and currently almost any progress in machine perception can be used to create better interfaces. The physical user interface is thus the largest and most visible customer for machine sensing and perception, and so the user interface will often be visible in the foreground of this thesis. But efforts to improve machine perception could have implications even beyond improving user interfaces.

Alan Turing, in one of his less famous papers, called "Intelligent Machinery," considered a perceptual route to machine intelligence, which he rejected as slower than the route that Artificial Intelligence (AI) actually attempted:

> One way of setting about our task of building a "thinking machine" would be to take a man as a whole and to try to replace all the parts of him by machinery. He would include television cameras, microphones, loudspeakers, wheels and "handling servo-mechanisms" as well as some sort of "electronic brain." This would be a tremendous undertaking, of course. The object, if produced by present techniques, would be of immense size, even if the "brain" part were stationary and controlled the body from a distance. In order that the machine should have a chance of finding things out for itself it should be allowed to roam the countryside, and the danger to the ordinary citizen would be serious. ... [A]lthough this method is probably the "sure" way of producing a thinking machine it seems to be altogether too slow and impracticable.
>
> Instead we propose to try and see what can be done with a "brain" which is more or less without a body...[Tur47]

Turing's second, "bodiless" path dominated AI for decades, but today, few would characterize it as a quick and easy approach. This thesis can be viewed as a small step along the other path.

Of course, the perceptual route to AI has not been completely neglected. Machine perception (comprised primarily of the subfields of machine vision and machine audition) is in fact a very mature field in which a large number of researchers have been gainfully employed for many years. However, I believe that the main stream of machine perception work suffers from a "commonplace" that Turing mentions in the same article:

> A great positive reason for believing in the possibility of making thinking machinery is the fact that it is possible to make machinery to imitate any small part of a man. That the microphone does this for the ear, and the television camera for the eye are commonplaces.[Tur47]

But in fact, the microphone and television camera were not designed to help machines perceive: they were designed to transduce signals that humans would ultimately consume. A television camera is in fact quite different from an eye, and inferior in crucial ways. Nevertheless, until recently, most work on machine perception used transducer systems that were designed for people, not for machines. In 1980, David Marr posed the problem of machine vision in the following way:

> The problem begins with a large, grey-level intensity array, which suffices to approximate an image such as the world might cause upon the retinas of the eyes, and it culminates in a description that depends on that array....[Mar80]

The grey-level intensity array is in fact what a television camera returns. Contrary to Marr's claim, it has very little to do with what a retina senses, since the retina is foveated: it has a small high resolution spot in the center, and much lower resolution elsewhere. From a mathematical point of view, this distinction is not significant, since it is not necessary to use all of the data returned by the TV camera. But from a computational and technical point of view, this is an enormous difference, since it is a difficult technical problem to get such large quantities of data into the computer, and a difficult computational problem to distill such large quantities of data into a useful representation.

Existing efforts at machine perception have imitated human capabilities at once too literally and too sloppily: the wrong aspects have been copied. Why are machines typically given only the senses that humans have? Why are highly parallel, high resolution sensor systems such as video cameras, which are well matched to human capabilities, fed into our presently serial computers, to which they are poorly matched? Like human eyes, the video cameras commonly used by the machine perception community are optical sensors. But perhaps machine perception research has not absorbed the abstract lessons that the example of the eye (and in particular, the fovea) can teach: apply a small amount of sensing, communication, and computational resources where they are really needed, rather than expend resources freely everywhere. It may be that the early efforts at machine perception, which have used transducers designed for human perception, will appear as quaint as early attempts to make flying machines with flapping wings...these copied features of flight that turned out to be specific to birds, rather than abstracting the deeper lessons of aerodynamics that bird flight also contains.

This thesis does not solve the problem of flight. My hope is that in fifty years, it might look like an early experiment with fixed wing aircraft—a step in the right direction.

As part of my thesis work, I have designed and built electric field sensing hardware specifically for the purpose of giving computers better perceptual capabilities. There is no guarantee that slavish imitation of fish sensing will bring us closer to machine intelligence than slavish imitation of humans did. However, we may have fewer preconceptions about a sense that we lack, and in the worst case, slavish imitation of something different is bound to produce a new perspective at the very least.

The School of Fish, which is described in some detail in chapter 3, is a network of intelligent electrodes capable of both generating and sensing electric fields. For simple perceptual tasks, a small number of units may be used; for more complex tasks, such as trying to infer the 3d geometry of a person's hands (to make a "Field Mouse"), more units may be strung together. Each unit can be operated as a transmitter or a receiver. This means that the system's "focus of attention" can be directed to a certain region by choosing which transmitters to activate. Unlike a video camera, which always returns a fully detailed image of whatever is placed in front of it, the School of Fish can focus its resources in a particular region, similar to the way the human visual system allocates its resources selectively by directing the fovea to points of interest.

The fact that the system can focus its attention in different areas means that a constant stream of commands from the host computer to the sensor system is required, to tell it where and how to "look." Until recently, most machine sensing devices have been primarily feed forward: information flows mainly from the sensor to the computer. The School of Fish is one of the few sensing devices I know of in which the rate of information flow *to* the device is comparable to the rate *from* the device. From a philosophical perspective, most existing sensors embody an empiricist view of perception and the mind.

If philosophy were the only requirement, then most sensing systems could have been designed by Hume, who argued that the senses create impressions on the mind in a unidirectional process.[Hum48] I certainly would not claim that the School of Fish embodies any philosophical breakthroughs, but I do believe that its invention would probably have had to wait at least until Kant, who recognized the mind's active role in structuring sensory experience.[Kan83] When discussing a system like the School of Fish, the term sense *data*—that which is *given* by the senses—becomes less appropriate, since the system is actively probing, asking questions of its environment, rather than passively receiving impressions from the world.

1.1 Organization of the thesis

The first six chapters of this thesis tell a coherent story, of how a machine can track the 3d configuration of the human body using electrical measurements. Chapter 7 describes a variety of industrial and artistic applications of the work presented in chapters one through six, and chapters 8, 9, and 10 discuss some extensions of the main ideas that do not contribute directly to the "plot." After the conclusion, several appendices provide additional technical details.

This introductory chapter (1) has three major sections remaining. The first of these discusses previous examples of sensing with electric fields. The second section explains the underlying physical mechanisms, and the third section introduces synchronous detection and related signal processing concepts that are important to the thesis. The second chapter (2), "Synchronous Undersampling and the LazyFish," presents signal processing techniques and hardware that allow high quality electric field measurements to be made very inexpen-

sively. Next comes "The School of Fish," chapter (3), the scalable network of electric field transceivers that we use to collect the data for the inverse problem.

Then, a chapter that introduces inverse problems generally (4) is followed by one on the specific problem of inverse electrostatics (5). "The Sphere Expansion," chapter (6), presents a fast approximate method for solving the forward problem, a necessary piece of the electrostatic inverse problem, and describes a practical implementation with all the pieces working together: the data is collected with the School of Fish, and then the inverse problem is solved by searching the sphere expansion forward model space. The system is able to track the position and orientation of one or two hands.

Chapter (7) presents a variety of artistic and industrial applications of the work in the earlier chapters. "Code Division Multiplexing of a Sensor Channel," chapter (8), explores a variation of the LazyFish approach to software demodulation. It presents a software implementation of a spread spectrum, code division multiplexed electric field sensing system. The system allows multiple sensor channels to be simultaneously measured using a single analog sensor front end. The next chapter (9) illustrates an application of the spread spectrum detection technique in an apparently unrelated application domain: digital watermarking. Finally, chapter (10) explores the problem of how to automatically assign IDs in a distributed system of identical units such as the School of Fish. After the conclusion, chapter (11), are three appendices presenting technical details of three circuit boards I designed or co-designed: the LazyFish, the School of Fish, and the MiniMidi synthesizer on which I collaborated with Josh Strickon.

Having outlined the thesis, I will now continue on to the precedents and background for the work.

1.2 Precedents

In this section I will describe the "prior art," the examples I know of of sensing with electric fields. Biology, specifically fish, got there first, as I'll explain in the next section. Next came music: a version of electric field sensing was used in one of the first electronic musical instruments, the Theremin, early in the twentieth century. Not long after, geophysicists began using electric field measurements of the earth to prospect for oil and other minerals. The same techniques have recently been applied for archaeological purposes. In the last twenty years, electric field measurement techniques have been used in medicine. Electrical Impedance Tomography is a safe, inexpensive, high update rate, low resolution technique used to form impedance images of the inside of the body. In the next few sections, I'll describe these antecedents of electric field imaging.

1.2.1 Biological: Fish

Many species of fish use electric fields to perceive their environments. The capability has evolved independently more than once: species in two distantly related families of fish, *Mormyriformes* and *Gymnotoidei,* one from South America and one from Africa, have this capability. This split is reflected in figure 1-1, a "family tree" of the fish known to have electrosensory capabilities. The fact that electric field sensing does not require an external light source and is unaffected by optical scatterers like mud or silt is presumably advantageous to fish in dark, murky water anywhere. Electric Field sensing is another example of convergent evolution, the best known example being the eye, which evolved independently in squid and mammals.

In all examples of fish electric field sensing, a current source in the tail induces voltages along the lateral line. As the fish nears an object with a dielectric constant different than water, the induced voltages change. Figure 1-2 shows the electric field lines around a fish being distorted by a nearby dielectric object. Experiments have confirmed that fish are indeed sensitive to the dielectric constant, and have explored the size and distance limits of the fish's sensitivity. The [species of fish used in the experiment] prefer to be at a particular distance from obstacles. When an obstacle is slowly moved, the fish will adjust its position to maintain a particular distance from the object. The size of the electrically detectable object was decreased until the fish no longer followed it. The experiments controlled for other sensing modalities to make sure that the fish were indeed sensing dielectric constant, rather than using acoustic or optical cues. The target object was optically transparent and refractive index matched to the water. Furthermore, the cylindrical dielectric target was embedded in a larger plexiglass cylinder. Even as the size of the dielectric target was decreased, the size of the surrounding cylinder was kept constant, so that only the electric cues, and not the acoustic, would vary.

Another electric fish behavior that has been investigated is a tendency to curl the tail around objects of interest. Numerical models of the fish suggest that moving the tail gives a better "view." Figure 1-3 shows an electric fish, species *Eigenmania,* alongside an abstracted version used for electrical modeling of the fish's sensory capabilities. In figure 1-4, the simulated signals due to the two dielectric objects appear to be better resolved as the tail curls further around them. The fish may also be making use of information about the derivative of voltage with respect to tail position that it acquires as it curls the tail.

1.2.2 Musical: Theremin

In addition to being the first human implementation of sensing with electric fields, the Theremin was also notable as one of the first electronic musical instruments of any kind. Figure 1-5 shows a 1928 poster for a Theremin "concert demonstration" at Carnegie hall. Figure 1-6, taken from an RCA catalog, shows the Theremin in operation. Volume is controlled by the left hand, pitch by the right. Chapter 2 contains a more detailed discussion of the Theremin instrumentation.

1.2.3 Geophysical: Electrical Prospecting

Geophysical applications of electrical measurements are perhaps the most mature. Electrical prospectors use measurements of the earth's resistivity to hunt for oil or minerals. These electrical methods are a standard tool covered in introductory geophysics texts, and large successful companies, such as Schlumberger, have been built on the success of electrical prospecting.

Two geometries are commonly used for prospecting. In one, a voltage is applied between two distant electrodes on the earth's surface, and the voltage gradient in the region between these electrodes is mapped using a second pair of closely spaced electrodes. Figure 1-7 shows two electrode configurations: the drive electrodes are at the same location in both, but the sense electrodes have moved. Deviations from uniform resistivity lead to deviations from the baseline voltage gradient. The resistivity map can be used to locate ore deposits.

In the second technique, a probe is lowered down the bore hole of an oil well. A voltage is applied to the main probe electrode, and the resulting current is measured. Guard electrodes above and below the main electrode are driven at the same voltage to reduce the

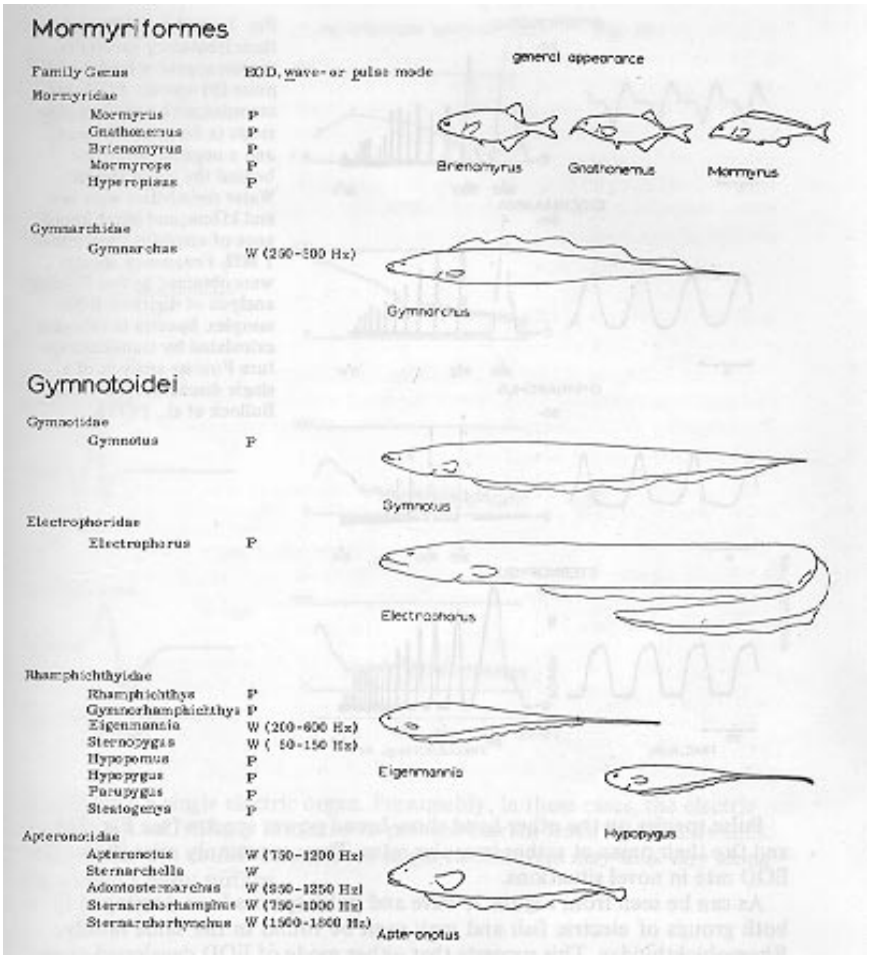

Figure 1-1: The two major families of weakly electric fish. Electric field sensing evolved independently in these two families.

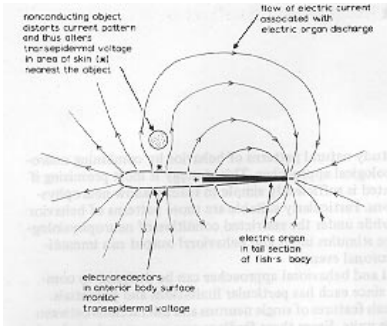

Figure 1-2: A fish's sensing field being distorted by a dielectric object.

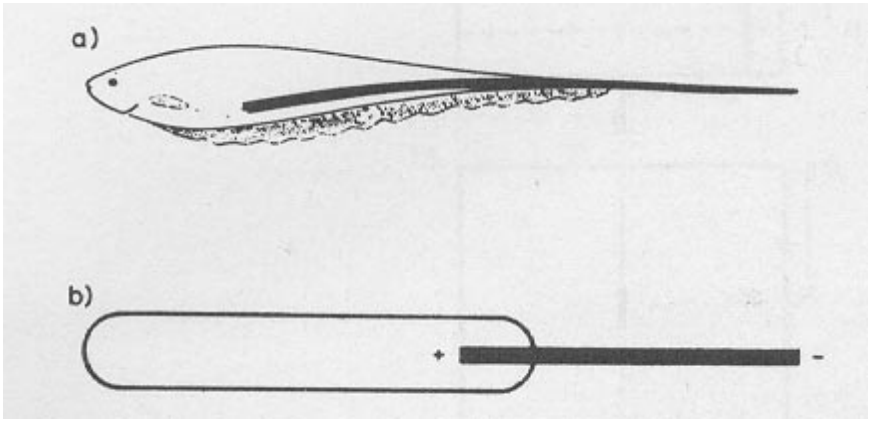

Figure 1-3: A weakly electric fish alongside an abstracted version used for electrical modeling.

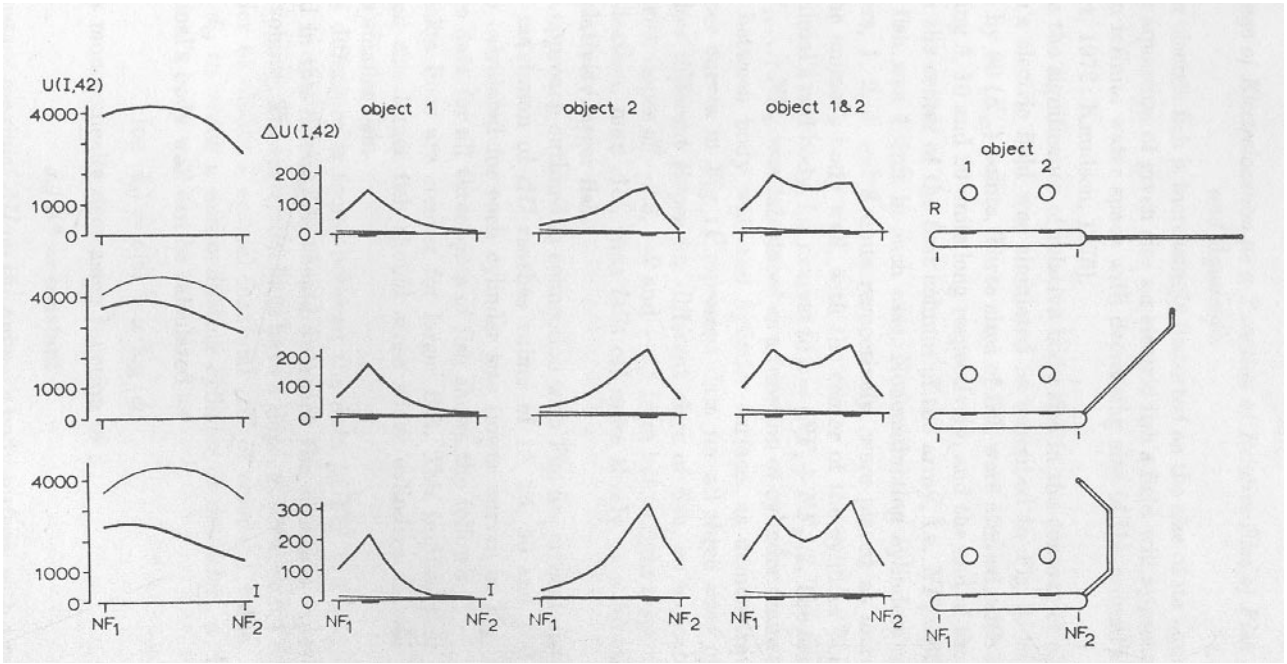

Figure 1-4: A fish "imaging" two objects. By wrapping its tail around the objects, it is able to resolve them better.

Figure 1-5: A poster from a 1928 Theremin concert at New York City's Metropolitan Opera House.

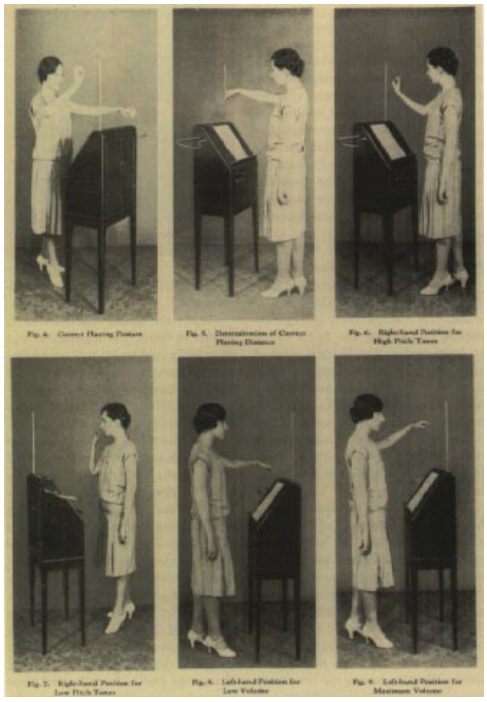

Figure 1-6: Pages from an RCA catalog showing the operation of the Theremin.

sensitivity of the measurement to features above or below the probe. Figure 1-8 shows a probe, with field lines from the main and guard electrodes.

1.2.4 Medical: Electrical Impedance Tomography

In Electrical Impedance Tomography (EIT), a ring of electrodes is attached to the abdomen, thorax, or occasionally an arm or leg. Currents are applied between pairs of electrodes, and the resulting voltages are measured. Using a variety of algorithms, the cross sectional impedance map is recovered. Figure 1-9 is a block diagram of a typical EIT system. Figure 1-10 shows the reconstructed impedance map of a forearm, alongside an anatomical section of the arm.

1.3 Physical Mechanisms

Figure 1-11 is a lumped circuit model of electric field sensing. The general term electric field sensing actually encompasses several different measurements, which correspond to different current pathways through this diagram. In all the sensing modes, a low frequency (from 10-100kHz) voltage signal is applied to the transmit electrode, labeled T in the figure. Displacement current flows from the transmitter to the other conductors through the effective capacitors shown in the diagram.

In *loading mode,* the current flowing from the transmitter is measured. The value of $C1$, and thus the load on the transmitter, changes with hand position: when the hand, labeled H in the figure, moves closer to the transmitter, the loading current increases. The term capacitive sensing ordinarily refers to a loading mode measurement. However, the capacitances other than $C1$ in the figure suggest other measurements. We will see that

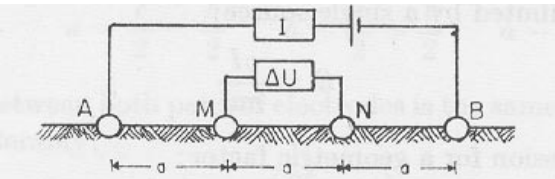

FIG. 55.

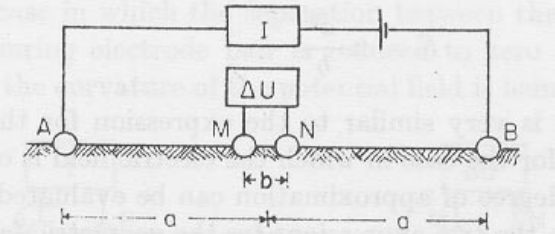

FIG. 56.

Figure 1-7: Configuration of apparatus used to make surface electrical measurements for geophysical prospecting.

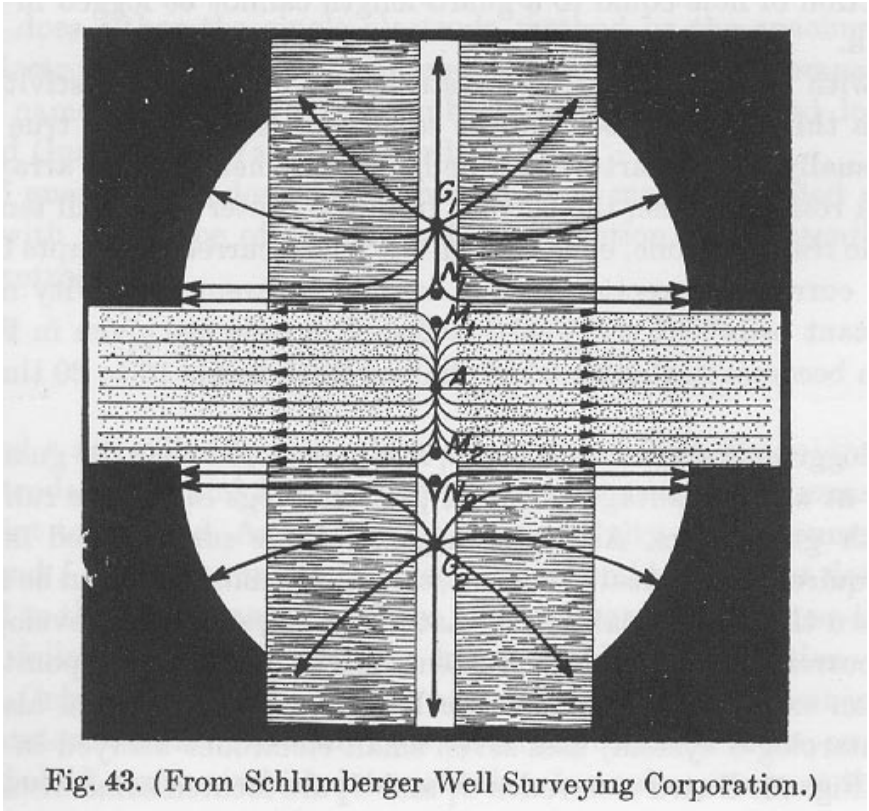

Fig. 43. (From Schlumberger Well Surveying Corporation.)

Figure 1-8: Borehole electrical measurement apparatus.

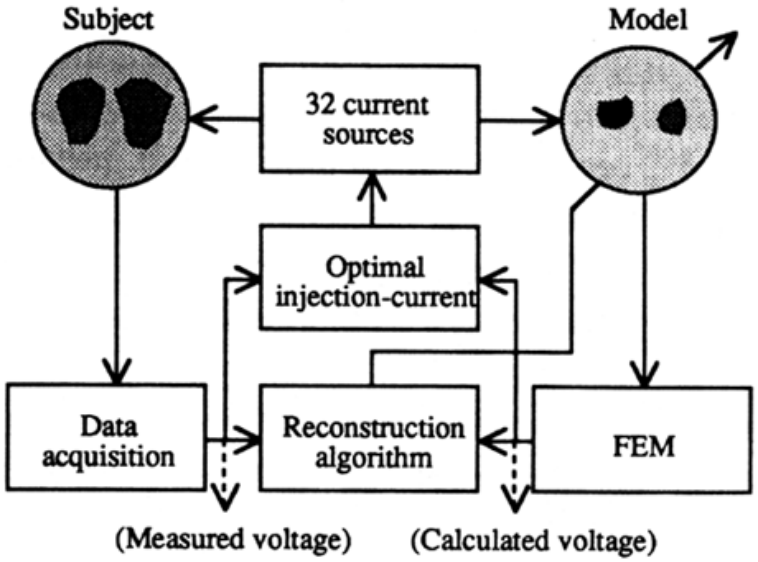

Fig. 1. Block diagram of EIT system.

Figure 1-9: Block diagram of a typical Electrical Impedance Tomography system.

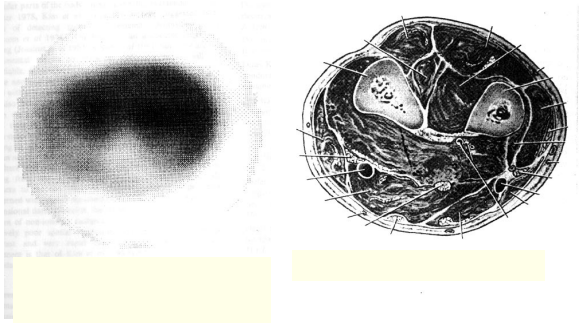

Figure 1-10: A cross-sectional reconstruction of the impedance map of an arm, alongside a drawing of the actual arm cross section.

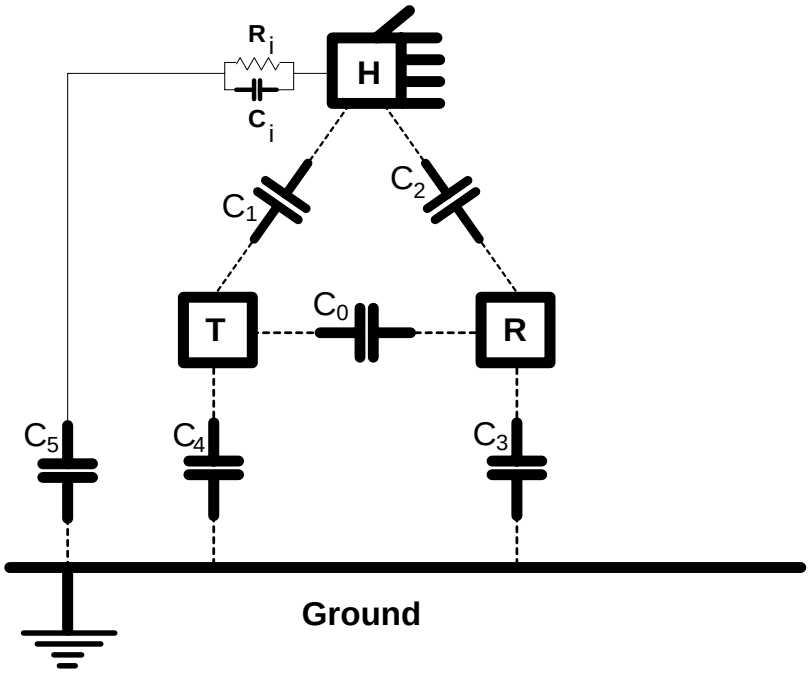

Figure 1-11: Lumped circuit model of Electric Field Sensing

there is more to capacitive sensing—defined broadly—than loading mode measurement.

In *transmit mode*, the transmitter is coupled strongly to the body—C1 is very large—so the hand is essentially at the potential of the transmitter. As the body approaches the receive electrode, labeled R in the figure, the value of C2 (and C0—the two are not distinct in this mode) increases, and the displacement current received at R increases.

Shunt mode measurements are most important for this thesis. In the shunt mode regime, C0, C1, and C2 are of the same order of magnitude. As the hand approaches the transmitter and receiver, C1 increases and C0 decreases, leading to a drop in received current: displacement current that had been flowing to the receiver is shunted by the hand to ground (hence the term shunt mode). We measure a baseline received current when the hand is at infinity, and then subtract later readings from this baseline.

With N ordinary capacitive sensors (loading mode), one can collect N numbers. These N numbers turn out to be the diagonal of the capacitance matrix for the system of electrodes. In shunt mode, one measures the $N(N-1)$ off diagonal elements. Because the capacitance matrix is symmetrical, there are ideally only $\frac{1}{2}N(N-1)$ distinct values. In practice, measured deviations from symmetry provide valuable calibration information.

The basic physical mechanisms of electric field sensing have not changed since my 1995 Master's thesis, or, for that matter, since the 1873 publication of Maxwell's Treatise on Electricity and Magnetism.[Max73] Therefore, I am reproducing the discussion of the basic physical mechanisms from my Master's thesis. The discussion beginning below and continuing until the section on signal processing is reproduced from my Master's thesis.

1.3.1 Derivation of Circuit Model from Maxwell Equations

This derivation follows Fano, Chu, and Adler closely.[FCA60] Maxwell's equations can be written in the form

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{1.1}$$

$$\nabla \times \mathbf{H} = \mathbf{J}_f + \frac{\partial \mathbf{D}}{\partial t} \tag{1.2}$$

$$\nabla \cdot \mathbf{D} = \rho_f \tag{1.3}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{1.4}$$

$$\nabla \cdot \mathbf{J}_f = -\frac{\partial \rho_f}{\partial t} \tag{1.5}$$

where, for linear and isotropic media,

$$\mathbf{D} = \epsilon \mathbf{E}$$

$$\mathbf{B} = \mu \mathbf{H}$$

$$\mathbf{J}_f = \sigma \mathbf{E}$$

Electric Field Sensing uses low frequency fields. To study the properties of low-frequency solutions of the Maxwell equations, we can introduce a time-rate parameter α and a new, scaled time $\tau = \alpha t$. Small values of α map long periods of real time t into a unit of scaled time. Thus slow or low-frequency behavior corresponds to small values of α. The low-frequency behavior is therefore described by the low order terms in an expansion of the fields in a power series in α.

We can put a rough physical interpretation on α: its value is the ratio between the time τ for an electromagnetic wave to propagate across the longest lengthscale in the problem (the characteristic time for wave behavior), and the smallest time t of interest, in our case the period of the highest frequency that our oscillator can produce. Note that the period of our oscillator is slow compared to the wave propagation time, so α is small. The expansion of $\mathbf{E}$ in powers of α has the form

$$\mathbf{E}(x,y,z,t) = \mathbf{E}(x,y,z,\tau,\alpha) = \mathbf{E}_0(x,y,z,\tau) + \alpha\mathbf{E}_1(x,y,z,\tau) + \alpha^2\mathbf{E}_2(x,y,z,\tau) + \ldots$$

where

$$\mathbf{E}_0(x,y,z,\tau) = [\mathbf{E}(x,y,z,\tau,\alpha)]_{\alpha=0}$$

$$\mathbf{E}_1(x,y,z,\tau) = \left[\frac{\partial \mathbf{E}(x,y,z,\tau,\alpha)}{\partial\alpha}\right]_{\alpha=0}$$

$$\mathbf{E}_k(x,y,z,\tau) = \frac{1}{k!}\left[\frac{\partial^k \mathbf{E}(x,y,z,\tau,\alpha)}{\partial\alpha^k}\right]_{\alpha=0}$$

When the frequency is low enough that all but the zeroth and first order terms can be neglected, the solution is called quasi-static.[FCA60]

Using the new, scaled time τ, time derivatives will be multiplied by α, for example:

$$\frac{\partial \mathbf{B}}{\partial t} = \frac{\partial \mathbf{B}}{\partial \tau}\frac{\partial \tau}{\partial t} = \alpha\frac{\partial \mathbf{B}}{\partial \tau}$$

The three Maxwell equations involving time derivatives become

$$\nabla \times \mathbf{E} = -\alpha\frac{\partial \mathbf{B}}{\partial \tau} \tag{1.6}$$

$$\nabla \times \mathbf{H} = \mathbf{J}_f + \alpha\frac{\partial \mathbf{D}}{\partial \tau} \tag{1.7}$$

$$\nabla \cdot \mathbf{J}_f = -\alpha\frac{\partial \rho_f}{\partial \tau} \tag{1.8}$$

Substituting the expanded $\mathbf{E}$ and $\mathbf{B}$ fields back into the scaled Maxwell equation 1.6 and grouping terms, 1.6 becomes

$$\nabla \times \mathbf{E}_0 + \alpha(\nabla \times \mathbf{E}_1 + \frac{\partial \mathbf{B}_0}{\partial \tau}) + \alpha^2(\nabla \times \mathbf{E}_2 + \frac{\partial \mathbf{B}_1}{\partial \tau}) + \ldots = 0$$

Each term in the sum must equal zero individually for the equation to hold for all values of α. This defines a series of equations whose solution is the series of fields that make up our expansion. Because the $\mathbf{B}$ term in 1.6 is multiplied by α, and the $\mathbf{E}$ term is not, kth order $\mathbf{E}$ terms are related in the infinite series of equations to $k-1$th order $\mathbf{B}$ terms. The expansion of equation 1.7 will yield a series of equations coupling kth order $\mathbf{B}$ fields to $k-1$th order $\mathbf{E}$ fields. Since all fields are coupled only to lower order fields, any number of terms can be evaluated, by starting from the zeroth order solution, using that to find the first order, and so on. The zeroth order $\mathbf{E}$ field equations are

$$\nabla \times \mathbf{E}_0 = 0 \tag{1.9}$$

$$\nabla \times \mathbf{H}_0 = \mathbf{J}_{f0} \tag{1.10}$$

$$\nabla \cdot \mathbf{J}_{f0} = 0 \tag{1.11}$$

The Maxwell equations that do not involve time derivatives become:

$$\nabla \cdot \epsilon \mathbf{E}_0 = \rho_{f0} \tag{1.12}$$

$$\nabla \cdot \mu \mathbf{H}_0 = 0 \tag{1.13}$$

Next we will write out the first order fields. Since all values of α correspond to physically realizable fields, any field can be viewed as the original, "unscaled" field. Therefore no loss of generality results from setting $\alpha = 1$, and writing t instead of τ:

$$\nabla \times \mathbf{E}_1 = \mu \frac{\partial \mathbf{H}_0}{\partial t} \tag{1.14}$$

$$\nabla \times \mathbf{H}_1 = \epsilon \frac{\partial \mathbf{E}_0}{\partial t} + \mathbf{J}_{f1} \tag{1.15}$$

$$\nabla \cdot \epsilon \mathbf{E}_1 = \rho_{f1} \tag{1.16}$$

$$\nabla \cdot \mu \mathbf{H}_1 = 0 \tag{1.17}$$

$$\nabla \cdot \mathbf{J}_{f1} = -\frac{\partial \rho_{f0}}{\partial t} \tag{1.18}$$

Because the curl of any vector field V equals zero if and only if V can be written as the gradient of a scalar potential, equation 1.9 implies that $\mathbf{E}_0 = \nabla \phi_0$. In a region with no sources or sinks, any vector field satisfies $\nabla \cdot \mathbf{V} = 0$, so if there are no free charges, $\nabla \cdot \nabla \phi_0 = \nabla^2 \phi_0 = 0$; that is, ϕ_0 satisfies Laplace's equation. If free charges are present, then ϕ_0 satisfies Poisson's equation, by a similar argument.

Quasistatic limit

In terms of our expansion above, the quasi-static condition holds when $\alpha = \frac{\tau}{t} = \frac{L}{ct} \ll 1$, because higher powers of α are negligible when $\alpha \ll 1$. Again, τ is the time for an electromagnetic wave to propagate across the longest lengthscale in the problem, and t is the period of the transmit oscillator. If L is 10 meters and the transmit frequency is 100kHz, so that $t = 1.0 \times 10^{-5}$, then $\alpha = 3.3 \times 10^{-3} \ll 1$, so we are comfortably in the quasistatic regime.

When α is vanishingly small, so that only the zeroth order terms are required, we are in the regime of DC circuits. For small but finite rates of change, the first order terms must also be taken into account. This is the regime of AC circuitry. In the next section we will see in more detail how the concepts and laws of circuit theory emerge naturally as the quasistatic limit of the Maxwell equations.

Circuit Theory

There are three basic types of solutions to the zeroth and first order Maxwell equations, which correspond to the three basic types of circuit components: capacitive, inductive, and resistive. For Electric Field Imaging, only the capacitive solutions are relevant;[1] for Electrical Impedance Tomography only the resistive solutions matter (for this reason the

[1] We will see later that the situation is slightly more complicated than this.

name Electrical Resistivity Tomography would be more accurate). We will see, however, that the equations specifying the "capacitive" and "resistive" fields are identical in form, which might be guessed from the fact that resistance and capacitance can be viewed as special cases of the generalized circuit concept of impedance.

The three types of quasi-static fields can be classified according to their zeroth-order terms. The first two types arise when there is no conduction current. In these first two cases the right side of equation 1.10 is zero, and there is no coupling between the electric and magnetic fields, so the two can be treated separately. The first type of quasistatic solution, electrical, has no magnetic component, and will be associated with capacitance, as we will explain below. A magnetic solution with no electrical component will be associated with inductance. The solution associated with resistance arises when conduction currents are present. If $\mathbf{J}_{f0} = \sigma \mathbf{E}_0$, then equation 1.10 becomes $\nabla \times \mathbf{H_0} = \sigma \mathbf{E}_0$. Thus in resistive solutions the zeroth order electric field is coupled to the zeroth order magnetic field through a finite conductivity.

To see why the electrical solution is associated with capacitance, first recall the circuit definition of capacitance:

$$I = C\frac{dV}{dt} \tag{1.19}$$

A capacitance couples a current to the time derivative of a voltage. Now consider the "capacitive" field. Because of equation 1.15, a zeroth order electric field induces a first order magnetic field proportional to the time derivative of the electric field. Associated with the zeroth-order electric field is a zeroth-order charge; by equation 1.18, the time derivative of this charge induces a first order current. Since the zeroth order electric field may be represented by a scalar potential, this first order current is coupled to the time derivative of the zeroth order potential. As we saw in equation 1.19, this type of coupling is referred to as capacitive in circuit theory. Similar arguments demonstrate the correspondence between the other types of fields and circuit components.

Electrostatics

Our expansion showed that static (zeroth order) electric fields satisfy Laplace's equation. The behavior of the static fields is crucial to Electric Field Sensing, because, as we shall see in section 1.3.1, though EF sensing requires first order fields to operate, no new information is contained in the first order fields; it is all present in the zeroth order. We now will show how to use quasistatic field solutions to calculate macroscopic circuit quantities such as capacitance and received current.

The static charge on a conductor i is due to the $\mathbf{E}_0$ field:

$$Q_i = -\int_{S_i} \epsilon \mathbf{n} \cdot \nabla \phi_0 da$$

where S_i is the surface of i, $\mathbf{n}$ is the outward normal to S_i, and ϵ may be a function of position, since the medium need not be homogeneous.

Using the standard definition, the capacitance of conductor i due to a conductor j is the ratio between the charge on Q_i and the voltage between j and a reference. Of course if we know the capacitance and voltages for a pair of electrodes, we can find the charge induced on one by the other. Because of the linearity of all the equations involved, the total charge on i induced by all the other conductors is the sum of the separately induced

charges[FCA60] (note that the capacitances are not linear functions of position):

$$Q_i = \sum_j C_{ij} V_j \tag{1.20}$$

The off-diagonal terms of this capacitance matrix C_{ij} represent the ratio between Q_i and V_j when all the other Vs are zero. The diagonal "self-capacitance" terms C_{ii} represent the charge on i when it is held at V_i and all the other electrodes are at zero. The diagonal terms represent the "loading" of the transmit electrode by the body being measured. The matrix is symmetrical.

We will now see that from the capacitances, we can calculate the currents received at the electrodes. This is because equation 1.18 relates the first order current to the zeroth order charge. By charge continuity (expressed microscopically in equation 1.18), the current I_i entering receiver i is given by the time derivative of the charge on i: $I_i = \frac{dQ_i}{dt}$.

$$I_i = \frac{d}{dt} \sum_j C_{ij} V_j = \sum_j C_{ij} \frac{dV_j}{dt} \tag{1.21}$$

The currents that we measure in Electric Field Sensing are first order phenomena. However, we only use the currents to measure capacitance, the zeroth order property that is geometry dependent and therefore encodes the geometrical information that we ultimately want to extract. This tells us something about the physical limits on the time resolution of EF sensing: the "frame rate" must be much shorter than the characteristic time for first order phenomena, that is, the oscillator period.

Component Values

What are some the component values in figure 1-11? Do the details of the body's interior affect the signals? At the frequencies we are concerned with, the real impedance of free space is essentially infinite (capacitors block direct current), and the real impedance of the body is almost zero. Barber [BB84] gives resistivity figures on the order of $10\Omega m$ (Ohm-meters), plus or minus an order of magnitude: cerebrospinal fluid has a resistivity of $.65\Omega m$, wet bovine bone has $166\Omega m$, blood has $1.5\Omega m$, and a human arm has $2.4\Omega m$ longitudinally and $6.75\Omega m$ transverse.

A simple parallel plate model of feet in shoes with 1cm thick soles gives a capacitance of 35 pF, using $C = \epsilon_0 A/d$, and taking $A = 2$ feet $\times 20cm \times 10cm$ and $d = 1$cm. For 10 cm thick platform shoes, the value of $C = 3.5$pF. (We have neglected the dielectric constant of the soles.)

Having introduced the basic concepts and physical quantities on which Electric Field Imaging is based, we'll now introduce some of the signal processing techniques necessary to measure these quantities.

1.4 Signal processing: synchronous detection

Synchronous detection (also known as synchronous demodulation) is the basic signal processing primitive needed to make good quality capacitance measurements. In principle we could make the capacitance measurements at any frequency down to D.C., but because of $1/f$ noise, 60 Hz pick up, and other low frequency noise, it is preferable to make the measurements at higher frequencies. Since we will transmit at a particular, known phase

and frequency f, we can reject noise by filtering out received signal power that is outside a narrow band around f.

Synchronous demodulation is a way to make a filter with an extremely narrow pass band whose center frequency is precisely tuned to the transmit frequency. To explain synchronous demodulation, I'll now describe the basic measurement process again in signal processing terminology.

A sinusoidal carrier signal is applied to the transmit electrode, which induces a received signal consisting of an attenuated version of the transmit signal, plus noise. As the hand moves, the amount of attenuation changes. In other words, the hand configuration effectively amplitude modulates the carrier. Information about the hand configuration can be recovered by demodulating the carrier.

Figure 2-4 illustrates demodulation in the time and frequency domains. Our sinusoidal carrier $cos(2\pi ft)$ can be written $\frac{1}{2}(e^{2\pi ft} + e^{-2\pi ft})$ to make the negative frequency content explicit. The figure shows the sinusoid on the left, and its frequency domain representation, two delta functions at $+f$ and $-f$, on the right. The hand affects the amplitude of the received carrier. To demodulate, the attenuated (and phase shifted—more on this later) version of the carrier is multiplied by the original carrier, represented in the second row of the figure. This operation can be implemented using an analog multiplier. Later, I will describe digital implementations of the demodulation process. In the frequency domain, the multiplication operation is equivalent to convolution. The third row shows the time and frequency domain results of the operation: algebraically, it is $\frac{1}{2} + \frac{1}{4}e^{2\pi 2ft} + \frac{1}{4}e^{-2\pi 2ft}$, and it shows up as three delta functions in the frequency domain representation. To recover the amplitude information we're interested in, we low pass filter the resulting signal. This rejects the side bands at $+2f$ and $-2f$, leaving the DC value we are interested in. In the figure, the low pass filter is represented in the third frequency domain plot as a window around DC. Everything outside this window should be substantially attenuated.

1.4.1 Abstract view of synchronous detection

If the carrier is regarded as a vector in a Hilbert space—either finite or infinite dimensional—and we make certain assumptions to be explained below, then there is a simple algebraic interpretation of synchronous detection. Denote the carrier vector by ϕ. If the hand is stationary on the timescale of the measurement, then the signal at the receiver is $a\phi + N$, where the constant prefactor a represents the attenuation due in part to the hand, and N is a noise vector.

If the low pass filter operation is simply integration over the time of interest (recall that we have assumed the hand to be stationary during the measurement time—basically we are considering "pulsed" rather than continuous wave sensing), then we can interpret the demodulation operation as an inner product. In the continuous case the inner product would be defined $< f, g >= \int_0^T f(t)g(t)dt$. In the discrete case, the inner product would be $< f, g >= \sum_t f_t g_t$.

If ϕ is normalized, then the demodulation operation is $< \phi, a\phi >= a$. We will consider the discrete time demodulation operation in more detail later as we use it in particular cases.

1.4.2 Quadrature

So far we have assumed that the phase of the received signal is identical to the phase of the transmitted signal. This assumption is not justified in practice since the various capacitances in the system, particularly the capacitance due to shielded cables, cause phase shifts. With a quadrature measurement, we can learn both the magnitude and phase of the signal, and thus avoid confusing a phase shift for a magnitude change.

The Fourier series representation of a signal has independent sine and cosine coefficients for each frequency—the sine and cosine for a particular frequency are actually orthogonal basis functions. With our synchronous measurements, we are interested in just one frequency, but we must project onto both the sine and cosine basis functions associated with this frequency if we do not know the phase in advance.

In a quadrature measurement, we do precisely this. In addition to taking the inner product of the received signal with the original transmitted signal (which may be thought of as the cosine), we also take the inner product with the sine, i.e., the original signal shifted in phase by $\pi/2$. If the amplitude of the cosine component of the received signal is I and the amplitude of the sine component is Q, then the two measured coefficients I and Q, may be viewed a cartesian representation of the phase and magnitude of the received signal. The magnitude is therefore $(I^2 + Q^2)^{\frac{1}{2}}$, and the phase is $\arctan \frac{Q}{I}$.

A quadrature measurement can be implemented in analog hardware, using an additional analog multiplier and low pass filter, or in software, by applying additional processing steps. We will explain software implementation of quadrature demodulation later in this chapter.

1.4.3 Variants of synchronous detection

The description of synchronous demodulation given in section 1.4.1 made no reference to sinusoids. Our only assumption was that the modulating and demodulating functions are identical (up to a scale factor). In one useful variant of synchronous demodulation, square waves are used instead of sinusoids. It is simple to generate the transmit square wave using digital logic, and the analog multiplier is commonly replaced by simpler hardware: an SPDT switch alternately connects the inverted and then the non-inverted copy of a signal to the final integrator. The inverter multiplies the signal by -1, and the non-inverted signal is effectively multiplied by +1, so switching between the inverted and non-inverted copy of the signal is equivalent to multiplication by a square wave, yet the hardware is simpler, since the special case of multiplying by a square wave is easier to implement in hardware than multiplying by a sinusoid.

From an abstract view, it makes no difference whatsoever whether a sine or square wave is used. In practice there could be performance differences, because the square wave spreads some of the signal energy to higher harmonics of the fundamental square wave frequency. Typically there is less noise at higher frequencies, so in fact the square wave may yield slightly better SNR performance. Also, given the same maximum amplitude constraint, the total power in the square wave is higher than the power in one sinusoid with the same maximum amplitude.

Approximate versions

Sinusoid and square wave It is also possible to relax the requirement that the modulating and demodulating carriers be identical. The filter performance will degrade somewhat, but often other gains will offset the loss due to the "misalignment" of the modulating and

demodulating carriers. For example, the School of Fish, described in chapter 3, uses a sinusoidal transmit wave form and square wave demodulation. A School of Fish unit's PIC microcontroller generates a 5V square wave that drives a resonator to produce an 80V sinusoid. Because a sinusoid is transmitted, there is no signal energy at the higher harmonic "windows" that the square wave demodulation leaves open to noise. Fortunately, the amplitude of the harmonics falls off as $\frac{1}{k}$, where k is the index of the harmonic, so these windows are not open very wide, as we will now see quantitatively.

We will compare the SNR achieved when the received signal is demodulated with a sinusoid to the SNR that results from demodulating with a square wave. Let the square wave range from $+1$ to -1, with a period of 2π. Assume that the received signal r is a sinusoid with amplitude a plus broadband noise: $r = acos(w_0 t) + N(t)$. We will consider one period, from 0 to 2π. Let ϕ_n be a cosine of frequency nw_0 normalized on the interval 0 to 2π: $\phi_n = \frac{1}{\sqrt{\pi}} cos(nw_0 t)$ for $n > 0$, and $\phi_n = \frac{1}{\sqrt{2\pi}}$ for $n = 0$. With this definition, $\int_0^{2\pi} \phi_n(t)dt = 1$. The representation of the square wave $s(t)$ in terms of these basis functions is found by projecting each basis function onto the square wave:

$$s(t) = \sum_{n=0}^{\infty} < s, \phi_n > |\phi_n >= \frac{4}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{1}{2n+1} |\phi_n >$$

In this discussion, I will assume that the transmit carrier is $n = 1$. In both cases, assume that the received signal is the same, $a\phi_1$. The constant a represents the attenuated version of the signal picked up at the receiver...a contains the information we are interested in measuring. For the purposes of calculating the SNR, assume that a is at its maximum value. To find the decrease in SNR caused by the additional harmonic windows of the square wave, we will assume that the amplitude of the cosine used to demodulate in the "correct" case has exactly the same amplitude as the fundamental of the square wave, namely $\frac{4}{\sqrt{\pi}}$. Finally, we will denote the amplitude of the noise in one orthogonal component of the spectrum $< N >$. The signal power for the cosine demodulation scheme is $| < a\phi_1, \frac{4}{\sqrt{\pi}}\phi_1 > |^2 = \frac{16a^2}{\pi}$. The noise power in this case is $| < N, \frac{4}{\sqrt{\pi}}\phi_1 > |^2 = \frac{16}{\pi} < N >^2$. Thus the SNR is $\frac{a^2}{<N>^2}$. The signal in the square wave demodulation case is the same because we had chosen the amplitude of the demodulating cosine in the other scheme to be the same as the fundamental of the square wave in this one. The noise for this scheme is $| < N, s > |^2 = 2\pi < N >^2$, since the total power of the square wave is 2π. So the SNR is $\frac{8a^2}{\pi^2 <N>^2}$. Thus demodulating a sinusoid with a square wave decreases the SNR by a factor $\frac{8}{\pi^2} \approx .81$. We lose about 20 percent of our SNR doing this, so if other gains offset this loss, it can be worthwhile to demodulate a cosine with a square wave.

Another, perhaps more relevant example is to compare the case of demodulating a square wave with a square wave, and a cosine with a square wave. The signal is $| < as, s > |^2 = 2\pi a^2$. The noise is again $| < N, s > |^2 = 2\pi < N >^2$, so the signal to noise is $\frac{a^2}{<N>^2}$. For the cosine case, we will assume that the received signal is $a\frac{4}{\sqrt{\pi}}\phi_1$, so that its amplitude matches that of the square wave's first harmonic. The signal in this case is $| < a\frac{4}{\sqrt{\pi}}\phi_1, s > |^2 = (a\frac{16}{\pi})^2$. The noise is unchanged, so the SNR is $\frac{64}{\pi^4}\frac{a^2}{<N>^2} = .66\frac{a^2}{<N>^2}$. Thus in this example, we lose about 1/3 of the signal. Since a square wave is trivial to generate and gives better SNR for the same amplitude, one might wonder why we would consider transmitting a sinusoid instead of a square wave. The answer is that by making the transmitter resonant, the square wave generated by the digital circuitry gets transformed into a sinusoid with an amplitude greater than the original by a factor Q. Since switching

from square wave to sinusoidal transmit decreases our SNR by $\frac{2}{3}$, then as long as the Q of the resonator is greater than $\frac{3}{2}$, we will get a net increase in SNR by transmitting with a sinusoid rather than a square wave.

Sampled By demodulating with a train of delta functions of alternating sign, we can reduce the hardware complexity even further, venturing into the realm of "software radio." This demodulation scheme can be implemented using just a microcontroller with an analog to digital converter—the inverter and integrator become software operations on the samples acquired by the ADC. Chapters 2 and 8 discuss practical implementations of the basic signal processing ideas discussed in this section.

Chapter 2

Synchronous Undersampling and the LazyFish

This chapter describes the LazyFish, a board I designed that implements 8 channels of Electric Field Sensing in a very small footprint. Its small footprint is made possible by my Synchronous Undersampling technique, which is explained in this chapter. Before doing so, I will review some other implementations of electric field sensing, starting with the Theremin.

2.1 Background

2.1.1 Theremin

Figure 2-1 shows a schematic of Clara Rockmore's Theremin, drawn by Bob Moog in 1989. Here is a "back of the envelope" analysis of the sensing and pitch synthesis. Changes in hand proximity affect the capacitance in one LC resonant circuit (the one on the right side of the figure, with the labeled pitch antenna), changing its resonant frequency. Though not all component values are provided in this diagram, if we assume $L = 38\text{uH}$ and $C = 663\text{pF}$, the resonant frequency $f = \frac{1}{2\pi\sqrt{LC}} = 1\text{MHz}$ and $Q = \sqrt{\frac{L}{C}} = 240$. The sound is synthesized by demodulating this hand "detunable" signal f_1 with a fixed frequency carrier f_2. The fixed carrier is generated by the resonator on the left side of the schematic, and the mixing occurs in the vacuum tube in the center of the figure. The audible output frequency is the difference $f_1 - f_2$. With our estimated component values, producing a difference frequency of 20KHz (the upper frequency limit on human hearing) requires a change in capacitance of 27pF, which is somewhat large but not unreasonable estimate of how much a hand could change the capacitance. These figures are just ballpark estimates, but they are sufficient to illustrate the basic idea that a change in capacitance that could be produced by the motion of a hand near an electrode causes the difference frequency to span the entire human audible frequency range.

2.1.2 Classic Fish

Neil Gershenfeld built the "small box" shown at the top of figure 2-2, which was a hand-wired, all analog implementation of Electric Field Sensing that required an external analog-to-digital converter board in a PC. The Classic Fish, shown below the "small box," was a

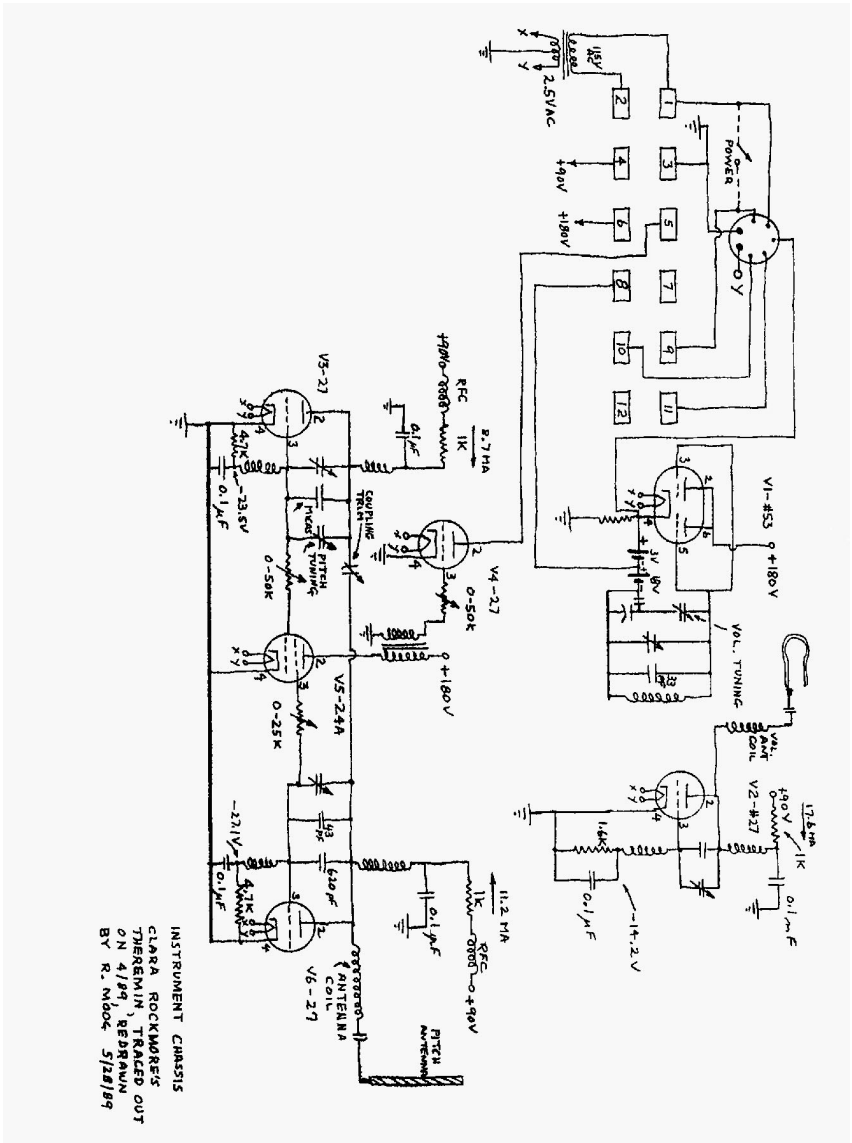

Figure 2-1: A schematic of Clara Rockmore's Theremin, drawn by Bob Moog.

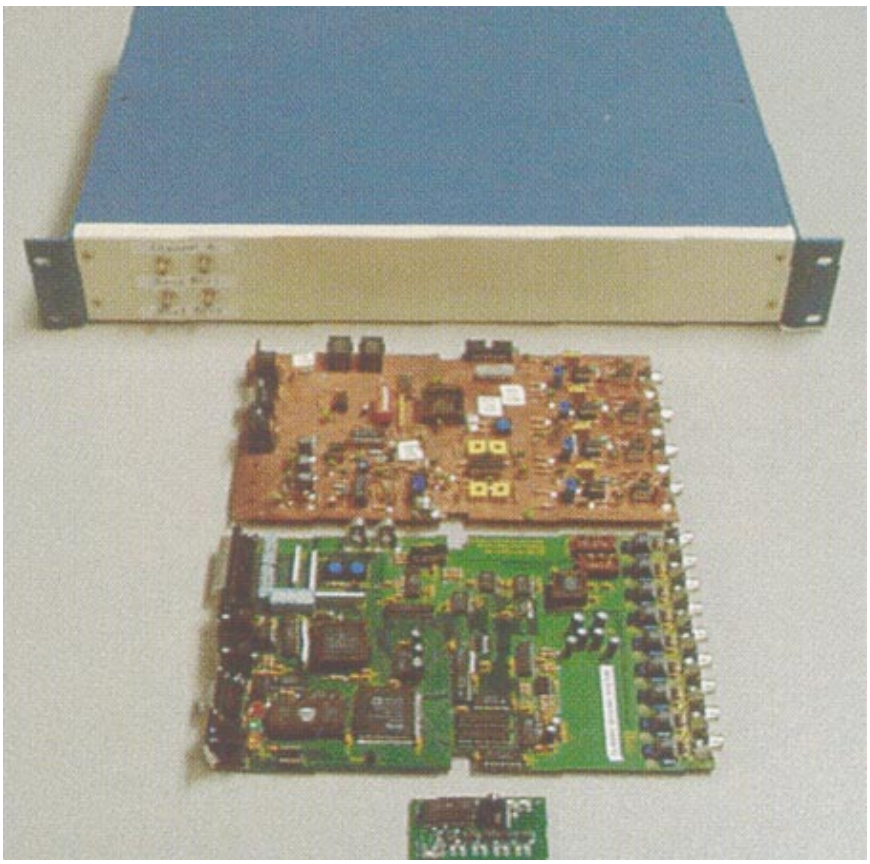

Figure 2-2: The LazyFish family tree. From bottom to top: the LazyFish, the SmartFish, the Classic Fish, and the "Small Box."

printed circuit board that had four analog channels of Electric Field Sensing, plus a Motorola 68HC11 8 bit microcontroller for analog to digital conversion and serial communication to the host PC or MIDI synthesizer. Following Neil's initial guidelines, Joe Paradiso designed the analog portion of the Classic Fish, Tom Zimmerman did the digital portion, and I wrote the board's firmware.

The Classic Fish had one transmitter, tunable (via a potentiometer) from 10kHz to 100kHz. Another pot controls transmit amplitude. Each of the four receive channels consists of a transimpedance gain stage, analog multiplier, and low pass filter gain stage. Four phase shifting circuits provide independently shifted versions of the transmitted signal to the multipliers, so that the received signal can be synchronously demodulated. The phase for each receive channel is hand adjusted with four potentiometers, to compensate for phase changes due to cable capacitance. The DC offset and gain of the final stages is adjustable via eight additional pots, to match the analog output to the useful working range of the ADC.

Below the Classic Fish in figure 2-2 is the Smart Fish, which was a brute force attempt to do the demodulation in software. Depending on the version, it had one or two transmitters and eight or nine programmable gain receive channels, a fast ADC, fast DSP, as well as a 68HC11. The Smart Fish was plagued by problems and never worked well.

The sensing portion of the LazyFish is shown at the bottom of the figure.

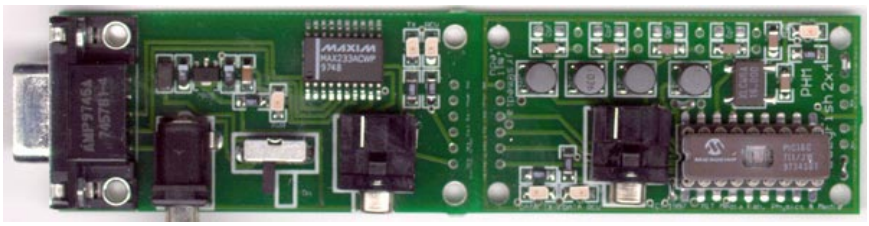

Figure 2-3: A closeup of the LazyFish.

2.2 Synchronous Undersampling

This section explains Synchronous undersampling, the measurement technique used by the LazyFish. First I will explain ordinary synchronous detection, the method used by the Classic Fish.

2.2.1 Synchronous detection

Figure 2-4 illustrates traditional synchronous detection. A 100kHz carrier voltage is applied to the transmit electrode. A 100kHz current whose magnitude depends on the hand position is induced on the receive electrode. This signal is amplified in a transimpedance gain stage, and then mixed down to DC by an analog multiplier (with access to the original transmitted signal) followed by a low pass filter. As shown in figure 2-4, multiplying the received signal by the original transmitted signal produces sidebands at $+2f$ and $-2f$ as well as a DC value. The low pass filter eliminates these sidebands, and the amplitude of the remaining DC signal contains the desired information about the hand proximity.

Quadrature detection

If the phase of the received signal is unknown, it can additionally be demodulated with a $\frac{\pi}{2}$ phase shifted version of the transmitted signal. These two demodulated components (called the in phase and quadrature components) are a cartesian representation of the magnitude and phase of the received signal.

2.2.2 Synchronous sampling

Figure 2-5 illustrates synchronous sampling. If the carrier frequency is f, then the signal after multiplication (but before the low pass filter) has components at $+2f$ and $-2f$. By Nyquist's theorem, it should be sufficient to sample this signal at $4f$.[1]

The sampling operation can be viewed as multiplication with a train of delta functions. Since multiplication is commutative, we can also regard the received signal to have been sampled before the demodulation operation occurs. If the amplitude of the demodulating (former) cosine is 1, then its sampled version consists of a train of $+1$ and -1 height delta functions. If the low pass filter is replaced by a zero order hold (that is, if we simply add all the signal amplitude), then the demodulation operation becomes simply addition

[1] It could be argued that because the original signal can be reconstructed from a train of samples at $2f$ it should not be necessary to sample at $4f$. In fact, the $4f$ figure is simply a very natural sampling rate to consider in the context of quadrature demodulation; the point being developed in this chapter is that the signal can actually be sampled at much less than $2f$.

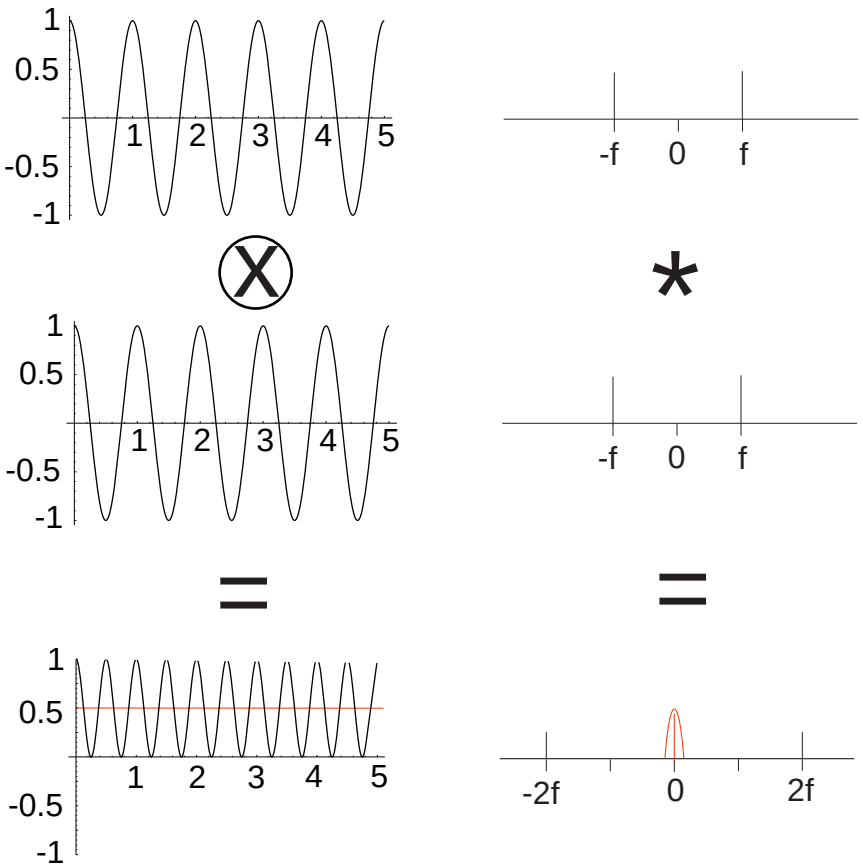

Figure 2-4: Synchronous detection.

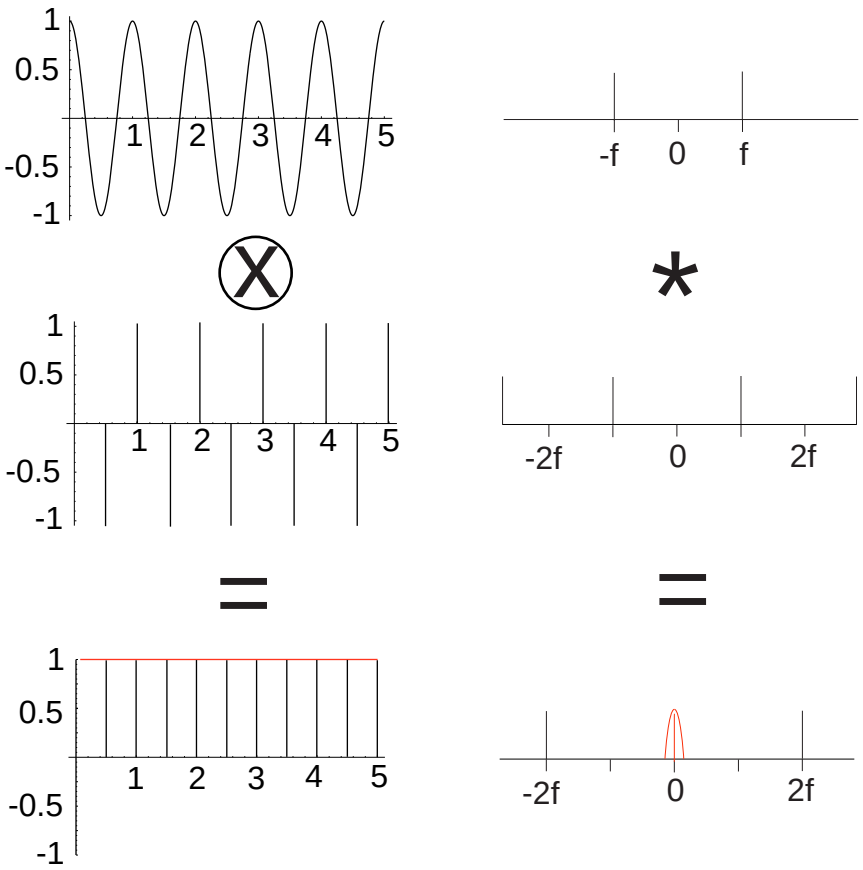

Figure 2-5: Sampled synchronous detection (synchronous sampling).

and subtraction of samples of the received signal. The analog multiplier and low pass filter have been eliminated. To implement quadrature detection, we perform the same sequence of operations on samples spaced 90 degrees apart, keeping two separate accumulation variables, one for the in phase channel and one for the quadrature channel. Note that this technique leaves harmonic windows open to noise at multiples of $2f$. For this reason, it is desirable to include a bandpass filter centered on f in the front end gain stage, although this feature is not present in the (original) LazyFish.

2.2.3 Undersampling

Unfortunately, the ADC on a typical microcontroller such as the PIC16C711 is quite slow. On this microcontroller, the maximum sampling rate turns out to be about 40kHz. If we require our sampling frequency to be $4f$, where f is the carrier frequency, then our carrier is limited to 10kHz. As explained in section 1, the received signal $I = 2\pi fCV$ is proportional to frequency. Thus signal to noise is linear in the frequency. But there is an additional advantage to moving to higher frequencies. As will be explained below, in practice the V term (transmit amplitude) also scales linearly with f, so actually the signal to noise scales at least quadratically with frequency.

The transmitter is a series LC circuit with resonance at $f = \frac{1}{2\pi\sqrt{LC}}$. When it is driven on resonance, its steady state amplitude is given by the product of the driving amplitude

and the resonator's quality factor $Q = \frac{1}{R}\sqrt{\frac{L}{C}}$. Thus V, the final transmitted amplitude, is given by $V = vQ$, where v is the amplitude of the PIC's driving signal. So, substituting in the expression for Q, the final received current $I = 2\pi f C v Q$.

In practice, L is a limited resource: 2.5mH was the largest inductance commonly available in a surface mount package at the time the LazyFish was designed. L should be as large as possible in order to maximize the Q. If L is a constant (2.5 mH), then the only way to increase the Q is to go to a higher frequency. By manipulating the expressions for f and Q, we can write Q in terms of f:

$$Q = \frac{2\pi L}{R} f \tag{2.1}$$

Because L is constant, the frequency f can be adjusted only by changing C, not L. With L constant, it is clear from equation 2.1 that Q is linearly proportional to frequency.

Now we can rewrite the expression for the received current for the last time

$$I = (2\pi f)^2 LC \frac{v}{R}$$

It is clear from this expression that signal strength scales quadratically with frequency. Thus if we could find a way to move our synchronous sampling scheme from 10kHz to 100kHz, it would increase the received signal strength by a factor of at least 100.

Synchronous undersampling allows us to do precisely this. Although the ADC on the PIC16C711 is slow to convert voltages to bits, the ADC has a built in sample and hold with a much shorter aperture time. Furthermore, this sampling aperture can be placed very precisely in time. So while the actual conversion operation apparently limits us to sampling 10kHz signals, in reality the PIC can sample much higher frequency signals as long as they are repetitive. This makes sense if we consider the fact that a short aperture that can placed precisely in time corresponds to a high input bandwidth.

Figure 2-6 shows the basic idea of synchronous undersampling. The PIC generates a squarewave burst that causes the resonator to ring up. Then it samples at a precisely known time—call it phase 0—and waits for the value to be converted. Then it starts over, generating a new burst, this time waiting a bit longer before sampling. In this way, 0 degree, 90 degree, 180 degree, and 270 degree samples of a 100kHz signal can be collected.

The spikes in figure 2-6 indicate roughly when the sample and hold was open. Figure 2-7 through 2-10 are "closeups" of figure 2-6. These figures show the 0 degree sample S_0, as well as S_{90}, S_{180}, and S_{270}. Note that some additional immunity to noise with the sampling periodicity could be gained by pseudorandomly varying the time between samples. Another strategy worth considering would be pseudorandomly varying the order in which S_0, S_{90}, S_{180}, and S_{270} are taken.

As the samples are collected, the demodulation operation is performed by updating the inphase channel accumulator using $I' = I + S_0 - S_{180}$, where I' is the new value and I is the old value. Similarly, the quadrature accumulator $Q' = Q + S_{90} - S_{270}$. The PIC's ADC returns 8 bit values, and the I and Q accumulators must be 16 bit variables because multiple samples are added together. The number of samples integrated is controlled in software, which allows signal to noise to be traded off with update rate. The LazyFish firmware computes an approximate magnitude from the I and Q components. The true value that should be returned is $\sqrt{I^2 + Q^2}$; the approximate value returned by the LazyFish is $|I| + |Q|$. As explained in chapter 3, the error of this approximation depends on the phase (0 error at 0

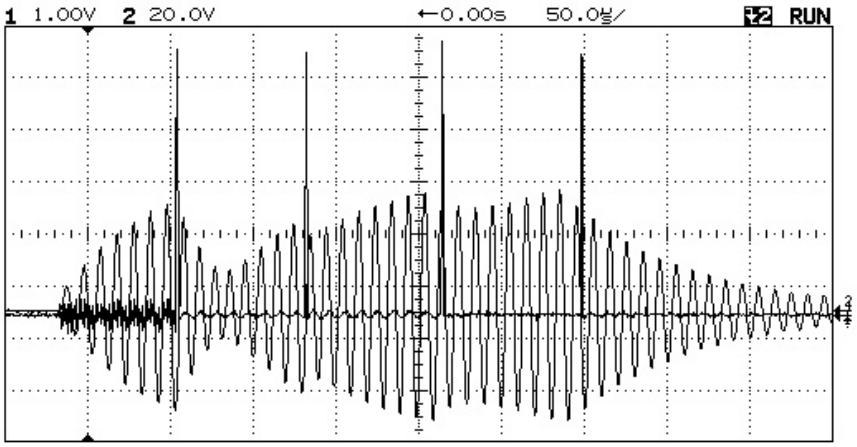

Figure 2-6: All four samples.

or 90 degrees, maximum error at 45 degrees). Since (unlike the School of Fish) the LazyFish is doing straightforward homodyne detection, with the receiver and transmitter absolutely synchronous, and since the phase offset between transmitter and receiver is constant in most applications, the approximation corresponds to a constant scale factor, and causes no problems.

2.2.4 Controlling gain by adjusting TX burst length

The LazyFish provides software programmable gain control, by making use of the resonator's transient response. This feature is important because if the front end clips (due to a received signal that is too strong), the synchronous demodulation operation does not work correctly. The obvious but more complex approach would be to use programmable gain in the receiver. The LazyFish controls the transmit gain instead, leaving the receive gain fixed. This approach turns out to require simpler hardware.

To adjust the transmit gain, the LazyFish firmware controls the length (number of periods) of the square wave burst used to excite the resonator. By using a shorter burst, it can avoid exciting the resonator all the way. Figures 2-11 through 2-18 show progressively shorter bursts (starting with 20 and working down to 1). In a "continuous wave" (rather than pulsed) application, one can also imagine controlling the frequency or duty cycle of the square wave drive. In our "pulsed" situation, controlling the resonator amplitude by controlling the duration of the square wave burst is the simplest to implement.

Quenching the resonator as quickly as possible is also important to achieving a high update rate. After the sample has been taken, the output pin driving the resonator is put into a high impedance state, which causes the resonator to quench in just a few cycles. If the output pin was left at a fixed voltage, the resonator would continue to ring for much longer (20 cycles) because of its high Q.

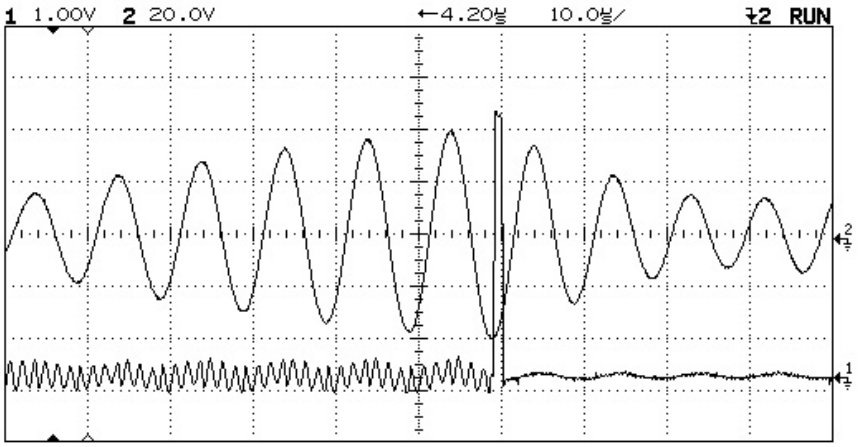

Figure 2-7: 0 degrees.

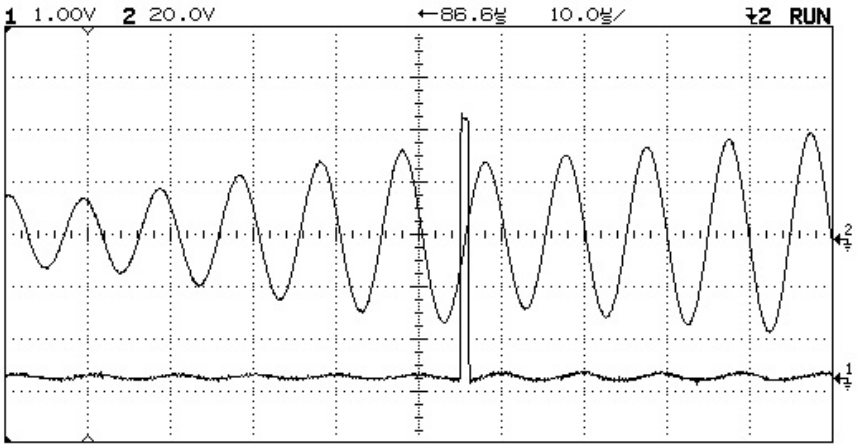

Figure 2-8: 90 degrees.

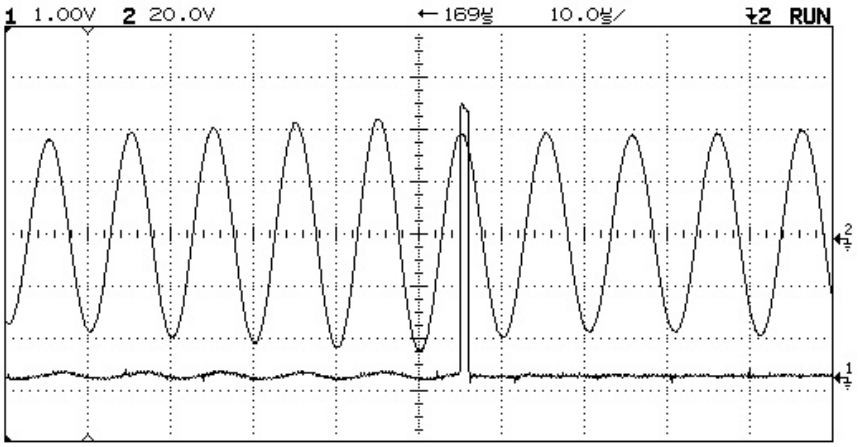

Figure 2-9: 180 degrees.

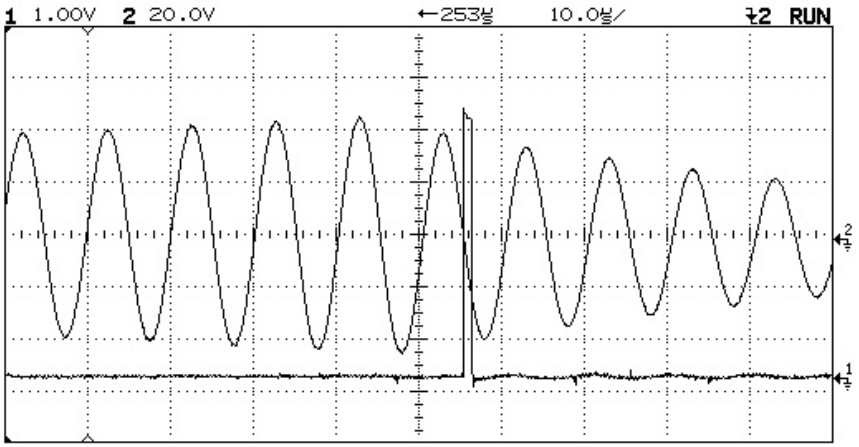

Figure 2-10: 270 degrees.

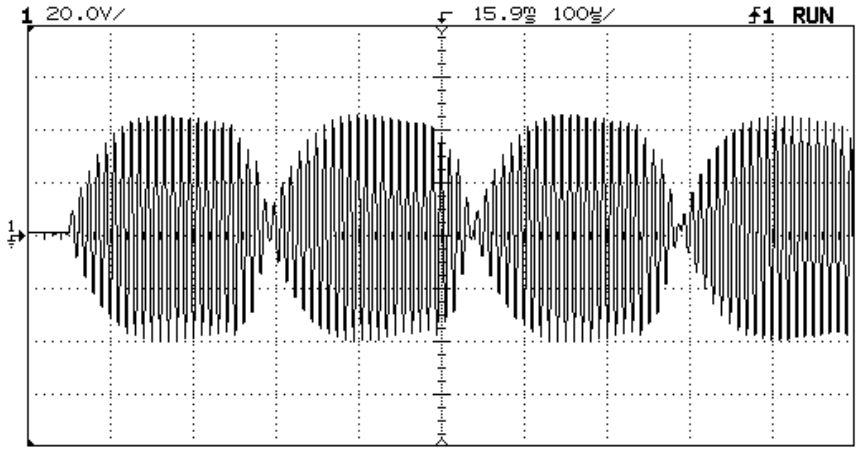

Figure 2-11: Burst length: 20.

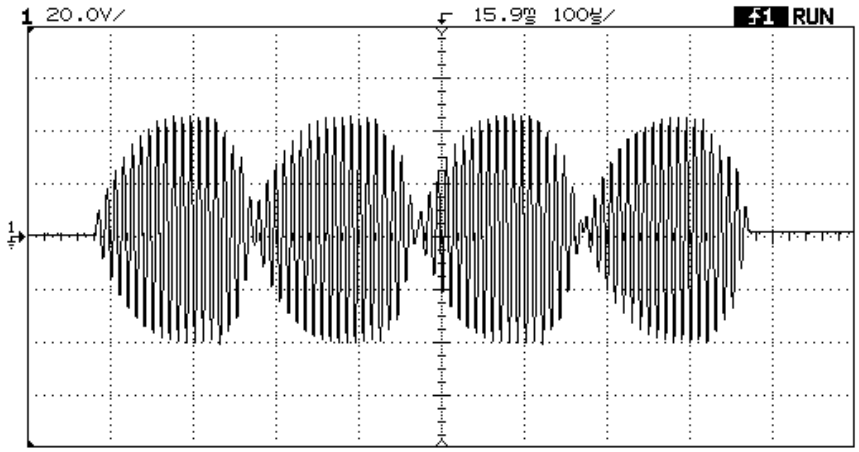

Figure 2-12: Burst length: 15.

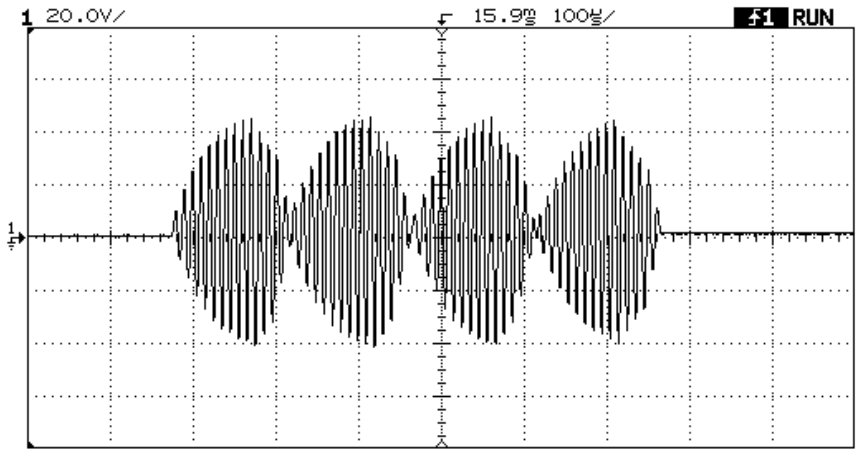

Figure 2-13: Burst length: 10.

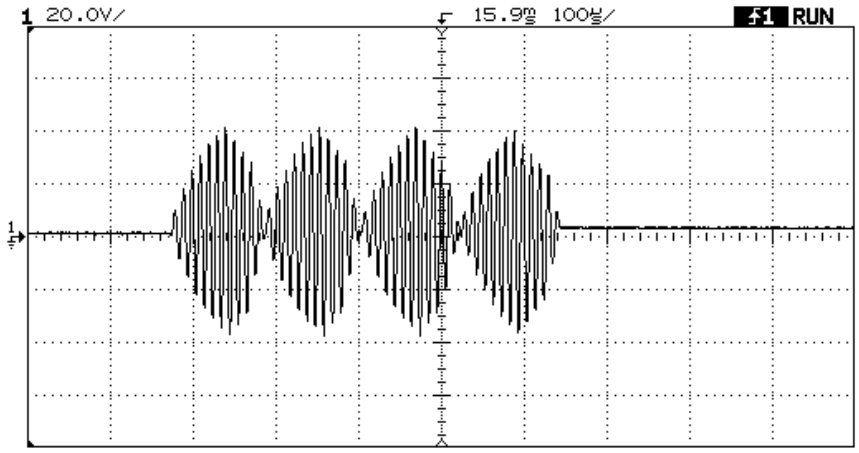

Figure 2-14: Burst length: 7.

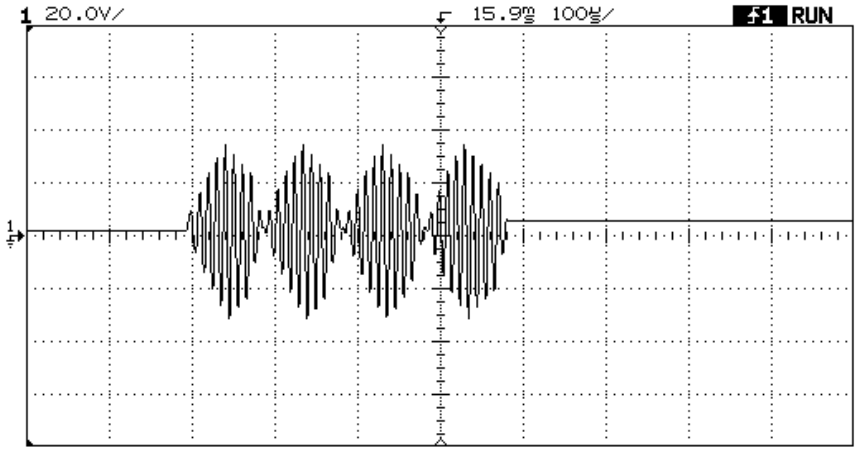

Figure 2-15: Burst length: 5.

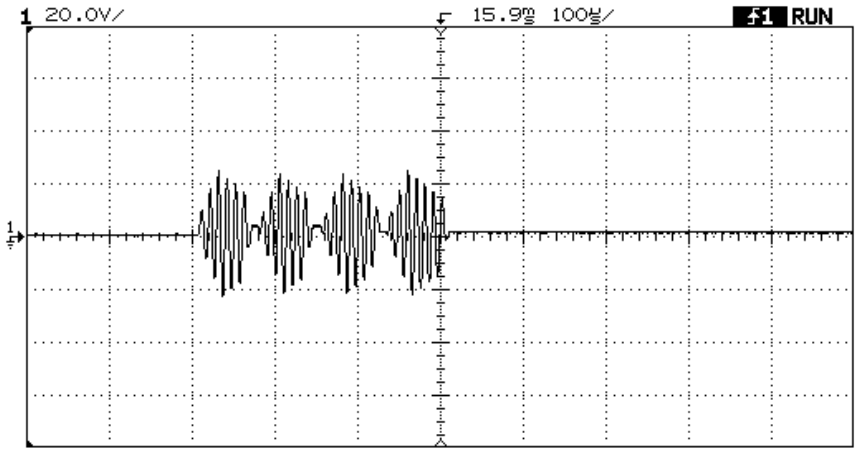

Figure 2-16: Burst length: 3.

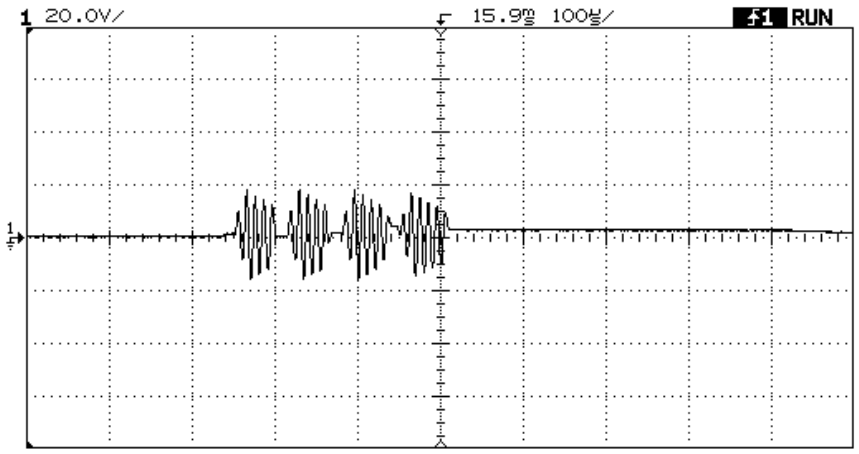

Figure 2-17: Burst length: 2.

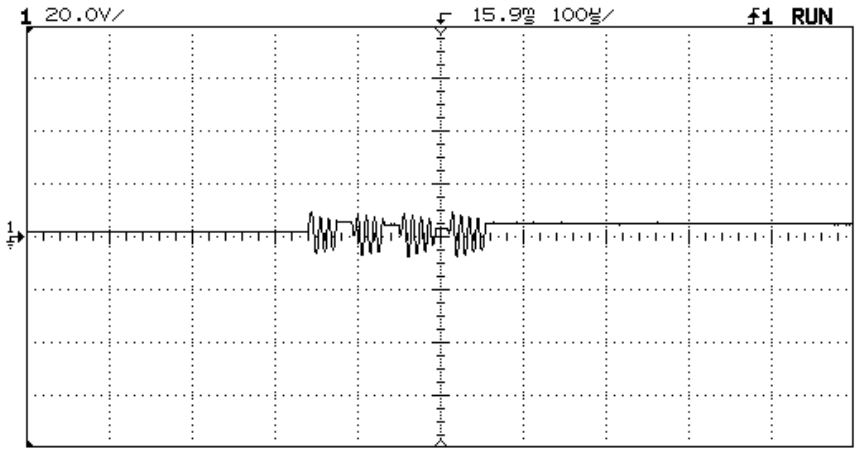

Figure 2-18: Burst length: 1.

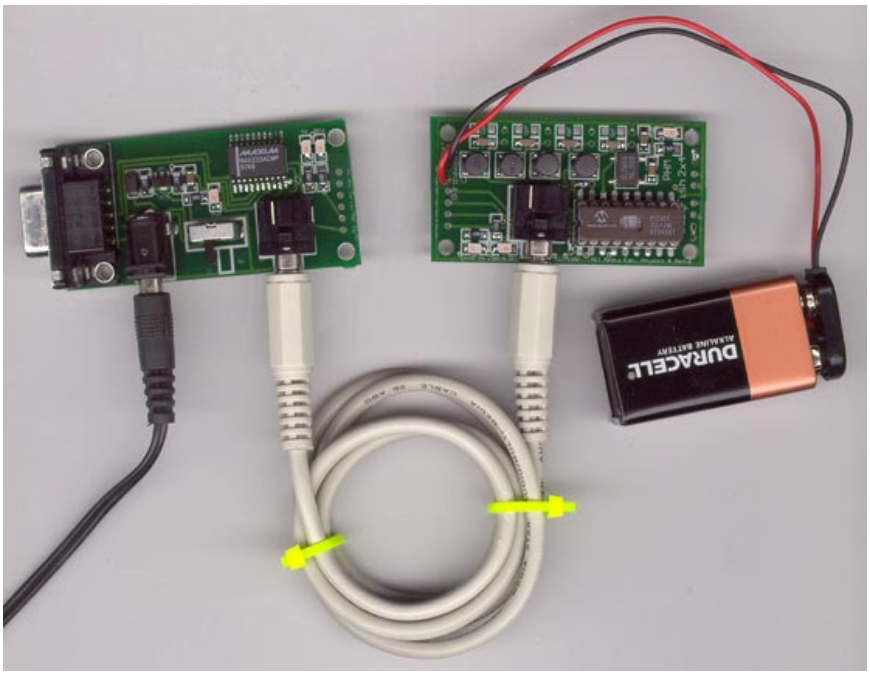

Figure 2-19: Remote Sensing.

2.3 The LazyFish

Because of the synchronous undersampling technique described in the first chapter, the LazyFish is able to implement 8 channels of electric field sensing using surprisingly little hardware. It has 4 resonant transmit channels and 2 receive front ends. All demodulation is performed in software on a PIC16C711 microcontroller, which has a built in analog-to-digital converter.

The LazyFish board consists of two initially integrated portions that may be split in two: the sensing portion and an RS-232 digital communication interface. When the LazyFish is used as a computer input device, it would typically be deployed in the integrated configuration, shown at the top of this page. When the LazyFish is built in to a handheld device, or connected to an RF transceiver or other device that uses TTL voltage levels, the sensing portion would be used without the communications portion. The sensing and RS-232 portions each have a stereo audio jack that can be used to connect the two halves using an ordinary stereo audio cable. The RS-232 interface portion can be connected permanently to the computer, and handheld LazyFish devices can be connected to the computer temporarily for debugging purposes through the stereo cable and RS-232 interface. This "remote sensor" configuration is shown below. For applications in which a smaller footprint RS-232 capable device is required, the stacked configuration, in which the two sections are mounted together, can be used. This configuration is also pictured below. Finally, the RS-232 interface may be used by itself to connect other TTL devices to a computer.

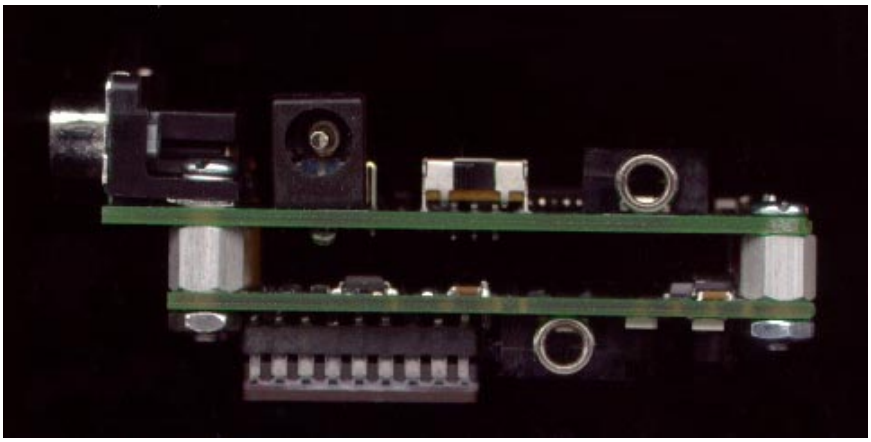

Figure 2-20: Stacked configuration.

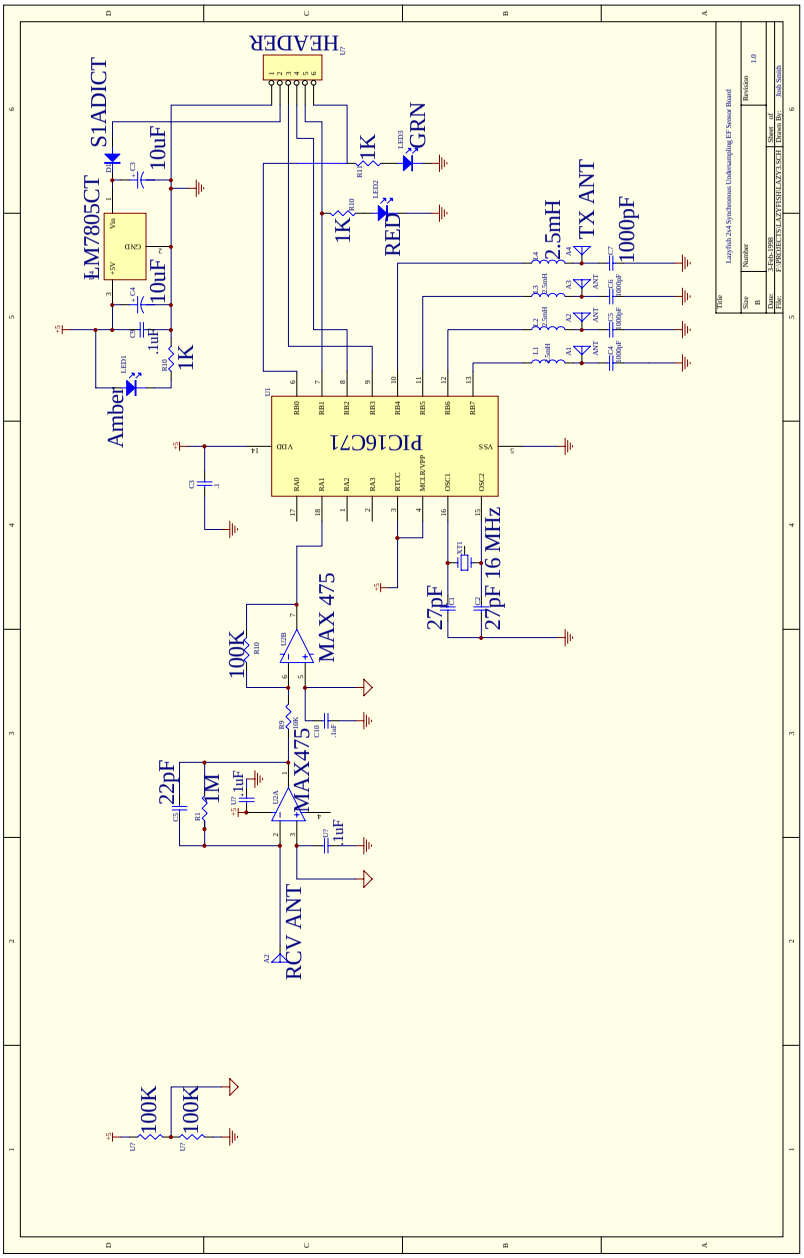

Figure 2-21: LazyFish Schematic.

Chapter 3

The School of Fish

3.1 Introduction and Motivation

The School of Fish is a network of "intelligent electrodes" for electric field sensing. Each school of fish board contains all the circuitry needed to drive an electrode as either a transmitter or receiver. Sensing arrays of arbitrary size can be assembled by connecting as many units as are required to solve the problem at hand. For a 3d mouse, one might use 7-10 units; for a simple proximity sensor, a single unit with an additional electrode would suffice; in an imaging application, one might use as many 256 units. Because the units can be used as both transmitters and receivers, one can make $O(n^2)$ different measurements with n units, by making all transmit-receive pairs of measurements.

The units communicate over an RS-485 serial bus. Only power, ground, and the digital communication signals are shared between the units. Sensitive analog signals are localized within each board. As long as the sensing electrode is located near the electronics—the typical configuration—there is no need for shielded cable.

As well as being a practical sensing system, the school of fish units were designed to be a platform for exploring ID-assignment strategies. To that end, a hardware noise circuit for breaking symmetry among otherwise identical units and a Dallas Semiconductor Silicon Serial number are included on the board.

The School of Fish eliminates the many analog adjustments that made the original Fish electric field sensing board difficult to use. The original fish had one fixed transmit channel and four fixed receive channels. It required 14 potentiometers to accomplish this. The School of Fish units and the SmartFish have no pots. The Smart Fish board represents the "proper" approach to eliminating these adjustments. In addition to a microcontroller, it has a fast DSP, programmable gain amplifiers, and other (relatively) specialized and costly components. The school of fish embodies a low cost, microcontroller-software-centric (and less general) solution to the problem of eliminating hardware adjustments. The SmartFish is capable of operating at any frequency from DC up to megahertz, while the school of fish units cannot operate above about 100kHz. But school of fish units consist merely of CMOS switches and opamps under the minute supervision of a microcontroller. Adjustments to parameters such as gain can be controlled by changing parameters of the code running on the microcontroller.

The School of Fish differs from both the original Fish and the SmartFish in one crucial way: a School of Fish transmitter and receiver are not located on the same board, which means that they do not share a clock, and therefore are not truly synchronous. I will

describe my software solution to this problem in the section below on firmware.

3.2 Description of Hardware

The heart of a School of Fish unit is a PIC microcontroller. Not only does it manage communications and handle analog-to-digital conversion, which is common in so-called embedded data acquisition systems, but it also controls the modulation and demodulation operations at the finest scale, which allows these to be adjusted very precisely in software.

The PIC receives commands—which might tell it to become a transmitter, or a receiver—from the RS-485 bus. When the unit is operating as a transmitter, the PIC originates the transmit waveform by toggling one of its pins. When it is operating as a receiver, the PIC controls the synchronous demodulation process, by toggling a different pin. Because the behavior of these pins is software controlled, we are not confined to simple periodic signals. We can for example generate a set of bursts of some frequency and duration, separated by gaps of variable duration, or even generate pseudo-random sequence for a spread spectrum scheme.

The School of Fish typically operates at 75 kHz. A high frequency is desirable from the point of view of signal-to-noise, since the strength of the capacitive coupling between the transmitter and receiver is proportional to frequency. With the 16MHz crystals used on the school of fish boards, the PIC 16C71 runs at 1 MHz, so we could in principle operate at frequencies higher than 75kHz. However, running at this relatively low frequency gives us flexibility...it is easier to hide the fact that the waveforms are being generated in software if the shortest timescale in the sensing process is somewhat longer than the shortest timescale accessible to the microcontroller, that is, one instruction time. For example, one straightforward way to generate more than 256 periods of a transmit waveform is to use two nested loops. Inside the inner loop, the processor toggles the output pin. Whenever the inner loop terminates, an extra couple of processor cycles will be required to execute the outer loop one time before the processor can resume toggling the pin. Unless the time required to manage the outer loop is small compared to the period of the waveform being generated, the waveform will be badly distorted.

3.3 School of Fish Firmware

The transmission and demodulation operations are driven by the microcontroller, so its software is actually a crucial part of the hardware. When transmitting, the PIC generates a number of 75kHz square wave bursts. The number of bursts and the length of each burst (number of periods) are parameters that can be set by the host or by one of the other units over the RS-485 link. The effective DC gain can be adjusted by controlling the length of these bursts. The original fish had a gain and offset pot for each channel to align the DC values with the ADC's range. Because the school of fish adds the results of multiple measurements in software, its dynamic range is not limited to the 8 bits provided by the ADC, so it is not crucial to scale the demodulated values to fit into a particular narrow range.

However, if single demodulated values exceed the ADC's range—say because the transmitter and receiver are very close to one another— the values may be brought back into the linear regime by shortening the sensing bursts. Shortening the bursts decreases signal to noise (though presumably the signal must be strong, or there would be no saturation

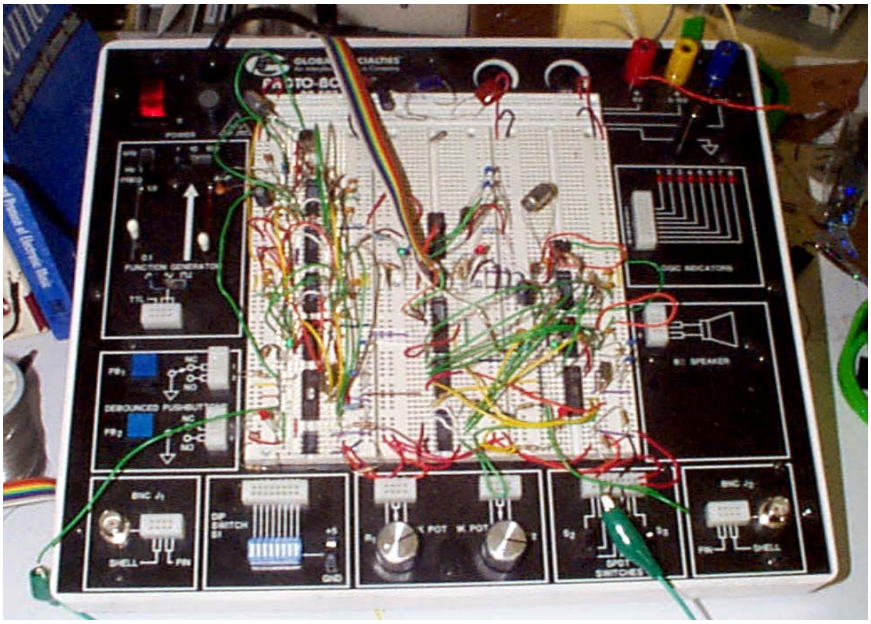

Figure 3-1: Breadboard prototype school of fish unit.

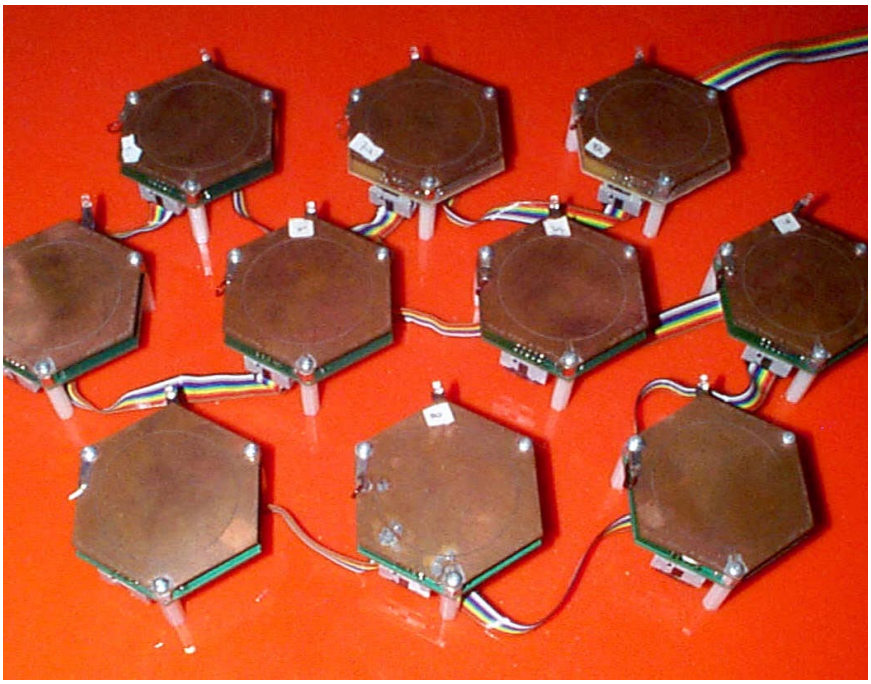

Figure 3-2: Array of 10 school of fish units.

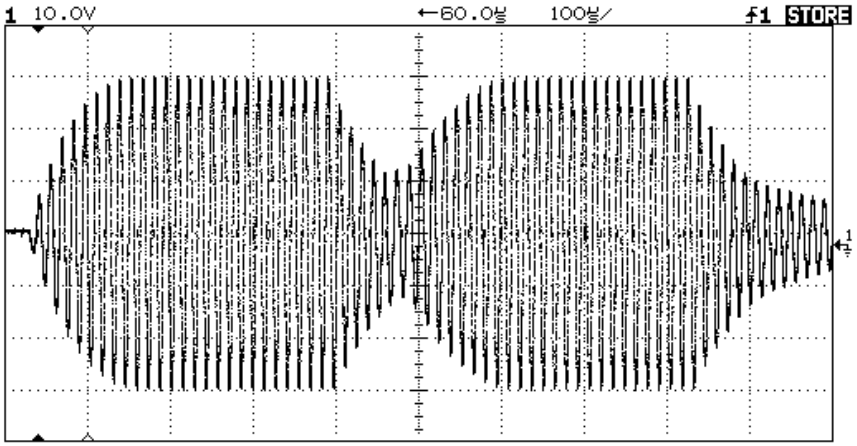

Figure 3-3: Two 26-cycle transmit bursts. Note the 60V swing achieved by the resonant tank. The stronger the transmitted signal, the less gain required on the receive side, which in turn translates into less amplifier induced noise. Smaller noise figures allow shorter integration times, and thus faster scans.

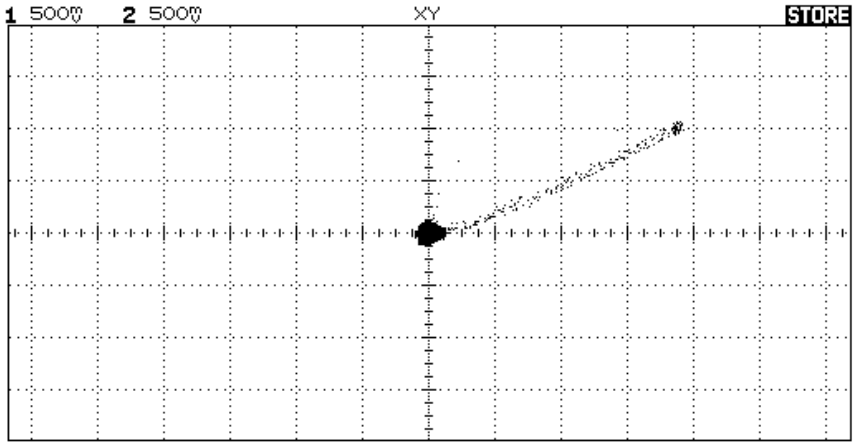

Figure 3-4: X axis: in phase demodulated value. Y axis: quadrature demodulated value. As the integration proceeds, the trace travels outward from the origin. When the measurement concludes, the integrators are reset and the trace returns to the origin. The length of this vector represents the magnitude of the signal detected by the receiver, and the angle of the trace gives the phase of the transmitted signal relative to the receiver's demodulation operation.

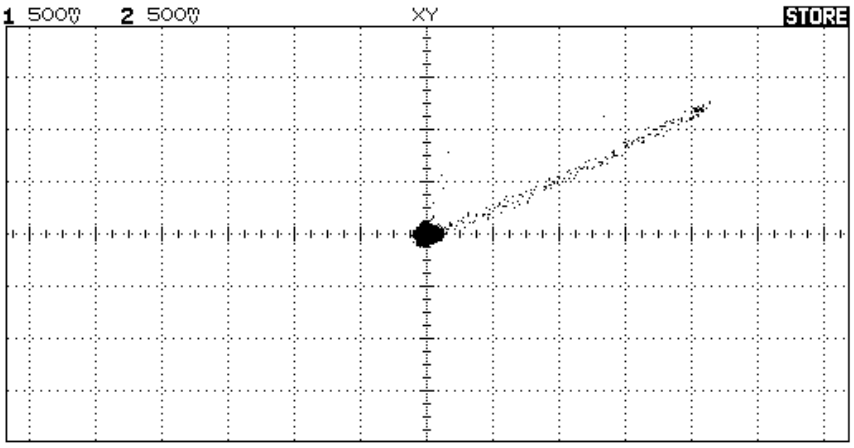

Figure 3-5: X axis: in phase demodulated value. Y axis: quadrature demodulated value. Note the phase difference (angle difference
between this signal and that of the previous figure. On the timescale separating the two measurements, the phase of the clock on the transmit unit is random with respect to the phase of the receiver's clock. Even when the host computer polls the units relatively frequently, the host's timescale is long enough that the phase of the transmitter and receiver jitter randomly from one measurement to the next. (This problem is compounded by the fact that if the computer is running the Windows operating system, the intervals between polls is typically irregular.)

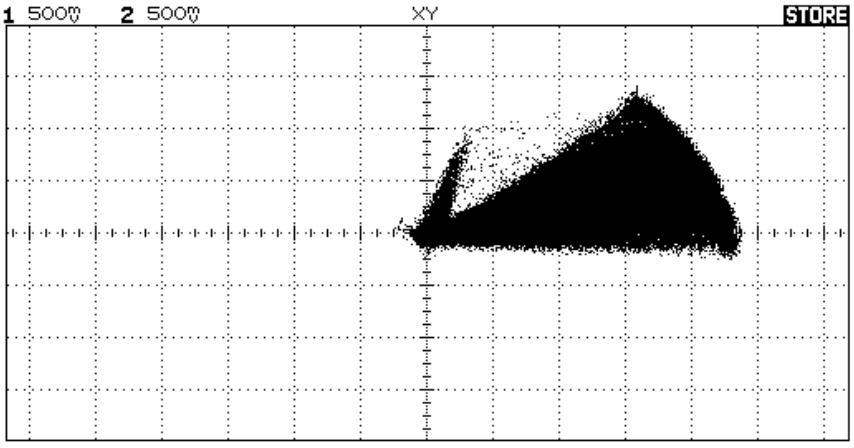

Figure 3-6: Many single shot measurements have been overlayed to show the extent of the phase jitter.

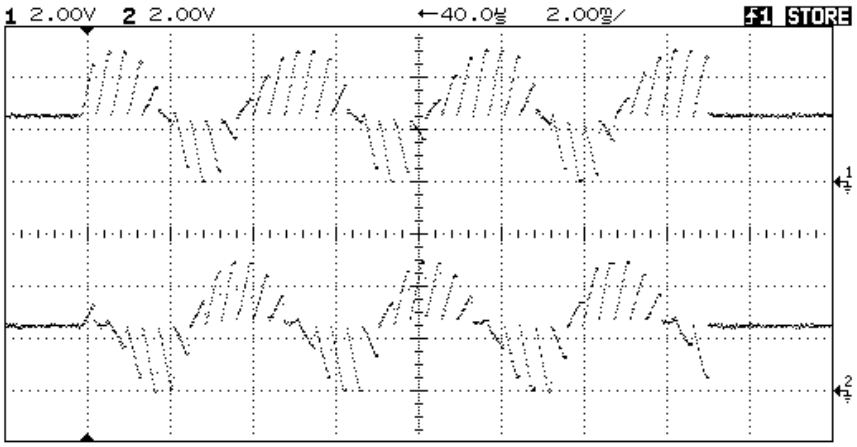

Figure 3-7: Top trace: output of integrator for in phase demodulation channel. The trace represents forty bursts. Each straight line segment is the demodulated value being integrated up. Bottom trace: output of integrator for quadrature demodulation channel. After each burst, the integrators are reset with a CMOS switch, so that the next measurement may be made. Also after each burst, the phase of the receiver is lagged slightly relative to the transmitter. This gives rise to the "helical" pattern.

problem), but signal to noise can be improved as much as desired by making multiple measurements and accumulating the results in software, using 16 bit math on the PIC. The purpose of the repeated bursts is to improve signal to noise and dynamic range. Even though the ADC only has 8 bits of dynamic range, by adding multiple measurements, the PIC can increase the effective dynamic range beyond what its ADC is capable of.

The square waves generated by the PIC raise and lower the non-inverting input of the front end opamp, and the opamp raises or lowers its output in order to make the inverting input follow the non-inverting input. In this mode, one of the diodes in the feedback network goes into conduction, so the opamp does not see the impedance of the feedback resistor and capacitor. The inverting input connects to the series inductor capacitor resonant tank, which rings up to a large (around 60V peak to peak) sine wave within the first few periods of each burst. If this tank were not present, the crossed diodes in the feedback network would not be required, because the opamp would be able to source enough current to drive the electrode even through the impedance of the feedback network. But the opamp is not able to drive the tank properly through the resistor and capacitor, so the crossed diodes were used.

In the present version of the software, (virtually) all the commands are ASCII, so that the School can be operated "by hand" from a terminal program. The School of Fish communications protocol is documented in Appendix B.

When receiving, a unit drives the in phase and quadrature switches for the number of periods in one burst, then uses two of its ADC inputs (A0 and A1) to sample the values in the in phase and quadrature integrators. After it has read the values, it uses the 4066 switches to reset the integrators.

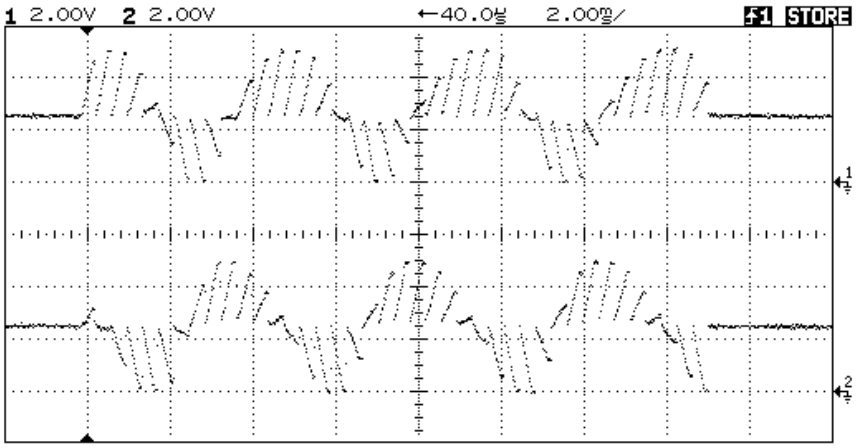

Figure 3-8: The same helical pattern as above, for a different measurement. Note that the global phase (the phase of the starting and ending arms of the helix) is different for the two figures. This is the randomization of transmitter phase relative to receiver phase that occurs between measurement commands issued by the host computer. One the timescale of a single measurement, the phase difference between the transmit clock and receive clock is essentially constant, which is why the helical structure is not distorted. There is essentially no unintentional phase drift on the time scale of the screen shot...there is only the intentional phase lag that we create each time the integrators are reset.

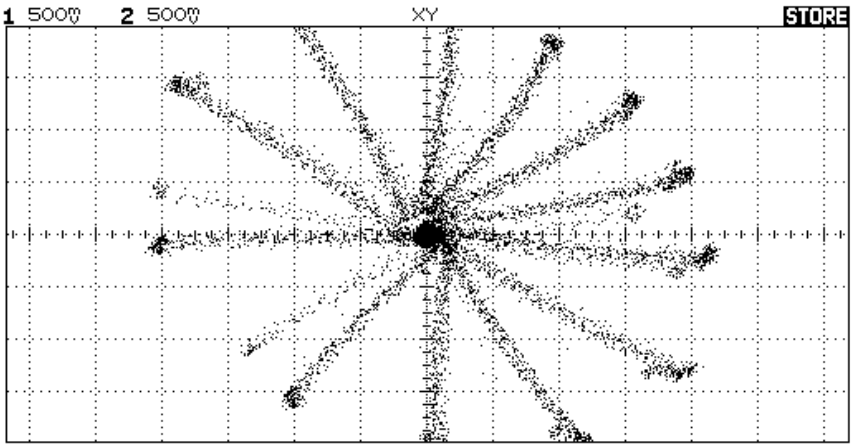

Figure 3-9: X axis: in phase demodulated value. Y axis: quadrature demodulated value. This is an XY plot of the same information that appears in the previous figure. This shows the receiver phase actively "precessing."

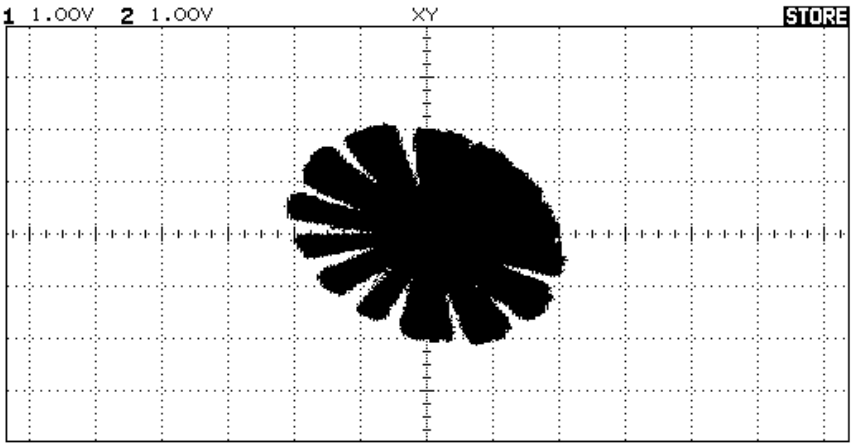

Figure 3-10: All the phases visited by receiver, obtained by overlaying many measurements.

The in phase and quadrature pairs are elements of a 2 dimensional vector space. It is easy to satisfy oneself that the in phase and quadrature components of the raw signal must be orthogonal to one another. As the relative phase between the transmitter and receiver is changed, consider the result of our demodulation operation, in the case where a pure cosine is transmitted and no noise is added. If the in phase demodulated value is +N when the phase difference between the transmitter and receiver is zero (perfect correlation), then the in phase value would be -N when the phase difference was 180 (perfect anticorrelation). By symmetry, the demodulated value would be 0 for a phase shift of 90. Thus the quadrature component (defined as a sinusoid phase shifted by $\pi/2$ from the in phase component) is orthogonal to the in phase component. The in phase and quadrature demodulated values are obtained by projecting the raw signal onto these two orthogonal components. These two values completely characterize the signal at the carrier frequency.

By putting a scope into XY mode, one can plot the in phase and quadrature demodulated values against one another, to visualize this vector, as in figures 3-4 3-5 3-9. To make visualizing this vector easy, the school of fish units have scope probe attachment points where the in phase and quadrature demodulated values can be observed.

The magnitude of the <in phase, quadrature> vector is the value we want to sense. The angle of this vector represents the relative phase of the transmitter and receiver, offset by any phase shift induced by the capacitance separating them. So we are interested in the magnitude of the vector, and not its angle.

Since each school of fish unit has its own clock, the units are not truly synchronous, unlike the original fish or the smart fish, where the transmitter and receiver are located on the same board. On the timescale of a single measurement burst, the units are phase coherent. But from one measurement to the next, there is typically some phase drift. If the host makes several measurements consisting of a single burst, the angle will drift somewhat from one measurement to the next. Figures 3-4 and 3-5 show two single burst measurements. The phase has drifted somewhat from one to the next. Figure 3-6 shows many single burst measurements overlayed, which reveals the total angular extent of the phase jitter.

In principle, this drift should not matter at all, since we only care about the magnitude of the signal. In practice, the PIC needs to be able to average many measurements, so that it can translate more measurement time into higher contrast to noise figures if desired. But the PIC does not have time to repeatedly calculate the square root of the sum of squared in phase and quadrature values, and the communication bandwidth to send many subsamples back to the host for processing is not available. Instead, we use a "Manhattan metric" approximation to the magnitude: $|I| + |Q|$ instead of $\sqrt{I^2 + Q^2}$, where I and Q are the in phase and quadrature components. This approximation has zero error at phase differences that are multiples of $\frac{\pi}{2}$ ("right angles") and a maximum in the error at angles that are $\frac{\pi}{4}$ away from these (the diagonals). As the phase jitters, the amount of approximation error jitters, and this shows up as measurement noise.

The solution I found was to intentionally step the phase of the receiver relative to the transmitter after each measurement burst, thus making sure that we have a good sampling of all possible phase errors. This gives rise to the "starburst" pattern shown in figure 3-9. Figure 3-10 shows many of these starbursts overlayed. (The overlay figure reveals that not all phases are visited. This is not strictly speaking necessary since the goal is just to achieve a constant sum of phase errors.) Though the phase uncertainty means that entire starburst structure rotates from one measurement to the next, the structure is rotationally symmetric, so the distribution of phase errors—and thus the sum of the phase errors—is roughly constant. Essentially we are integrating out the error that arises from the phase uncertainty. We solved the problem of the absence of phase lock by making it worse.

3.4 From Sensing to Perception: Using the School of Fish

The school of fish is a sensing system with small amounts of computation distributed among the sensor units, which are separated by a low bandwidth connection from the large amount of computational power embodied in the host processor. Because each has a processor, the receivers can make measurements in parallel, which means that the total time to collect all n^2 measurements is $O(n)$. But because the connection to the host is via a serial bus, the time required to communicate all this data back to the host is $O(n^2)$. Thus naively trying to make all possible measurements every cycle would result in a quadratically sluggish sensing process.

The technology available suggests a couple of strategies for speeding up the performance. We can imagine using the processing power distributed in the sensors to lessen the communication load. One possibility would be to compress the data in various ways, for example by sending just changes in signal values, instead of the raw signal values. Ideally one want to send back just the most informative ("surprising") data. A practical way to combine these two ideas would be to make all the n^2 measurements in time $O(n)$, and then have each unit send back the one measurement with the largest change. But in reality, the "surprise" we'd attribute to one of our n measurements must be conditioned the other $n - 1$ values.

A good way to implement this compression would be to collect large amounts of raw sensor data (all n^2 values) at the host processor, and then for each receiver, calculate the covariance matrix for the data vectors it saw. Then one would diagonalize these covariance matrices, extract the eigenvalues and eigenvectors, and download the first couple of eigenvectors (those associated with the largest eigenvalues) to the units. In operation, after a receiver had collected a data vector (a vector of measurements made by transmitting from all the other units), it would project this data vector onto the principle eigenvectors, and

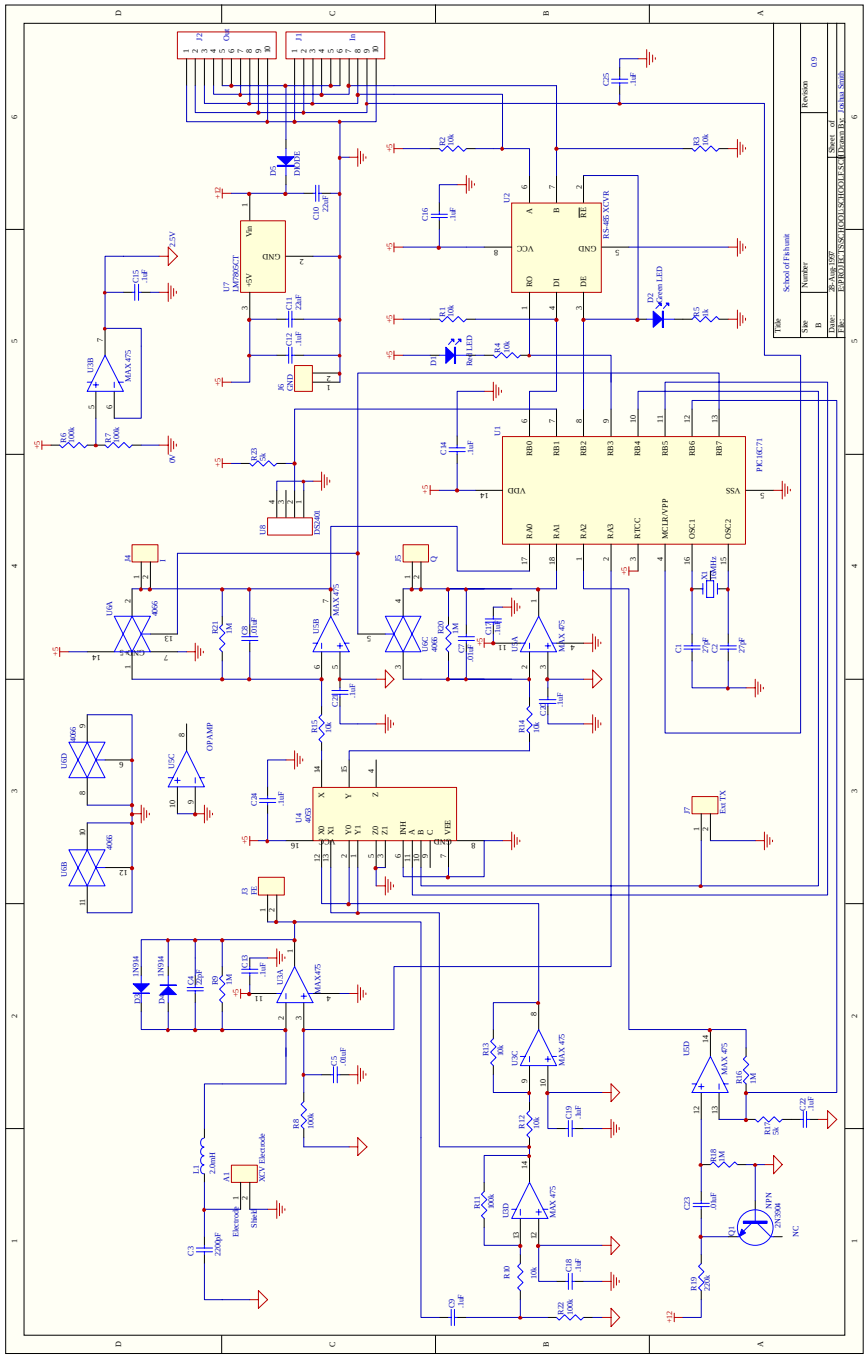

Figure 3-11: Schematic of a School of Fish unit.

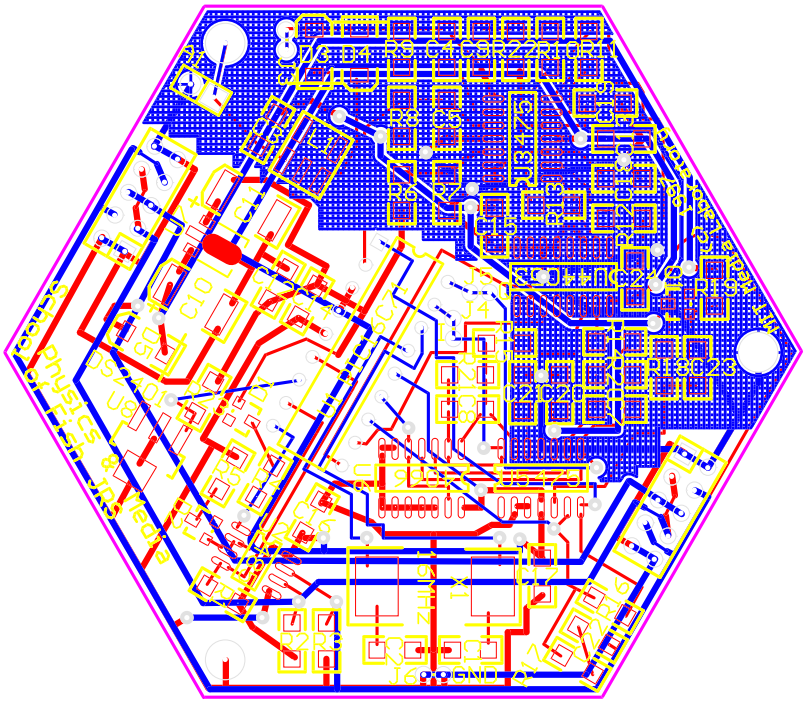

Figure 3-12: Board layout of a School of Fish unit.

Figure 3-13: School of Fish board.

return just these coefficients. In addition to being a good compression scheme, this strategy would probably amount to performing some kind of "salient feature" recognition locally, in the hardware.

Another intriguing possibility is only to collect the most relevant data, by "tracking" the object with the transmitter. This allows the measurement time to be reduced to a constant (instead of being linear). One nice algorithm we have employed is to transmit from one unit, collect the data from the other n, and then make the receiver that returned the largest signal become the next transmitter. Since the capacitance matrix is symmetrical, when an object is nearest a particular pair of electrodes, the selected transmitter oscillates back and forth between the two units closest to the object. Then if the hand moves closer to a different pair, the transmission follows.

This strategy is analogous to the tracking of objects by a foveated eye. In the case of an eye, the scarce resource is presumably the data pathway from the retina to the visual cortex, as well as the processing power at the visual cortex. (Perhaps rods and cones packed at high resolution are also a scarce resource, though if a high resolution fovea can be grown, I don't see why the entire retina couldn't be packed just as densely. Another limited resource may be the area that is actually in focus—an optical constraint.)

In our case, there is no fovea, in the sense that all the units are equally capable (though the units on the periphery are not as useful since they don't have neighbors). However, there is limited sensing time available, limited communication bandwidth (thus limited communication time, in practice) between the sensors and the processor, and limited processing time available at the host end. Which part of "the world" gets most access to these resources can be adjusted by selecting the transmitter. Tracking an object with the transmitter is in this sense analogous to tracking an object with an eye in order to keep the object centered on the fovea.

A capability analogous to peripheral vision also suggests itself quite naturally. If the units far from the current action (i.e. far from the currently selected transmitter) make very fast, low signal-to-noise measurements, they will be able to detect large changes that can then be examined in more detail with the "fovea."

A very appealing feature of this style of sensing is that the highest level perceptual process (the current model of the object's location, for example) feeds back to the lowest levels of the sensing hardware. Information flows bidirectionally, not just from the sensor to the computer. Which transmitter is selected depends on where the host "thinks" the object currently is, or perhaps where the host expects the object to appear. How much time is expended on sensing in regions where nothing interesting is expected to occur is also controlled by the high level perceptual process. A sensing architecture with features like these appears to be more or less necessary—or at least a very good idea—given finite resources. The form it takes in our case is possible because of the architecture of the school of fish: because the transmit/receive state and measurement times are under software control.

Chapter 4

Introduction to Inverse Problems

Inverse problems involve estimating an underlying continuous function from a set of measurements of some aspect of the function. The mapping from the underlying continuous function to the measurements is known as the forward or direct problem. The forward problem must be a well-posed physical problem. If x is an underlying function and y is a set of measurements, then x and y are related by a forward mapping operator F: $F(x) = y$. Note that x and y are not members of the same space—often x is infinite dimensional and y is finite dimensional. The inverse problem is to recover x from a measured y using the inverse operator F^{-1} and (as we will see) regularization, or extra information about the underlying function.

The reason we need the extra information about x is that inverse problems are typically ill-posed. Strictly speaking, to be well-posed, an inverse problem would have to have the following properties: (1) solutions x must be unique, (2) for any data y a solution must exist, and (3) solutions must be stable, so that small perturbations in the data do not lead to large perturbations in the reconstruction.[Isa89] The second, existence condition is often ignored, since in practical situations the data will have been generated from an actual forward problem, guaranteeing the existence of an inverse. (However, it is conceivable that measurement noise might complicate this.)

The real difficulties in practice have to do with the first and third conditions: in some problems, the same data set may have multiple feasible explanations (first problem), or two very similar data sets may have very different explanations (third problem). In the latter case, the noise inherent in any measurement process may cause the system's explanation to vary wildly with the measurement noise.

4.1 Regularization

Regularization is the term for methods for selecting certain solutions when the inverse is not determined, and for stabilizing the inversion process. Standard regularization techniques chose the minimum norm reconstruction (zeroth order regularization), the reconstruction that most resembles a constant (first order or Tikhonov regularization), the reconstruction with the minimum curvature (second order regularization), and so on. Other methods (Backus-Gilbert) try to maximize the stability of the reconstruction. As Press et al. point out, though this form of regularization is fundamentally very different than these other linear regularization methods, the reconstructions it returns tend to be quite similar, because stability and smoothness are correlated. There are also non-linear regularization

techniques such as Maximum Entropy, which, in the context of image reconstruction, maximizes the smoothness of the brightness histogram of the data, without regard to spatial smoothness.[P+92][1]

4.1.1 Bayesian view of regularization

In the Bayesian view of inverse problems and regularization, the forward model becomes an analytical forward probability specifying the probability of a data value conditioned on a setting of model parameters ($p(d|m)$). This requires a model (or at least assumptions) about the noise, as well as the physical forward model. The inversion is accomplished via the magic of Bayes' theorem, which is derived from the trivial probabilistic identity

$$p(m, d) = p(d, m)$$

$$p(m|d)p(d) = p(d|m)p(m)$$

$$p(m|d) = p(d|m)p(m)/p(d)$$

The inverse problem is then reduced to finding the setting of model parameters which maximize the posterior probability $p(m|d)$.

The first form of ill-posedness mentioned above manifests itself as multiple peaks in the $p(m|d)$ distribution. The Hessian matrix (of second partial derivatives) of $p(m|d)$ evaluated at a maximum contains information about the stability of the reconstruction. The eigenvalues and eigenvectors of the inverse Hessian give measures of uncertainty in the principle directions (including the direction of maximum and minimum uncertainty); the product of these eigenvalues (the determinant) is often used as a summary of the total uncertainty (for example, see [Gul88]).

The $p(m)$ term is the prior probability of the model parameter settings; the $p(d)$ term is an often irrelevant normalization constant. (However, $p(d)$, which can be found by normalization, can be useful when comparing different families of models, or models with different numbers of parameters.[Mac91]) Regularization in the Bayesian picture amounts to choosing a prior $p(m)$ that imposes the desired constraints (such as smoothness).[Jay83]

In the sections below, I will continue the discussion of inverse problems by considering two particular cases, computed tomography and electric field imaging, and speculate about a new way to relate them.

4.2 The Radon Transform and Computed Tomography

In a CT (computed tomography) scan, multiple 2d projections of a 3d body are collected, each from a different angle. Each pixel of one of the 2d images represents a line integral of the body's X-Ray attenuation. For simplicity, we will consider parallel projections, in which the X-rays propagate in straight, parallel lines. (Fanbeam projections are also sometimes used.) Since the beam paths are parallel, the 3d structure can be built up from a set of parallel 2d layers. (The term tomography refers to any imaging process in which the image is built up layer by layer.) The problem of recovering 3d structure from 2d projections can

[1] All the material in this paragraph is cited from Numerical Recipes, except for the remark about the smoothness of the MaxEnt brightness histogram, which may not represent the opinions of the Numerical Recipes authors.

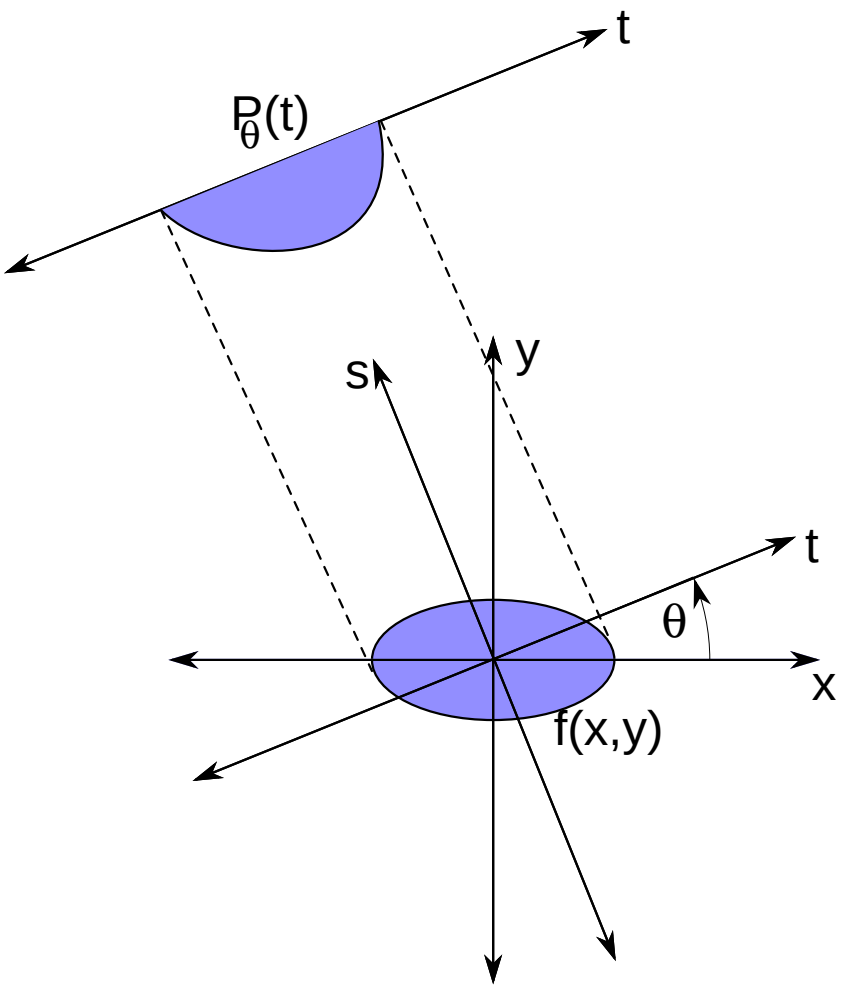

Figure 4-1: The projection $P_\theta(t)$ of an object $f(x, y)$.

be solved by repeatedly recovering 2d structure from 1d projections. For the rest of our discussion of CT, we will therefore focus on reconstructing a 2d attenuation function from its 1d projections.

Figure 4-1 shows the attenuation function $f(x, y)$ and $P_\theta(t)$, a projection of f at the angle θ. Mathematically, $f(x, y)$ and its projections are related by the Radon transform

$$P_\theta(t) = \int_{(\theta,t)line} f(x, y)ds$$

which can be written more explicitly:

$$P_\theta(t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\delta(x \cos\theta + y \sin\theta - t)dxdy$$

Radon showed in 1917 that the function $f(x, y)$ can be recovered given knowledge of $P_\theta(t)$ for all values of θ, that is, given all of its infinitely many projections. In 1973, K.T. Smith proved the apparently discouraging result that $f(x, y)$ is underdetermined by a finite set of projections. His theorem states: "Given an infinitely differentiable object f and a finite

number of directions, there is a new infinitely differentiable object f' with exactly the same shape (i.e. the supports of f and f' are the same), exactly the same projections from these directions, and completely arbitrary on any compact subset of the support f." (In [Kat78].)

The theorem is saying (in strong terms) that the reconstruction is not unique. For a linear measurement operator, non-uniqueness is equivalent to the existence of "ghost" distributions that are invisible to all of the finite set of projections. In other words, the null space N of the measurement operator has non-trivial members. By adding to f an element of N, we get a new distribution f' that yields the same projections as f. The theorem above indicates not only that N is not empty, but that we can find or construct members of N with almost any property we desire.

The practical success of CT scans suggests that these results should not be interpreted too pessimistically. The real significance of the non-uniqueness theorem is that the problem posed above is too general: more realistic constraints need to be put on f: some form of regularization is needed.

In the context of the "simple" problem of reconstructing a continuous function from a finite set of samples, the sampling theorem suggests a kind of regularization: bandwidth limiting. Because a finite set of Fourier coefficients uniquely specifies a bandwidth limited function, imposing bandwidth limits makes the problem of recovering the function from a finite set of samples well posed (invertible).

In practical CT imaging, this same restriction can be used. The problem can be recast so that instead of trying to reconstruct an arbitrary f, we only recover a finite resolution, sampled (and by assumption bandlimited) version of f.[2]

4.3 Fourier Slice Theorem

The Fourier Slice theorem is very useful for understanding the process of collecting projections. It is the mathematical basis of the filtered backprojection algorithm, a simple and practical inversion scheme that I'll describe in the next section. The theorem will also be helpful when we discuss the electrostatic inverse problem.

The Fourier Slice Theorem equates the 1d Fourier transform of $P_\theta(t)$ with a sample (or slice) along the projection angle θ, of $F(u,v)$, the two-dimensional Fourier Transform of $f(x,y)$. The proof of the Fourier Slice Theorem[KS88] is straightforward:

Consider the coordinate system (t,s), a rotated version of (x,y) (as shown in figure 4-1):

$$\begin{bmatrix} t \\ s \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.1}$$

In this coordinate system, a projection along lines of constant t is written

$$P_\theta = \int_{-\infty}^{\infty} f(t,s)ds$$

[2]In the case of CT imaging, the assumption that the object is bandlimited is not always justified. The edges of bones lead to sharp discontinuities in X-Ray attenuation, and the high frequency content in these discontinuities can lead to aliasing. Nuclear relaxation times in the body vary more smoothly, so NMR imaging is less subject to this particular type of aliasing.

Its Fourier Transform is given by

$$S_\theta(\omega) = \int_{-\infty}^{\infty} P_\theta(t) e^{-2\pi i \omega t} dt = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(t,s) ds \right] e^{-2\pi i \omega t} dt = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t,s) e^{-2\pi i \omega t} ds dt$$

We can transform the expression for $S_\theta(\omega)$ back into the (x, y) frame using 4.1:

$$S_\theta(\omega) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) e^{-2\pi i \omega (x \cos\theta + y \sin\theta)} dx dy$$

Since rotations are measure preserving, the $dxdy$ just becomes $dsdt$. This expression represents the two-dimensional Fourier Transform at a spatial frequency of $(u = \omega \cos\theta, v = \omega \sin\theta)$, that is, along a line through the origin at angle θ:

$$S_\theta = F(\omega, \theta) = F(\omega \cos\theta, \omega \sin\theta)$$

.

Now we can use the Fourier Slice Theorem to analyse the process of collecting projections. This will lead us to the backprojection algorithm.

4.4 Filtered backprojection algorithm

The Fourier Slice Theorem tells us that a series of projections in increments of 10 degrees, for example, provides a set of samples of the Fourier transform plane along a set of lines 10 degrees apart, as shown in Figure 4-2. To reconstruct the image $f(x, y)$, we might naively sum all the samples of the Fourier Transform plane and then take the inverse transform. But this would introduce a systematic error. The low spatial frequencies in the center of the Fourier Transform plane have been sampled more densely than the higher spatial frequencies at the outskirts of the plane, because we sampled along lines through the origin. In particular, the origin (the spatial DC component) received a contribution from each sample line. To correct this, we could scale each frequency domain sample by $|k_r|$, where $|k_r|$ is the distance of the sample from the origin in k space (the Fourier Transform plane). Figure 4-3 shows a radial slice of the filter. This is a high-pass filtering operation. Notice that this removes the overemphasized DC component entirely, which is no loss, since it contained no spatial information (just a global "brightness" that would typically be adjusted to make the picture look best anyway).

The final reconstruction could be found by adding together the two-dimensional inverse Fourier Transform of each weighted slice. But to carry out the reconstruction, we do not actually have to perform the forward and inverse Fourier transforms. We can perform the high pass filtering operation on each 1d projection $P_\theta(t)$ separately (and in the original (t, s) domain), by convolving each $P_\theta(t)$ with the Fourier transform h of a slice of the "cone" weighting/filtering function $|k_r|$. But instead of actually using either the "sliced" highpass filter shown in figure 4-3 or the full two-dimensional cone, we can just use the 1 dimensional radial profile of the cone. Since the cone is radially symmetric, all the slices would be the same anyway. So the (t, s) space counterpart of adding together the weighted k-space samples is adding the functions $h(t) * P_\theta(t)$. Since this function has no explicit or implicit s dependence, $h(t) * P_\theta(t)$ has the same value for all s. This means that the value

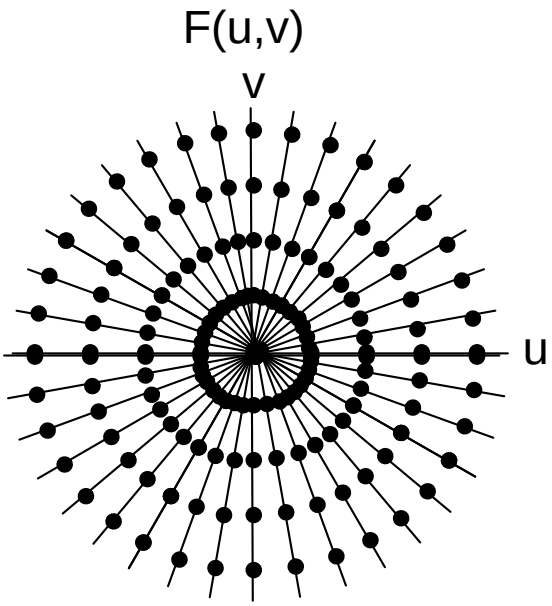

Figure 4-2: The non-uniform sampling of k-space corresponding to a set of projections 10 degrees apart.

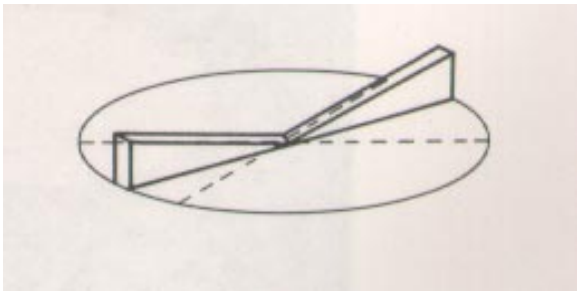

Figure 4-3: A slice of the $|k_r|$ filter high pass filter used to compensate for the non-uniform sampling.

of pixel $p(s,t)$ is given by

$$p(s,t) = \sum_{\theta} h(t) * P_\theta(t)$$

Thus we must "smear" the values $h(t) * P_\theta(t)$ along all values of s, or backproject, and sum the backprojections for each θ value. This is the filtered backprojection algorithm. It is notable that though we derived it starting from the Fourier Slice Theorem, the final algorithm bears little resemblance to the algorithm suggested by the theorem itself. Instead of taking the one dimensional Fourier transform of each of the n projections, interpolating, weighting (filtering), and taking the inverse two dimensional transform, we performed n one-dimensional convolutions, smeared each of these back along the path the X-Rays took, and summed the result. In addition to requiring less computation time, the filtered backprojection algorithm has the advantage that the data from each projection can be processed right away, before the remaining projections have even been measured.

Now we turn to the inverse problems that we really care about. We will point out analogies to X-Ray Computed Tomography when appropriate.

4.5 Nonlinear Electrostatic Inverse Problems

There are three domains in which electrostatic inverse problems have been considered: in geophysical prospecting for minerals and oil, in a medical imaging technique called Electrical Impedance Tomography, and in studies of the "electrolocation" behavior of weakly electric fish. The data interpretation algorithms of the fish cannot be studied as directly the algorithms from the other domains. Nevertheless, quite a bit is known about both the behavior and the neurophysiology of weakly electric fish.

In the most common formulation of the electrostatic inverse problem, one applies current, measures the resulting potentials, and then tries to recover an inhomogeneous conductivity distribution. (Fish are also sensitive to differences in conductivity, but it would probably be misleading to describe them as explicitly recovering a conductivity distribution.)

The forward problem is the Laplace equation with an inhomogeneous conductivity $\sigma(\mathbf{x})$. One way to derive the Laplace equation is to start from the definitions

$$\mathbf{J}(\mathbf{x}) = \sigma(\mathbf{x})\mathbf{E}(\mathbf{x}) \tag{4.2}$$

$$\mathbf{E}(\mathbf{x}) = -\nabla\mathbf{\Phi}(\mathbf{x}) \tag{4.3}$$

and use the fact that the current distribution satisfies

$$\nabla \cdot \mathbf{J}(\mathbf{x}) \tag{4.4}$$

away from all current sources. Substituting 4.2 and 4.3 into 4.4 yields Laplace's equation with an inhomogeneous conductivity:

$$\nabla \cdot (\sigma\nabla\mathbf{\Phi}) = 0$$

In principle, we could use the form of the equation with an inhomogeneous permittivity to describe the forward problem in Electric Field Imaging:

$$\nabla \cdot (\epsilon\nabla\mathbf{\Phi}) = 0$$

Because of the inhomogeneous impedance, this inverse problem is nonlinear: solutions cannot be arbitrarily superposed. However, the problem *does* have a unique inverse (at least for some geometries). In the 1930s, Slichter came up with an analytical inversion formula for the special case of layered structures with no horizontal variations, and showed that the inverse is unique.[Sli33] In the 1980's, a series of mathematical uniqueness results for the problem on the (two dimensional) unit disk were proved.[KV83, KV84, KV85, SU87] Simultaneously, practical work in Electrical Impedance Tomography began.[BB84, Web89] Later, these separate streams of work came into contact. Later still, connections between borehole seismic tomography and electrical impedance tomography were developed.[Ber90, BK90] Though electrical prospecting for oil has been a big business since the 1930s (the Schlumberger corporation was founded to do electrical prospection), the geophysical community does not seem to have been very interested in the problem of explicitly imaging until the contact with the mathematical and electrical impedance tomography literature.

At this point, I'll briefly survey the methods that have been used to solve this nonlinear inverse problem. The survey will be brief because in the next section I will consider in more detail a different, linear problem posed by A. Sezginer[Sez87] in 1989 that we may be able to employ instead of the nonlinear problem. Sezginer makes no reference to any of the literature mentioned above, but basically is geophysically oriented (he was a Schlumberger employee when he wrote the paper).

4.5.1 Inversion techniques from Electrical Impedance Tomography Methods

Most of the techniques for the nonlinear inverse problem have these steps: (1) guess a conductivity, (2) see what measured values that conductivity would produce, (3) adjust the conductivity to decrease the error between the predicted and actual measurements, and (4) iterate until the conductivity stops changing. For step 2, a forward model is needed. This is often a finite element or finite difference model. For step 3, it is necessary to know the Jacobian of the map from conductivity to sensor values, so that some kind of gradient descent can be used. Since the goal of Electrical Impedance Tomography is to image the interior of the thorax, the problem is almost always formulated on a two dimensional disk (which corresponds not to a two dimensional slice of the body, but to a cylinder with no structure in the z direction—every slice is assumed to be the same).

Backprojection

The first algorithm to be used to form impedance images of the interior of the human body was one called backprojection, and was inspired by the CT algorithm.[BB84] It has been criticized on the grounds that the forward measurement does not in fact correspond to a generalized Radon transform, contrary to their assumption.[Bre91] Typically, a current dipole is applied at one location, the resulting voltages are measured everywhere on the boundary, and then the impedance is backprojected along curved equipotential lines. The equipotential lines used are those calculated for the case of homogeneous conductivity. Thus the algorithm relies on a linearization which is appropriate for small deviations from the "default" conductivity, and thus appropriate for the low contrast images characteristic of the interior of the body. The low impedance-contrast assumption is not appropriate for us.

After a measurement is made, the next current dipole is activated, to measure the next projection.

Layer Stripping

In this method, the impedance of a thin strip around the outside of the disk is determined first; given this information, the "boundary conditions" inside this strip are determined, and then the algorithm works its way inward to the center of the disk.[SCII91]

Variational Approach

James G. Berryman proposed a very elegant variational approach to the EIT problem, based on Dirichlet's "principle of least power." His approach doesn't assume a specific geometry. The power dissipated into heat when a transmit electrode is activated is given by

$$P = \int \mathbf{J}(\mathbf{x}) \cdot \mathbf{E}(\mathbf{x}) d^3x = - \int \mathbf{J} \cdot \nabla \mathbf{\Phi} d^3x$$

Dirichlet's principle states that given a conductivity distribution $\sigma(\mathbf{x})$ and a potential distribution $\mathbf{\Phi}(\mathbf{x})$, the power dissipation P is the one that minimizes the integral $\int \sigma |\nabla \Phi|^2 d^3x$. Using this variational principle, Berryman was able to adapt an algorithm based on Fermat's principle that he had previously formulated for borehole seismic traveltime tomography[Ber89, Ber90]. There turns out to be an exact formal analogy between the problems. The dissipated power for impedance tomography corresponds to time in traveltime tomography; conductivity corresponds to the slowness (inverse velocity): high conductivity gives large power dissipation and high slowness gives large propagation time. Finally, the density of electric field lines $|\nabla \mathbf{\Phi}(\mathbf{x})|^2 d^3x$ corresponds to the length element dl. Berryman's algorithm finds the best conductivity distribution by searching for the one that minimizes the dissipated power.[Ber91]

4.6 The induced charge picture and the linear inverse problem

Instead of trying to recover a conductivity distribution, Sezginer considered the inverse source problem of recovering a charge distribution, the right hand side of the Poisson's equation with a *homogeneous* conductivity.[Sez87] As we describe Sezginer's approach, we will try to develop an analogy to computed tomography.

In both cases (X-Ray and electrostatic), a linear operator takes us from the actual distribution to the measured data. For CT scans, the measurement operator is the Radon transform operator. For the electrostatic problem, it is the Laplace operator. The Fourier Slice theorem is used in solving both problems.

Sezginer supposes that a charge distribution ρ induces a potential V that can be measured in the $z = 0$ plane. (See figure 4-4.) Physically, his problem corresponds naturally with the measurement we call "transmit mode," in which we put a transmit electrode in contact with the body being measured. Figure 4-6 shows the effective circuit diagram for a transmit mode measurement,[3] showing just a single receive electrode. Sezginer is imagining that we have a continuum of receive electrodes.

[3] In our case, the transmit electrode is a voltage source. Nevertheless, it induces charge in the body that in turn changes the signals on the receive electrodes. Another difference between our problem and Sezginer's is that we measure charge induced on the receiver, instead of voltage. A further, unimportant difference is that we apply an oscillating signal, and measure displacement current because this measurement is easier than actually measuring charge. But the mapping between the currents we actually measure and the charge

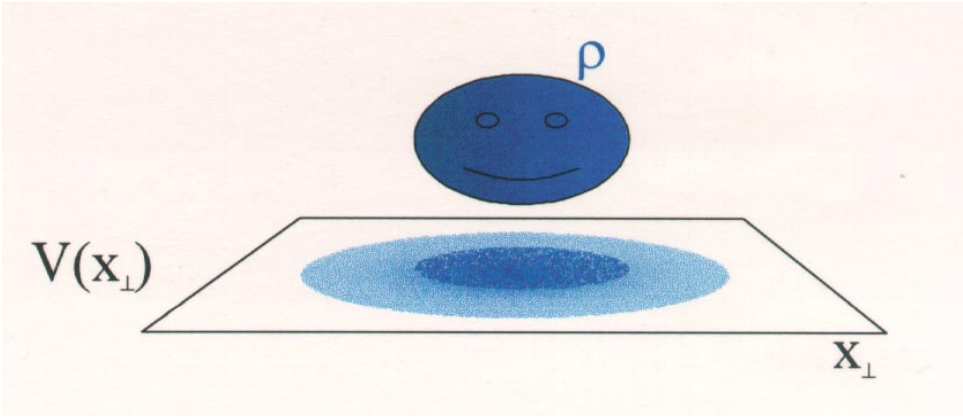

Figure 4-4: Sezginer's problem—imaging a given charge distribution ρ from the potential $V(x_\perp)$ it induces in the $x_\perp$ plane below.

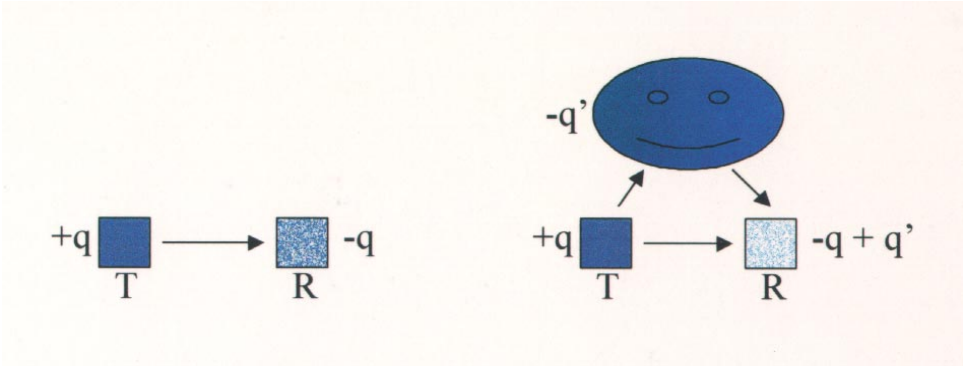

Figure 4-5: "Painting" with charge. Suppose initially that the transmitter has a charge of $+q$. It induces a charge of $-q$ on the receiver. If the charge on the transmit electrode is fixed at $+q$, then when we introduce the body (right), a charge of $-q'$ will be induced, which in turn will induce a charge of $+q'$ on the receiver. Thus the final charge on the receiver is $-q + q'$.

It is also possible to apply Sezginer's analysis to a shunt mode measurement, illustrated in figure 4-7. In a shunt mode measurement, there are separate transmit and receive electrodes, and the body's coupling to ground (capacitor C5 in the figure) is comparable or stronger than the coupling of the transmitter to the receiver. When the hand comes near the transmitter and receiver, current from the transmitter is shunted to ground, the value of the capacitance C0 drops, and less signal arrives at the receiver.

To apply Sezginer's framework to shunt mode measurements, we first would have to assume that the transmitter is a current source.[4] The idea is that the transmitter induces charge in the body being measured, which in turn induces charge in the receive electrode, leading to a change in signal, as illustrated in figure 4-5. If we are correct in assuming that the charge on the transmit electrode does not change as the hand approaches, then we know that the charge induced on the receiver by the transmitter does not change. This means that all the change in signal at the receiver is due to charge induced on the hand that in turn induces a change in charge on the receiver. Therefore, we can take our signal to be *changes* in received charge, and try to recover the charge distribution on the body that is responsible for the change in signal.

However, this procedure makes no use of our knowledge of the transmitter's location, which must be an important piece of information. It would (presumably) be hopeless to try to reconstruct a CT scan with knowledge only of the measured projections, and not the angles at which the projections were made. An important open question is whether we can keep the linear framework we get by reconstructing induced charge, but find a way to make use of knowledge of the transmitter location. One distinct possibility is that re-introducing this knowledge would make the problem non-linear.

Another possible problem with the induced charge approach is that charge may be the wrong thing to image: any part of the body with no induced charge will be invisible, and parts of the body nearer the transmitter will typically be "brighter." Nevertheless, since the body is virtually a perfect conductor, and we are not interested in forming a grayscale image of charge or impedance (we just want to determine at each point in space whether any charge is present), we could threshold the continuous charge distribution we recover, and infer the presence of part of a hand anywhere we see any charge. Note that we can put together the results of measurements made with different transmitters this way. However, it does not seem that this approach is making full use of our knowledge of the transmit location. If two different conductivity distributions happened to yield the same charge distribution when illuminated with two carefully chosen transmitters, we would use the resultant charge distribution the same way in either case, though given our knowledge of the transmit location, we should have been able to distinguish the two.

The relationship between ρ and V is given by Poisson's equation:

$$\nabla^2 V = -\rho$$

Taking 3d Fourier transforms, we can re-write this as

$$\hat{V}(\mathbf{k}) = \hat{\rho}(\mathbf{k})/||\mathbf{k}||^2$$

---

we could in principle measure is trivial.

[4] We could also measure the current leaving the transmitter—an additional "loading mode" measurement that we do not make at present. Another possibility is to assume that though it is a voltage source, it is not loaded heavily, so that the current leaving the transmitter is approximately constant. It would be necessary to investigate the conditions for the validity of this approximation.

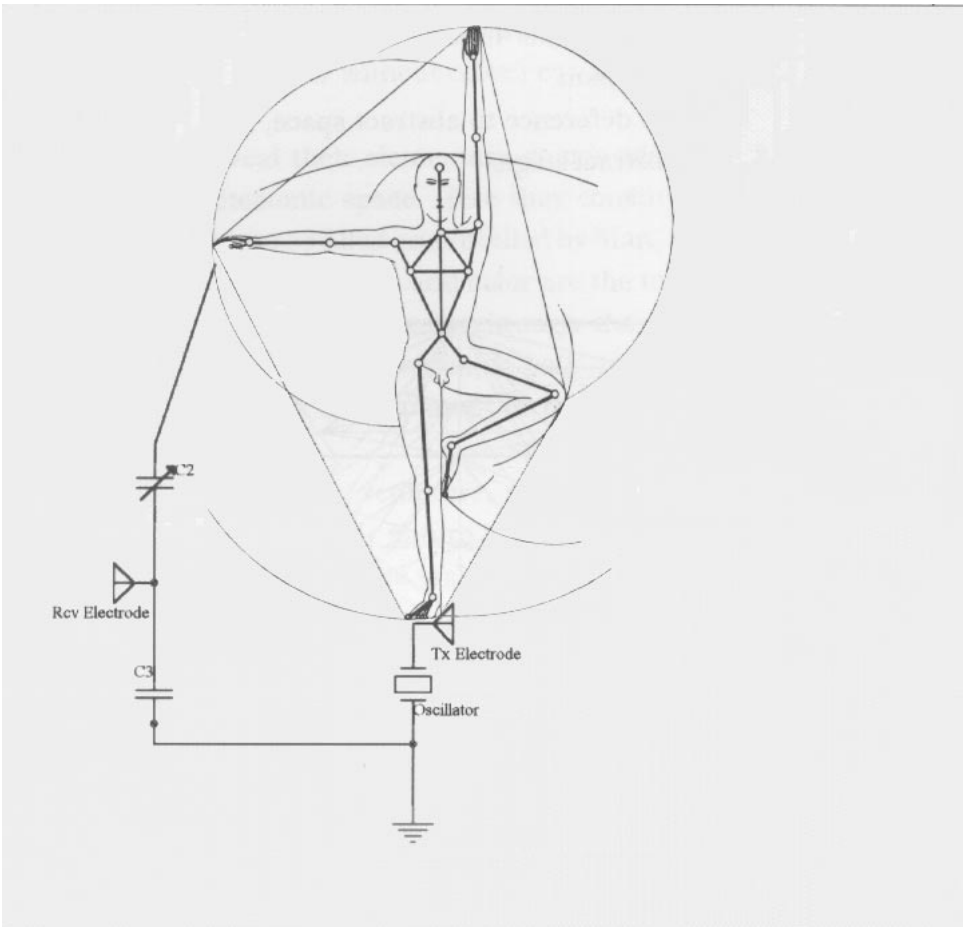

Figure 4-6: Effective circuit diagram for a transmit mode measurement.

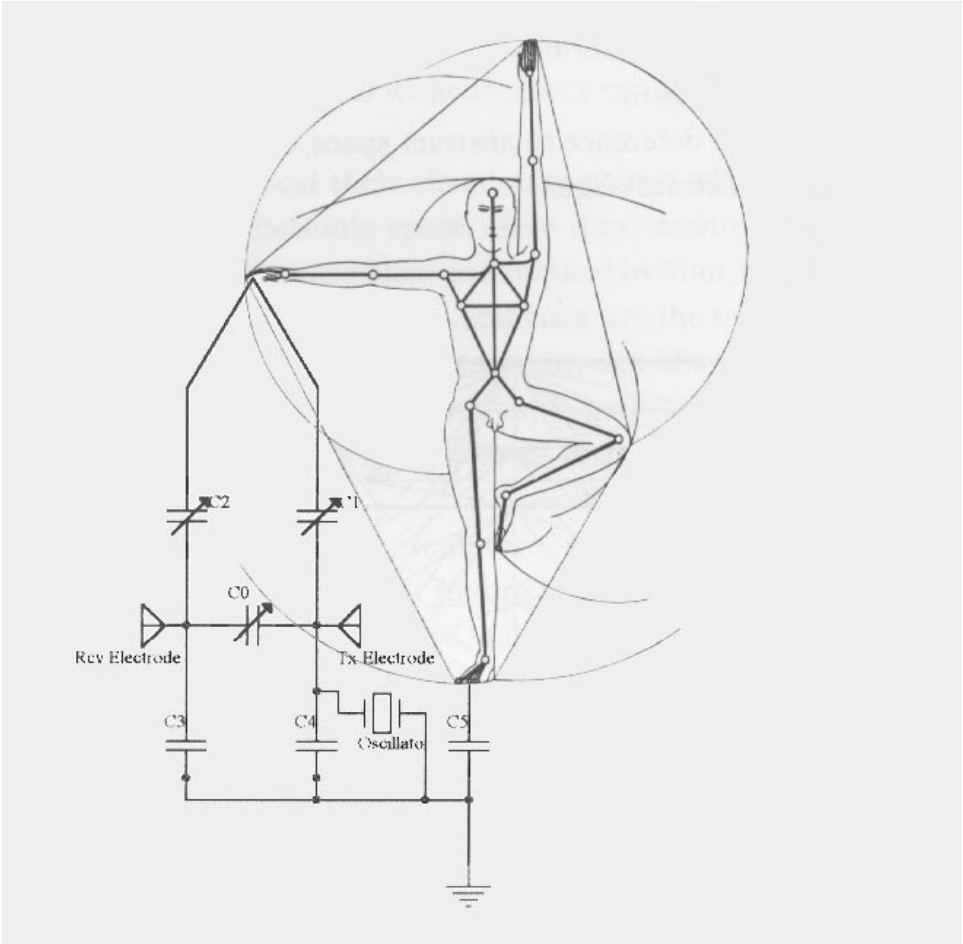

Figure 4-7: Effective circuit diagram for a shunt mode measurement.

By assumption, we only know V on a plane (or slice); we do not have access to the full 3 dimensional distribution. We will denote the slice of V data in our possession by $V(\mathbf{x}_\perp)$, and its Fourier transform by $\hat{V}(\mathbf{k}_\perp)$. This can be related to the charge distribution ρ using the Fourier slice theorem and Poisson's equation. Since $V(\mathbf{x}_\perp)$ is a slice (at angle 0) through V, we know by the FST that this slice is equivalent to the Fourier transform of the projection, at angle 0, through $\hat{V}(\mathbf{k})$:

$$V(\mathbf{x}_\perp) = F\{\int_{-\infty}^{\infty} dk_z \hat{V}(\mathbf{k})\}$$

where $F\{\}$ denotes the Fourier transform. Taking another Fourier transform, and using the fact that the FT is its own inverse, we end up with

$$\hat{V}(\mathbf{k}_\perp) = \int_{-\infty}^{\infty} dk_z \hat{V}(\mathbf{k}) = \int_{-\infty}^{\infty} dk_z \hat{\rho}(\mathbf{k})/||\mathbf{k}||^2$$

Before we consider the problem of trying to infer ρ from $V(\mathbf{x}_\perp)$, this new description of the measurement process gives us a lot of insight into what we can and can't hope to infer. The data $\hat{V}(\mathbf{k}_\perp)$ is a projection along the z direction of a low-pass filtered version of $\hat{\rho}$. Because the data is effectively a single projection, it is easy to construct distributions that yield the same set of measurements. But by making multiple measurements at various angles, we could form an essentially complete picture of $\hat{\rho}$. However, it is clear that high frequency spatial information is rolled off in proportion to the square of the spatial frequency, so this method is not ideally suited to forming detailed, high resolution pictures such as those needed in medical applications. However, for computer interface purposes, in which the computer is supposed to actually understand and use the data, rather than create an image for human consumption, high frequency detail may be a hindrance rather than a help.

But continuing the analogy with X-Ray Computed Tomography (which in a certain theoretical sense is impossible, but in practice is possible), we might ask what well-posed inverse problems we can construct by regularizing or supplying additional a priori information. Sezginer shows that the minimum norm solution is uniquely determined by the data. Selecting the minimum norm solution is known as zeroth order regularization. This seems to make some intuitive sense: the minimum norm solution is orthogonal to all members of the null space. Thus any component of the reconstructed distribution that does not contribute to the measured signal is omitted from the explanation. When put this way, it sounds like a kind of Occam's razor: do not multiply (or sum?) components beyond necessity. However, when we examine the results of zeroth-order regularization in this case, we will see a problem.

Sezginer gives the transform of ρ_0, the minimum norm solution, as

$$\hat{\rho_0}(\mathbf{k}) = (2/\pi)\hat{V}(\mathbf{k}_\perp)||\mathbf{k}_\perp||^3/||\mathbf{k}||^2$$

To make clear the properties of the regularization that is occurring, Sezginer expresses the inferred minimum norm distribution ρ_0 in terms of the actual ρ:

$$\rho_0(\mathbf{x}) = \int d^3\mathbf{x}'\rho(\mathbf{x}')\frac{1+3cos2\theta}{4\pi[||\mathbf{x}_\perp - \mathbf{x}'_\perp||^2 + (|z|+|z|')^2]^{3/2}} \tag{4.5}$$

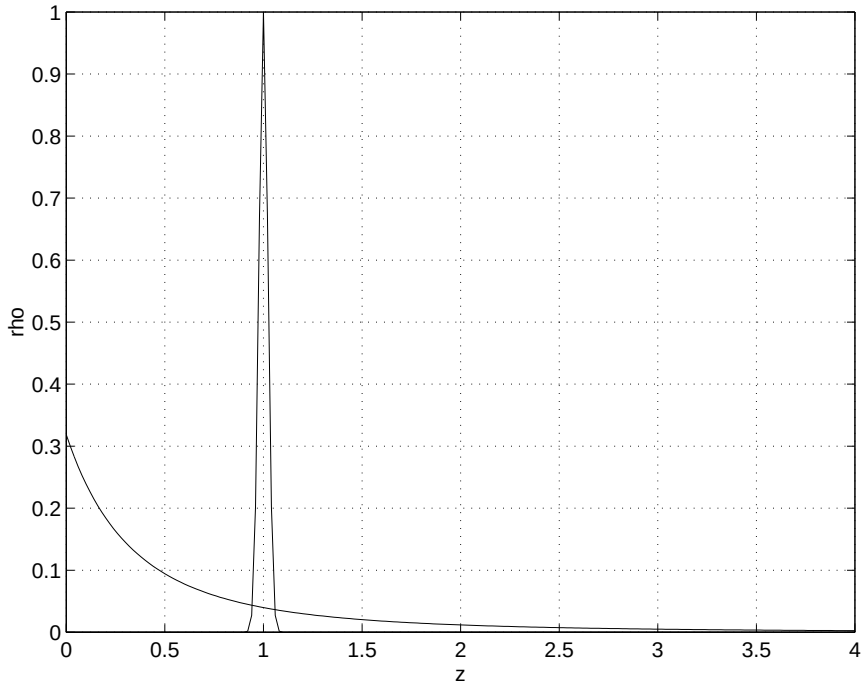

Figure 4-8: The minimum norm charge distribution inferred by Sezginer's formula when given a delta function charge at $(0, 0, 1)$. The delta function is shown along with its minimum norm reconstruction. The charge density is plotted along the line from $(0, 0, 0)$ to $(0, 0, 4)$.

where $\mathbf{x}_\perp$ and $\mathbf{x'}_\perp$ are the projections of the vectors $\mathbf{x}$ and $\mathbf{x}'$ onto the xy plane, and

$$cos\theta = (|z| + |z'|)/[||\mathbf{x}_\perp - \mathbf{x'}_\perp||^2 + (|z| + |z|')^2]^{1/2}$$

To see an example of a problem with this regularizer, consider a unit charge at location $\mathbf{x}' = (0, 0, 1)$, that is, the distribution $\rho(\mathbf{x}') = \delta(0, 0, 1)$. Substituting it into the filter described by equation 4.5, we get $\rho_0(\mathbf{x}) = \frac{1}{\pi(z+1)^3}$. So our point charge at distance one away from the imaging plane has been reconstructed as an inverse cubic distribution, as shown in figure 4-8. This is completely wrong for our purposes: for applications such as making a (one handed) mouse, we want the mean and perhaps the zeroth and second moments of the distribution to be correct, and other than that, we don't care.[5]

Because the regularization chosen by Sezginer loses all depth information, it is completely inappropriate for our purposes. Furthermore, his method recovers a charge distribution, rather than a conductivity distribution, which means that it is unclear how to integrate the information gathered by multiple transmitters. Rather than finding the continuous charge distribution containing the least total charge that could explain our data, we need to find the three dimensional position and orientation of a small number of conductive objects (the "gloveless dataglove" problem). The next chapter will more carefully articulate the difference between recovering a charge distribution and recovering a conductivity distribution, and the method presented in the following chapter will allow us to solve the

[5]Even though the reconstruction of the delta function looks totally wrong, you might wonder whether it had the correct mean. It doesn't.

gloveless dataglove problem.

Chapter 5

Inverse Electrostatics

The important message of this chapter is that while electrostatics is linear in the strengths of fixed-location charges, or in the voltages of fixed location equipotential boundaries, it is not linear in charge or boundary locations. Thus our problem, recovering the unknown position of one or more equipotential surfaces, is not linear.

5.1 Theory

The term inverse problem is almost always shorthand for non-linear inverse problem. If the relationship between the underlying object being studied and the data produced by the measurement process is linear, then the reconstruction involves finding the inverse of a linear operator, that is, inverting a matrix.

What about Electric Field Imaging? The Maxwell equations are linear, and so is the Laplace equation (a low frequency special case of the Maxwell equations, as explained in chapter 1). Are we dealing with a linear problem, then?

5.1.1 Linear formulation

The potential at a set of observation points $\mathbf{x_j}$ due to a set of point charges at locations $\mathbf{r_i}$ is given by

$$V(\mathbf{x_j}) = \sum_i \frac{q_i}{|\mathbf{x_j} - \mathbf{r_i}|}$$

If the locations of the charges are fixed, but the strengths unknown, then the problem is linear. The distances between the source charges and the observation points is a matrix of constants that can be precomputed.

$$V(\mathbf{x_j}) = D_{ij} q_i$$

In this formulation, the problem of recovering the charge strengths from the potential measurements reduces to inverting the distance matrix. Of course, whether this matrix is invertible at all (uniqueness) and how sensitive the reconstruction is to measurement noise (ill-posedness) depends on the specifics of the problem: how many charges are being sensed, and how many and which observation points are available.

The problem is still linear in the continuum limit in which the object being sensed is a continuous charge distribution. Sezginer analyzed the problem assuming that the potential

is observed in a single plane. This problem is underdetermined, but he presented a minimum norm solution, which corresponds to zeroth order regularization.

5.1.2 Nonlinear formulation

If the charge locations are not fixed, then the problem is nonlinear, whether or not the charge strengths are fixed.

$$V(\mathbf{x_j}) = \sum_i q_i f(x_j, r_i)$$

where $f(x_j, r_i) = \frac{1}{|\mathbf{x_j} - \mathbf{r_i}|}$. Note that f depends in a non-linear fashion on $\mathbf{x_j}$ and $\mathbf{r_i}$.

5.2 The Electric Field Imaging inverse problem

Neither of the formulations discussed above captures our problem exactly. The electric field imaging problem is to find the location of a boundary known to be at a particular potential. First of all, is this problem linear or non-linear?

Though we can superpose the fields due to source charges at specified locations, we cannot superpose the field due to an additional conductor. From a microscopic physical point of view, this is easily understood. By definition, free charge can roam about the surface of a good conductor. When we bring a new conductor in the vicinity of other charged conductors, the existing charges induce charge in the new conductor. But the charge on the new conductor also exerts attractive and repulsive forces on the free charge in the surface of the existing conductors. This causes the distribution of charge in the existing conductors to change when the new conductor is brought in proximity. Thus the field due to a collection of conductors cannot be found by superposition of the fields caused by the conductors separately. Ultimately the field due to a system of conductors is more like the second, non-linear formulation above, because the charges on the surface of the conductor are free to move around.

As explained in chapter 1, the charge on conductor i induced by voltages V_j on the other conductors is (perhaps confusingly) linear in the voltages on these conductors:

$$Q_i = \sum_j C_{ij} V_j$$

The capacitance matrix C_{ij} depends only on the geometry of the conductors, not on any particular potentials or charges. The definition of C_{ij} is given by $C_{ij} = \frac{Q_i}{V_j}$ when the potential on all the other conductors $V_{j'} = 0, j' \neq j$. If one has solved Laplace's equation and then found the C_{ij} factors, finding the charges as a function of voltages is a linear problem. But when solving Laplace's equation, all the conductors had to be specified as boundary conditions. It isn't possible to decompose the boundary conditions into small pieces of conductor. Laplace's equation is linear in the boundary voltages, but not in the location of the boundaries. The case for point charges (in which the problem is linear in the charge strengths but not in the charge locations) is analogous, and ultimately identical, since it is a surface distribution of charges across the conductor boundaries that ensures that the potential has a specified value. If the conductor locations are fixed, then the capacitance matrix is fixed, so scaling the voltage on a particular conductor by a factor a requires scaling the charge on that conductor by a. This is consistent with our assertion

that Laplace's equation is linear in charge *strengths* and boundary *voltages,* but not charge or boundary *locations.*

Since our goal is to track moving conductors (for example, hands), the forward problem is non-linear and non-trivial, and so the inverse problem is also non-linear and non-trivial.

5.2.1 Uniqueness

Null space analysis for linear imaging problems

We'll define a linear imaging method to be one with the property that input distributions ρ are related to measured signal vectors V by a linear measurement operator M: $V = M\rho$. Crucially, linear imaging techniques obey superposition, so if ρ is expanded in some set of basis distributions ρ_i, V can also be

$$V = \sum_i V_i = \sum_i M\rho_i = M\sum_i \rho_i = M\rho$$

. In particular, $V_i = M\rho_i$.

For linear measurement systems, uniqueness—the question of whether distinct inputs always result in distinct outputs—is relatively easy to understand. Suppose two distributions ρ_1 and ρ_2 give the same measured signals V. Then ρ_2 can be expressed in terms of ρ_1 and a distribution ρ_g:

$$\rho_2 = \rho_1 + \rho_g$$

Applying the measurement operator,

$$M\rho_2 = M\rho_1 + M\rho_g$$

which implies that

$$V = V + M\rho_g$$

so it is clear that

$$M\rho_g = 0$$

For linear measurement systems, non-uniqueness implies the existence of a "ghost" distribution that is invisible to the sensing system. In mathematical terms, it implies that the null space of the measurement operator is not empty. Conversely, by proving that the null space of the measurement operator *is* empty—by proving that ghosts don't exist—we can prove uniqueness.

Any practical sensing system will collect a finite number of measurements of an underlying continuous distribution, and thus strictly speaking will not allow continuous distributions to be uniquely identified. However, often it is possible derive constraints on the underlying distribution that will lead to uniqueness. One example of such a constraint is bandlimiting. In reconstructing a continuous audio signal from a finite set of samples, or reconstruction of a 2d attenuation distribution from a finite set of CT projections, the assumption that the continuous distribution is bandlimited allows it to be reconstructed uniquely from a finite set of measurements.

One approach to regularization (the procedure by which constraints are applied that render the reconstruction invertible) is to project any feasible (but not necessarily unique) reconstruction onto the null space, and then subtract the component of the null space, so that the resulting reconstruction vector is orthogonal to the null space. This is the approach

taken by Sezginer, as described in the previous chapter. This particular regularizer turns out to be particularly inappropriate for our problem, as explained in that chapter.

Unfortunately, none of the linear techniques are strictly speaking applicable for us, since our forward problem is in fact non-linear. However, it is reasonable to wonder whether the linear approaches could be applied to the first order, linearized version of our forward problem, to produce uniqueness proofs for solutions to the first order version of the problem. Though I have not done so, this seems to be a very reasonable goal, since this approach has been successfully employed in the context of CT scanning,[Kat78] which involves reconstructing an attenuation distribution that is non-negative, much like a conductivity distribution, and unlike the charge distributions considered by Sezginer.

Nonlinear problems and genericity

In considering the fully non-linear version of the problem, note that each additional measurement reduces by one the dimensionality of the feasible subset of parameter space,[1] as long as the measurements are independant (non-degenerate). We conjecture, by analogy with "embedding" results in non-linear time series analysis,[Tak81] that for non-linear problems, this condition holds generically: for almost any choice of measurements and parameters, all the measurements will be non-degenerate, and therefore each measurement will shrink by one the dimensionality of the space of parameters that are consistent with the data. Of course bad choices are possible, but perturbing these a small amount should lead to choices that are not bad.

5.2.2 Stability and Ill-posedness

The conjecture about genericity in the previous paragraph should not be interpreted too optimistically. The suggestion that perturbing a bad sensor geometry a small amount will create a "good" one is only true in the limit of no noise. Even if such a perturbation does result in a sensor geometry that, technically speaking, can be used to uniquely identify all the parameters of interest, the reality is that for a sensor geometry produced in this way, the solutions would still be unstable, and therefore not invertible in practice.

To reiterate, the existence of a unique inverse for a given set of parameters and measurements is a necessary but not sufficient condition for an inverse problem to be solvable in practice (to be "well-posed"). Typically, stability of reconstruction is a more difficult condition to satisfy than existence of a unique inverse.

The statement that the solution of an inverse problem is unstable means that small changes in data can lead to large changes in reconstruction. Section 4.1.1 explains how to use the Hessian matrix of the posterior probability distribution evaluated at a liklihood maximum to mathemtically characterize instability. For an example of this technique in use, see [Smi96].

[1] The feasible subset of parameter space is the "surface" in parameter space that is consistent with the data.

Chapter 6

Sphere Expansion

To solve the forward problem exactly, it would be necessary to solve the Laplace equation using numerical techniques such as successive overrelaxation or a finite element method. However, these approaches are too slow for our purposes. Our goal is to make a real time input device, which means we must solve the inverse problem at least 30 times per second. A single solution of the inverse problem requires the forward problem to be solved many times, so we will require a very fast method for solving the forward problem.

The need for computational efficiency, plus the fact the we typically have some prior knowledge of the geometry—either the shape of the object being tracked, or the position of an object whose shape we are trying to infer—suggests an additional desirable feature for our forward model: it should allow us to make use of prior geometrical constraints. We can formulate most of our important applications, such as the "gloveless data glove," in terms of a geometrical model (of a hand, say) with a few unknown parameters (position, orientation, and joint angles).

In video-based body tracking (computer vision), one collects a large array of numbers, from which a small number of body model parameters are extracted. Collecting this large data set requires high input bandwidth, and then substantial computational resources to extract the parameters—in essence to throw away most of the data that was so laboriously gathered. Electric Field Imaging offers the possibility of extracting the small set of relevant parameters from an input data set of comparable size.

The obvious brute force forward model, a finite element solution of the Laplace equation, would negate some of these advantages. The finite element solution requires stepping from the relatively small input space up into the much higher dimensional space in which the field at a mesh of points is found, before numerically integrating to return the low dimensional parameter space of interest.

Alternatively, one might imagine an "eigenfunction" or SVD approach which avoids stepping into the higher dimensional space by solving for the coefficients of the n (or fewer) principal components of the conductivity field in terms of n measured values. Or one might find n coefficients of a Fourier expansion of the field from n measured values. However, neither of these global representations is ideal for the purpose of creating input devices, because they leave us one step away from our ultimate goal of extracting body parameters from field measurements. Given a set of coefficients for these global basis functions, we would have to solve an additional "computer vision" problem to extract the body parameters of interest. Or, we could perform a search in the smaller body parameter space if we could move from the body parameter space into the global representation in an efficient manner.

However, this move is typically cumbersome computationally. It would involve operations such as taking the Fourier transform of the distribution represented by the model.

The "sphere expansion" method that will be presented in this chapter is a computationally inexpensive technique for solving the forward problem based on an approximate local representation of the conductivity distribution. The distribution is approximated by a set of conductive spheres. It is trivial to move from the body parameter representation into the local representation used for computation of the forward problem. For example, a hand might be represented by three colinear spheres a fixed distance from one another. In this example, the body parameter representation has 3 position coordinates and two orientation coordinates (pitch and yaw, roll being indistinguishable using only two spheres). The local representation consists of the positions and sizes of the spheres.

Given this explicit representation of the conductivity distribution, a series solution to the forward problem can be developed using the "series of images" technique explained below. As the number of terms in the series increases, the approximate solution of the forward problem converges to the exact solution, for the particular conductivity distribution.

6.1 Method of Images

The method of images is a classic analytical technique for exactly solving electrostatics problems that feature a point charge (or arbitrary charge distribution, more generally) in the presence of a planar or spherical equipotential surface. The boundary condition due to the equipotential surface is exactly matched by a single point charge, called an image charge.

6.1.1 Ground plane

The planar case is very simple. The field due to a point source of charge $+q$ at location $(x, y, z) = (0, 0, z_0)$ above a grounded plane at $z = 0$ is identical to that caused by charges $+q$ at $(0, 0, z_0)$ and $-q$ at $(0, 0, -z_0)$. The potential $V(r)$ due to a point charge q is $\frac{q}{r}$. Since the image charge was placed at $-z$, any point on the plane is equidistant from the $+q$ and the $-q$. Consider an arbitrary point $(x, y, 0)$ in the ground plane. The distance from the real charge $r_0 = ((x-0)^2 + (y-0)^2 + (0-z_0)^2)^{\frac{1}{2}}$; the distance from the image charge, $r_1 = ((x-0)^2 + (y-0)^2 + (0+z_0)^2)^{\frac{1}{2}} = r_0$. Thus $V(z=0) = \frac{q}{r_0} + \frac{-q}{r_1} = \frac{q}{r_0} - \frac{q}{r_0} = 0$.

6.1.2 Sphere

Figure 6-1 shows a point charge q located at $\mathbf{r}$ in the presence of a grounded sphere of radius a centered on the origin. The magnitude of the image charge needed to match the sphere boundary conditions is given by $q' = -\frac{a}{r}q$, and the location of the image charge is $\mathbf{r}'$, a distance $r' = \frac{a^2}{r}$ from the origin along the line connecting the center of the sphere to the source charge.

The potential at an arbitrary observation point $\mathbf{x}$ is given by

$$V(\mathbf{x}) = \frac{q}{|\mathbf{x} - \mathbf{r}|} + \frac{q'}{|\mathbf{x} - \mathbf{r}'|}$$

We will rewrite this in terms of unit vectors $\mathbf{n}$ and $\mathbf{n}'$ which have the same direction as $\mathbf{x}$

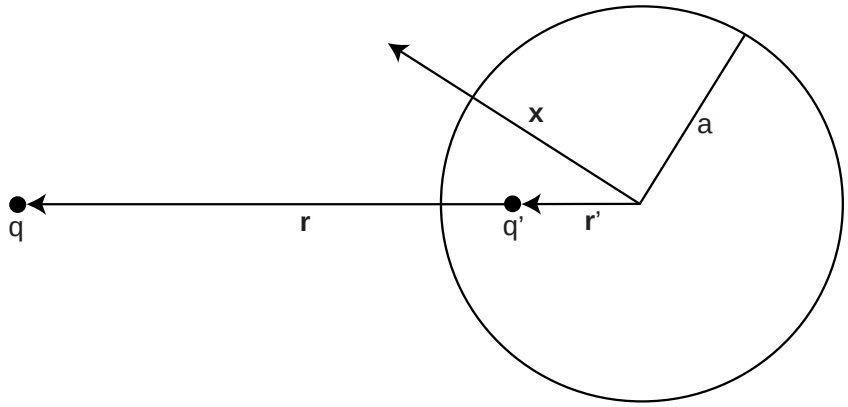

Figure 6-1: Matching the boundary conditions of a point charge in the presence of a grounded sphere with a single image charge.

and $\mathbf{r}$ respectively.

$$V(\mathbf{x}) = \frac{q}{|x\mathbf{n} - r\mathbf{n}'|} + \frac{q'}{|x\mathbf{n} - r'\mathbf{n}'|}$$

We want $V(x = a) = 0$.

$$V(x = a) = \frac{q}{|a\mathbf{n} - r\mathbf{n}'|} + \frac{q'}{|a\mathbf{n} - r'\mathbf{n}'|} = \frac{q}{a|\mathbf{n} - \frac{r}{a}\mathbf{n}'|} + \frac{q'}{r'|\frac{a}{r'}\mathbf{n} - \mathbf{n}'|}$$

It will be the case that $V(x = a) = 0$ when $\frac{q'}{r'} = -\frac{q}{a}$ and $\frac{r}{a} = \frac{a}{r'}$. The latter condition guarantees that the absolute value terms are identical. If we let $c = \frac{r}{a} = \frac{a}{r'}$, then the absolute value terms can be written $|c\mathbf{n} - \mathbf{n}'|$ and $|c\mathbf{n}' - \mathbf{n}|$. These quantities are equal, which can be seen clearly in figure 6-2. Triangles $O, c\mathbf{n}, \mathbf{n}'$ and $O, c\mathbf{n}', \mathbf{n}$ are similar, so $|c\mathbf{n} - \mathbf{n}'| = |c\mathbf{n}' - \mathbf{n}|$.

Thus to match the sphere boundary conditions, the necessary choices of r' and q' are

$$r' = \frac{a^2}{r}$$

and, substituting this expression into $q' = -\frac{r'}{a}q$,

$$q' = -\frac{a}{r}q$$

By symmetry, a source charge inside the sphere can be replaced by an image outside using the same equations. Note that in the limit of large a, the image formulae for the sphere approaches those for a plane.

Sphere at a potential other than zero

The case of a sphere at a potential V_c not equal to zero can be handled in a straightforward fashion by solving the problem for the $V = 0$ case, and then placing an additional image charge q_c located in the center of the sphere. Since $V = \frac{q}{r}$, $q_c = aV_c$. The first image charge was chosen such that $V = 0$ on the sphere's surface, so the potential on the sphere due to both image charges is V_c.

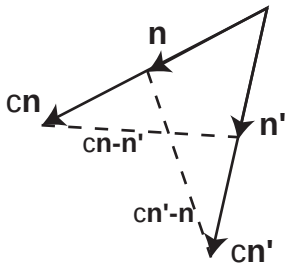

Figure 6-2: This figure illustrates a step in the derivation of the image charge formula: $|c\mathbf{n} - \mathbf{n}'| = |c\mathbf{n}' - \mathbf{n}|$.

6.1.3 Images of multiple charges and the method of inversion

Since charges can be superposed linearly, the method of images can be used to match the boundary conditions due to an arbitrary charge distribution in the presence of a conducting sphere. The transformation of replacing source charges with image charges corresponds to a classical geometrical technique, the method of inversion. The center of the conducting sphere is known as the center of inversion in geometry. This transformation, inverting about a point, preserves angles, so sometimes a geometrical proof can be more easily obtained after inversion.

The "source" charge distribution can be outside the sphere, inside the sphere, or both. A spherical source distribution transforms into a spherical image distribution as long as the spherical source does not pass through the center of inversion. If the source distribution does pass through the center of inversion, it is transformed into a plane. Symmetrically, a planar source distribution transforms into a spherical image distribution that passes through the center of inversion.

6.2 Series of Images

Unfortunately, the method of images applies only to planes and spheres. While the field due to two static point charges can be found by superposition, the field due to two charged spheres or to a point charge in the presence of two conducting spheres or planes cannot—the conductors interact with one another. At a microscopic level, the free charges in the conductor may roam anywhere on the conductor's surface. The free charges on one conductor influence the free charges on the other, so the final distribution depends in a nontrivial way on the geometry of both conductors. A pair of spheres in the presence of a point charge has much less symmetry than a single sphere in the presence of a point charge, so it would be very surprising if it were possible to simulate the final charge distribution using just two image charges.

However, by appropriately placing multiple charges, we can approximate these more complex boundary conditions as closely as desired. In placing these charges, we will be constructing a series solution to Laplace's equation. For the first term in the series solution, we find the image charges for the two spheres independently, neglecting interactions between the spheres. To find the next term in the series, the first order image charges are treated as source charges, and second image charges are placed to correct the deviation from the desired boundary conditions caused by the first order image charges. The process can be extended to as many orders as desired.

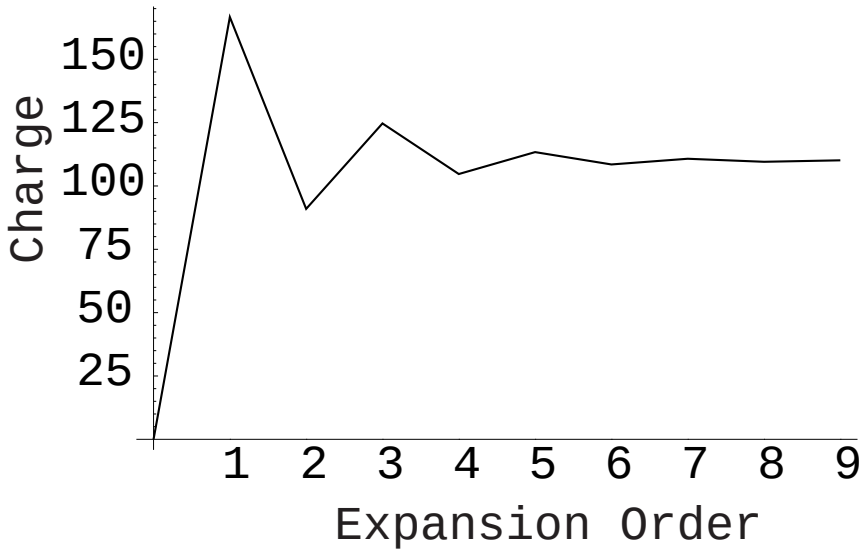

Figure 6-3: Convergence of sphere expansion. Total charge on a grounded "receive" sphere in the presence of a "transmit" sphere at V_t and a grounded "hand" sphere.

The number of charges one must keep track of grows exponentially with the order, but only geometrically with the number of spheres. For m spheres, one of which is a transmitter (containing a source charge at its center, which is equivalent to setting its boundary voltage to $\frac{q}{a}$, where a is its radius), the number of new image charges that are produced at order p is given by $(m-1)^p$.

6.2.1 Convergence

Figure 6-3 shows the convergence of the total charge induced on a grounded "receive" sphere of radius 5 at $(x, y, z) = (-15, 0, 0)$ by a radius 5 "transmit" sphere at a potential of -200 and location $(x, y, z) = (15, 0, 0)$ and a grounded radius 12 "hand" sphere at $(x, y, z) = (0, 20, 0)$. On physical grounds, one would certainly expect any series generated in this way to converge to a finite value.

Mathematically, it is easy to prove with the comparison test that for specific cases the charge series converges. One compares the charge series term by term with another series that is known to converge. The comparison test says that if our series is term-by-term less than a series known to converge, then our series is also convergent. A more general proof is not as easy.

In practice, for non-overlapping spheres, the series is well behaved for every example I have encountered. For non-overlapping spheres, the sign of the charge alternates at each iteration (order), and the magnitude decreases. This alone does not guarantee convergence. Like the harmonic series, it might not converge though each term decreases in magnitude. But it is not actually obvious that the total magnitude of the charge is guaranteed to decrease at every order. Perhaps for some strange, densely packed collection of spheres, the magnitude of the image charge induced in one round could exceed the magnitude of the source charge. From a physical point of view, this would presumably violate conservation of energy, an observation that might suggest a general proof.

6.2.2 Numerical Quadrature

For non-overlapping spheres, the sum over- and undershoots the true value as the order of approximation increases. This is true because the sign of the new terms is always opposite the sign of the previous terms, and the absolute value of the new terms is always less than the absolute value of the previous terms. This suggests a "numerical quadrature" technique for improving the accuracy of our estimate at any order. If we want the answer to order p, rather than simply summing the p terms in the series, we can take only half of the pth term. This will always yield a better answer than including the pth term itself.

6.3 Application Examples

In this section we present additional applications of the sphere model.

6.3.1 Crossover to Transmit mode

The experimentally observed phenomenon of "crossover" from shunt to transmit mode, in which, at short distances, the signal starts rising with decreased hand distance (instead of falling, as in shunt mode), cannot be modeled using spherical electrodes and hand models. It turns out that the phenomenon of crossover is specific to planar electrode geometries. The capacitance between the hand and the sense electrodes increases with decreased proximity, until the angle to the electrode becomes so acute that the capacitance starts dropping with decreased distance. Using the sphere model of the sense electrodes and the hand, one would not expect to see this crossover, because there is no angular dependence, and in fact one does not see crossover when spherical electrodes are used.

6.3.2 Finite Series of Images and Intersecting Spheres

Certain special cases involving multiple conductive objects can be handled exactly using a finite set of images. These cases occur when later images in the series precisely overlap (spatially) with earlier images. Four intersecting planes can be handled in this way. This and other examples are discussed by Maxwell.[Max73]

6.3.3 Estimating user ground coupling

The deviation from ground of the user's potential provides information about their ground coupling. A perfectly coupled user will always be at ground potential; an imperfectly coupled user may deviate from ground. By allowing the inverse search procedure the freedom to place a charge in the center of the spheres, we can effectively measure the potential of the user's body, which indirectly tells us the strength of their coupling to ground.

6.3.4 Ground plane with series of images

A ground plane in the presence of multiple spheres can be handled easily. We treat it like just another sphere (inducing image charges in it at each round that become source charges in the next round), but we use the plane image formulae instead of the sphere formulae. As we observed earlier, the plane formulae are the large radius limit of the sphere formulae, so we could even treat the ground plane as a very large sphere, simply by picking a radius that is much longer than the longest lengthscale in the problem.

6.4 3 Dimensional Position and Orientation Sensing FieldMouse

Figures 6-5 and 6-6 show a system that tracks the position and orientation of one or two hands. The system models each hand as three co-linear spheres that are constrained to be a particular distance apart. Using this model of the hand in conjunction with the series of images method described above, we can rapidly solve the forward problem of predicting the sensor values associated with any hand configuration, to any desired order of approximation (though the speed drops as the order increases).

A transmitter is treated as a point charge. A configuration of the model hands (position and orientation) determines sphere locations. From the transmitter location and strength, and the sphere sizes and locations, we can find the first (or higher) order image charges. Given the image charges, we can predict the received sensor values in several ways.

In some cases, we may wish to model the receivers as spheres, particularly if they actually are spheres, as was the case for the x,y,z only mouse shown in figure 6-4. The foil spheres are the electrodes.

6.4.1 Modeling flat electrodes

For the case of flat electrodes (which are virtual grounds) separated by ground, we would model the entire receive plane, including any ground between the electrodes, as a grounded plane. Using the method of images in the usual way, it is easy to show that a single charge of strength $-q$ and height z above the sense plane induces a charge distribution

$$\rho = \frac{2qz}{(x^2 + y^2 + z^2)^{\frac{3}{2}}} \tag{6.1}$$

in the plane. The charge on a particular electrode can be found by integrating ρ over the finite area of the electrode.

This integral can actually be done analytically, but in many cases, it is sufficient to numerically approximate the value by using the charge at the center of the electrode, or at several points on the electrode, since the field should not vary dramatically across an individual electrode.

Closed form surface charge integral

The integral of ρ with respect to x and y

$$\int \rho dx dy = q \tan^{-1}\left(\frac{xy}{z(x^2 + y^2 + z^2)^{\frac{1}{2}}}\right)$$

6.4.2 Orientation Sensing FieldMouse Details

To find the configuration of the hand or hands, we perform a Nelder-Mead search of the hand configuration, using the series of images-based forward model described above. The sensing electrodes are modeled as a ground plane, and the sensor values are predicted by evaluating equation 6.1 at the center of the electrode. The image charges induced in the hand model are treated as the sources that generate the induced charge distribution in the sensing plane. The sense plane charge distribution induced by each hand image charge can

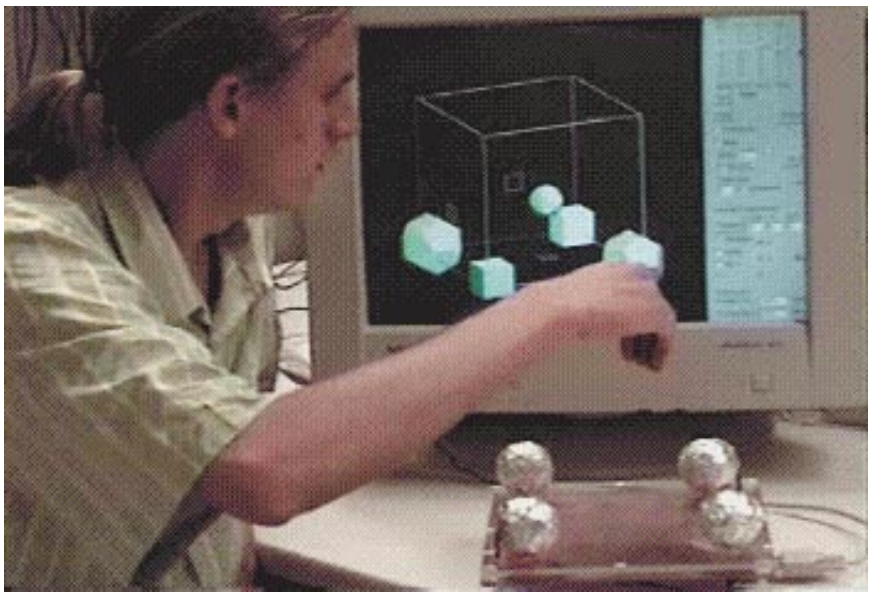

Figure 6-4: An x,y,z only mouse with the hand modeled as a single sphere. The foil electrodes are also spheres.

be superposed. To scale the forward model so that it uses the same units as the data, the demo program collects two calibration data points for each sensor pair. One of these data points is the sensor value when no hand is present—the baseline current. The other data point is the minimum sensor value. The demo program has a mode in which it collects data continuously, and keeps the minimum value on each channel. The program calculates the results of the forward model for the corresponding situations (no hand present, and hand very close by). A scale and offset can then be found, which enables the results of the numerical model to be put in the same units as the data, and also allows the zeroth order effects of conductive objects that are do not appear in the numerical model to be subtracted off.

A couple of constraints must be imposed for the search to work properly. Because of the mirror symmetry of the electrode array, hand configurations below the plane yield results identical to the corresponding above-plane configurations. The positive solution must be selected, at the graphics stage, if not during the search. A more serious problem is due to non-physical configurations, such as one sphere above the plane, one below, and one in the plane. Effectively, we must build our prior knowledge into the system. In the terminology of [Smi96], which explicitly casts the inference problem in terms of probabilities, we must impose a prior probability distribution that excludes these undesirable possibilities.

In the demo, this was accomplished by building the "prior" into the forward model. If a sphere is below the sense plane, an enormous image charge is induced on it. This results in impossibly large predicted sensor values that disagree with the data, so these undesirable solutions will be rejected.

The only difficulty with the approach is that sometimes one or more spheres will "tunnel" into the impossible region (typically just inside), and then become trapped there until the hand moves far away. From the user's point of view, the walls feel sticky, whereas it would be preferable for them to feel smooth, slippery, yet impenetrable. A more elegant implementation of the prior that produces slippery walls is a topic for future work.

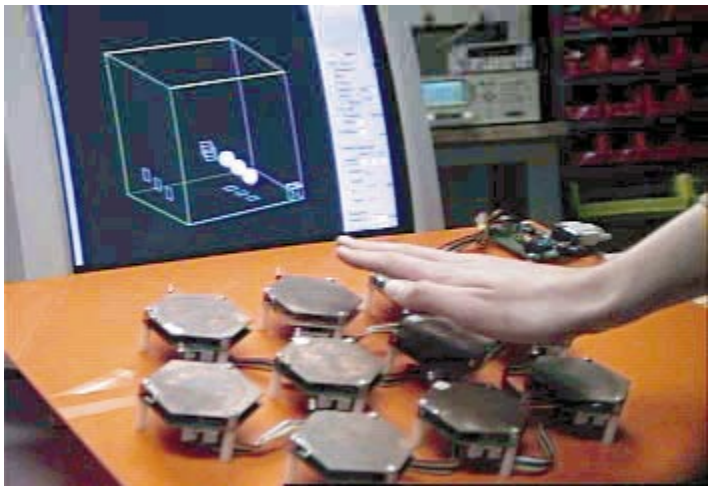

Figure 6-5: Tracking 3d position and orientation of one hand by searching a "sphere expansion" model of the hand. The School of Fish is shown in the foreground, and on the screen is the software environment for doing the inverse search and displaying the results.

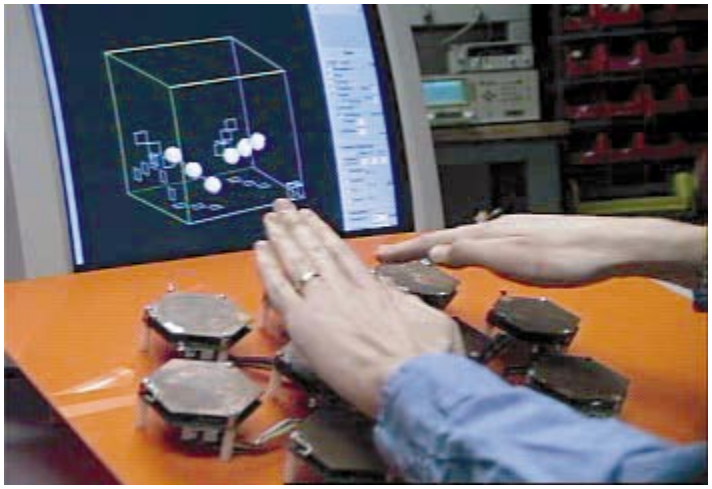

Figure 6-6: Tracking 3d position and orientation of two hands.

6.5 Electric Field Imaging and Metaballs

We approximated the geometry of the hand as a collection of spheres, and then approximately solved for the field due to those spheres in the presence of a point source. Where do these two approximations leave us? Typically, the answer would be, in a very vague and ill-defined place. However, our approximate solution of an approximation to our initial problem turns out to be the *exact* solution of a problem that may be closer to the one we really want to solve. Suppose we go to order one only. Then the collection of source and image charges implicitly define a surface (or surfaces) of $V = 0$ that in many cases turns out to be a "blobby" elongated shape that is much closer to the actual arm than the three spheres. Given a set of measurements predicted to arise from a particular model configuration, the associated $V = 0$ surface is the exact shape of the conductor that would cause those particular measurements.

Figures 6-7 and 6-8 show the exact solutions found by the method of images for two spheres. Figure 6-9 shows the first order approximate solution that results from solving the method of images problems independently, ignoring interactions between the spheres. This surface can be regarded as a (fairly poor) approximation of the actual two conductive sphere problem, but it is also an exact solution for a conductor of the shape shown. This object is clearly larger than the sum of the two spheres, which is consistent with the fact that an odd number of terms of our series of images solution always overshoots the magnitude of the deviation charge, as shown in figure 6-3. Figure 6-10 shows the second order solution, which is clearly smaller than the original conductors (though a much better approximation). Again, the small size of the object is consistent with the convergence behavior illustrated in figure 6-3. Figure 6-11 shows the first order "numerical quadrature" solution, found by taking only half the first order image charges. This is actually much closer to our desired hand shape than the two spheres.

The set of implicit surfaces defined in this way (placing point charges and plotting the $V = 0$ surface) is a very rich one that is well known in computer graphics as "metaball" modeling. It is particularly well suited for modeling the body. The point charges are typically placed along a "skeleton" with joint angles that can be articulated, and then the resulting "fleshy" $V = 0$ surface is plotted. Figures 6-13 and 6-14 show a hand modeled using metaballs, and a hand in a variety of poses modeled using metaballs.

Rich as the metaball representation may be, clearly not every surface can be represented using a small number of point charges, just some subset of all surfaces. The more charges we allow, the greater the space of possible surfaces.

Thus our sphere expansion is a strange kind of regularization. The number of conductive spheres we initially place and the geometrical constraints we impose on them are the major constraints on the space of surfaces that can be represented. The higher we go in order, the less regularization occurs. By plotting the $V = 0$ surface that corresponds to the exact solution of some electrostatic problem, we can view the members of this regularized space.

6.5.1 Integrated Sensing and Rendering

The connection between Electric Field Imaging and metaball graphics is intriguing. Modern imaging systems, for example, MRI machines used in medicine, are becoming much more tightly coupled to computer graphics. Imaging has come a long way from from ghostlike X ray images on film, or even Computed Tomography scans that, despite the intimate involvement of a computer, still use the pixel as the primitive element. One important

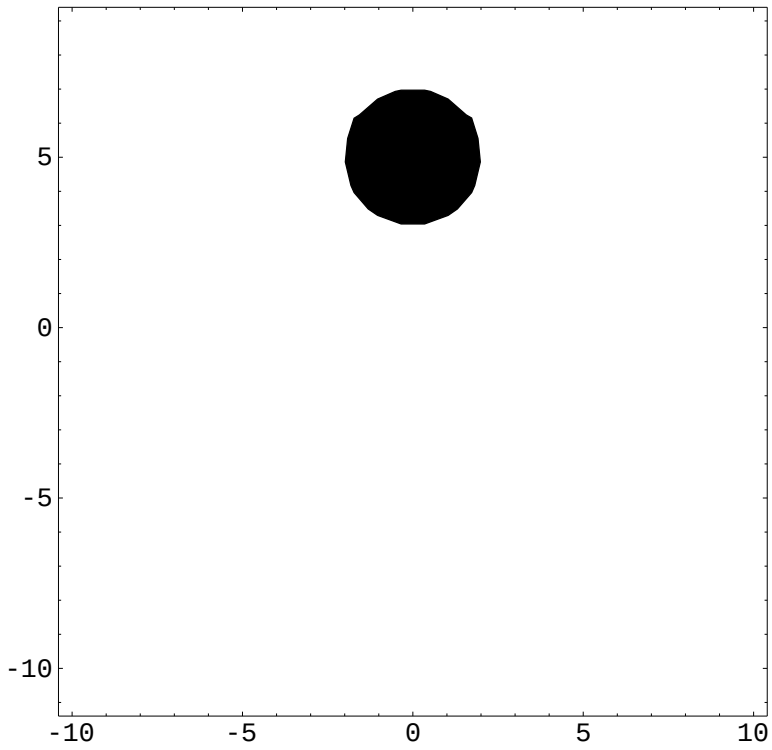

Figure 6-7: A $V = 0$ surface resulting from two appropriately placed charges. The location of the image charge was calculated using the method of images to ensure that the $V = 0$ surface has the desired radius and location.

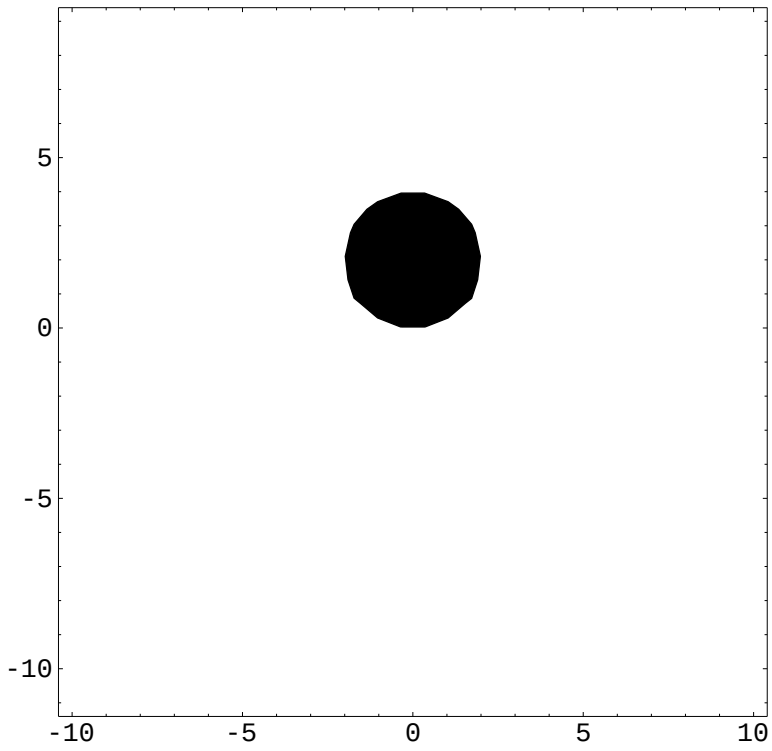

Figure 6-8: Another $V = 0$ surface constructed with two point charges using the method of images.

feature of modern medical imaging packages is extracting and displaying surfaces from voxel data. Although end users work with a high level representation (surfaces), most imaging processes still provide the underlying data in a relatively primitive form, voxels or pixels, which is why the conversion procedure is required.

There is a happy coincidence in the fact that our sphere expansion provides both a rich vocabulary for describing surfaces, and a computationally fast means for solving electrostatics problems for these surfaces. It points to an imaging method that integrates the sensing, surface representation, and surface display problems. Electric Field Imaging can effectively move straight from the raw data to the high level surface representation, without the intervening voxel level of description.

This thesis presented a method for extracting the skeleton parameters (e.g. joint angles) from electrostatic measurements. What we have not explored, but seems worth further investigation, is the possibility of extracting additional body surface geometry, represented as metaballs, from the same measurements.

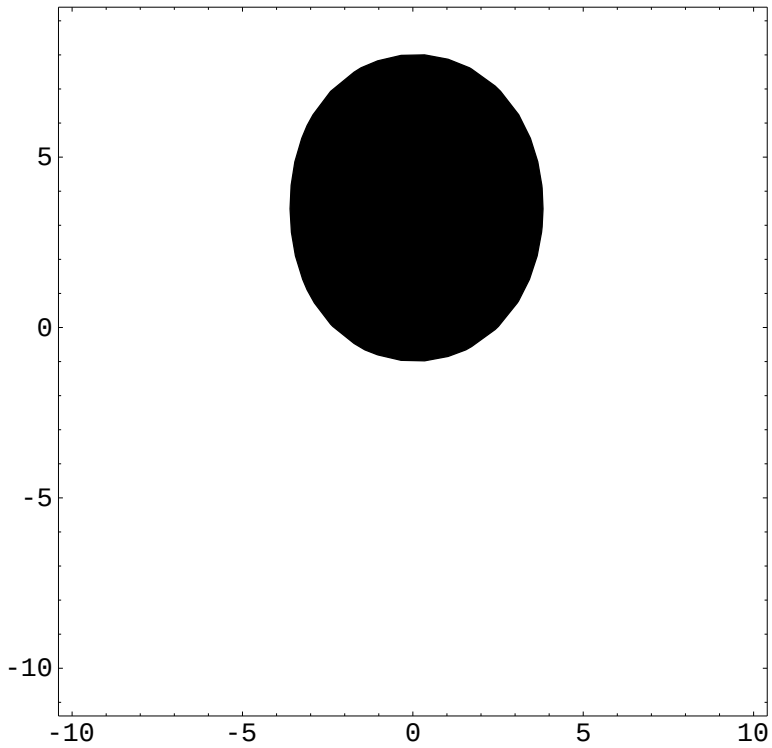

Figure 6-9: The surface that results when the source and two image charges are superposed. This is the surface implicitly defined by our first order calculation. The object is clearly larger than the two original objects. This is consistent with the fact that an odd number of terms of our series solution always overshoots the true deviation charge.

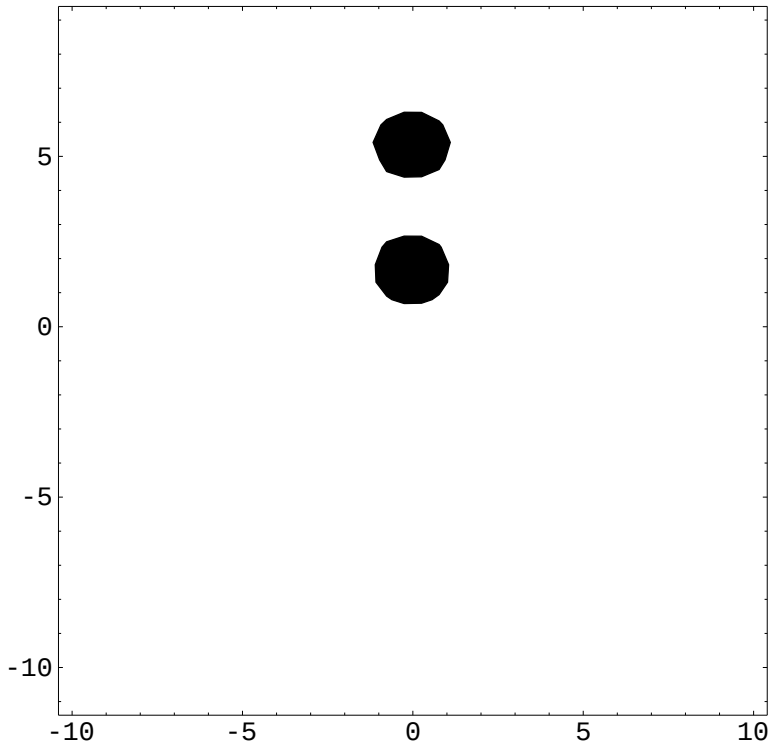

Figure 6-10: This is the surface associated with the second order solution for the two conductive spheres. Clearly the two objects are smaller than the actual conductive spheres, which makes sense: second order solutions always undershoot the true deviation charge.

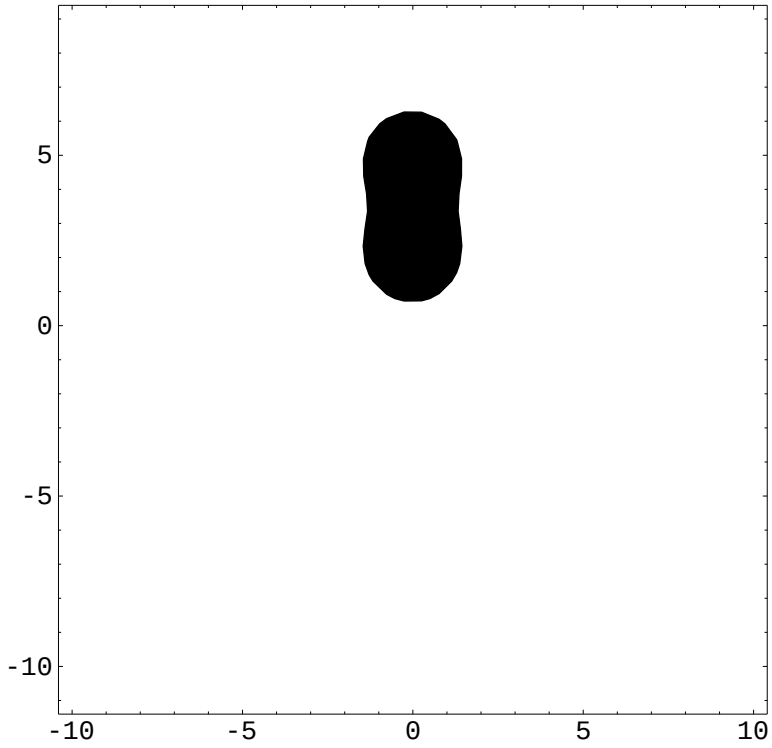

Figure 6-11: The implicit surface defined by the "numerical quadrature" first order solution, in which the strength of the highest order image charges are reduced by one half.

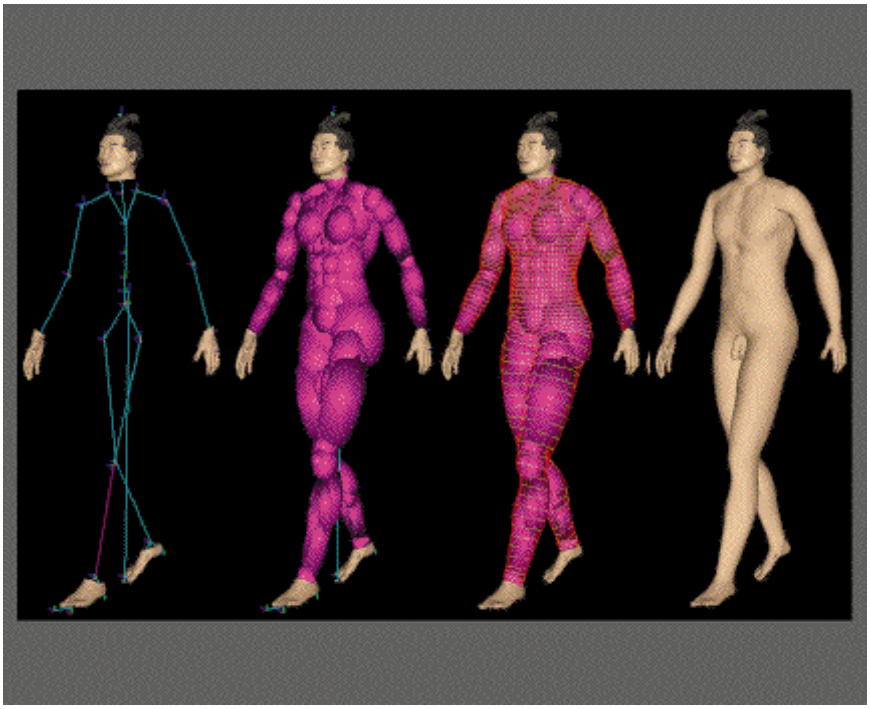

Figure 6-12: Steps in metaball modeling of the human body. First a skeleton is defined (first image). Then individual metaballs (point charges) are put in place and sized. At this stage, the $V = 0$ surfaces are plotted, ignoring interactions among the charges (second image). In the third image, the $V = 0$ surface for the superposition of all the charges is plotted.

Figure 6-13: A hand modeled with metaballs.

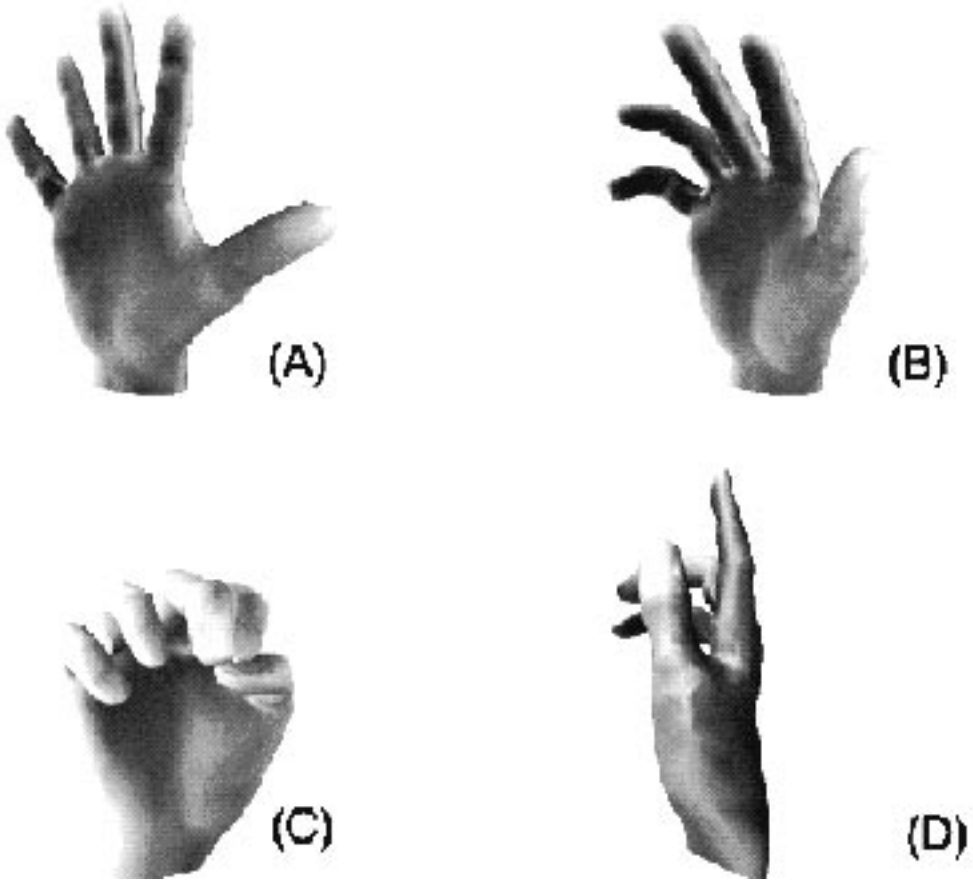

Figure 6-14: A hand in four configurations modeled with metaballs.

Chapter 7

Applications

This Chapter describes a variety of applications of the LazyFish or the School of Fish. Most of them involved collaboration with industrial or creative partners.

7.1 NEC Passenger Sensing System

The so called Rear Facing Infant Seat problem has gained a lot of media and regulatory attention in recent years. The problem is that infants in rear facing child seats have been injured and even killed by the force of the deploying airbag. Working at MIT and at an NEC Automotive Electronics plant in Atlanta, Georgia, I created a prototype smart car seat to address this problem. The seat uses electric field measurements to distinguish between an empty seat, adult passenger, forward facing infant, and rear facing infant, and disable the airbag in the case of a rear facing infant. NEC Automotive Electronics has developed this prototype into a product that is now for sale to auto makers.[GS94, GS98]

Figure 7-1 is a diagram of the smart seat. Flexible conductive fabric electrodes mounted beneath the seat cover generate the field that will be perturbed by the sensed body. The sensing circuitry is built in to the seat. Figure 7-2 shows a situation in which the airbag should not be deployed: a rear facing infant is occupying the seat, perturbing the field in a recognizable fashion.

Figure 7-3 shows the sixteen values comprising the full capacitance matrix of the four electrodes in the seat shown in 7-1 for the empty, adult, rear facing, and front facing cases. Figure 7-4 shows the author's initial application prototype for distinguishing these cases. A left or right facing infant is displayed in the rear or front facing cases, along with the NON DEPLOY message.

Based on the success of the initial prototype, NEC then built a special purpose seat with an illuminated sign to display the appropriate decision, to avoid creating the impression that a PC might be a necessary part of the sensing system. NEC presented the demonstration seat, which is shown in figure 7-5, at automotive trade shows and research conferences, such as SAE (the Society of Automotive Engineers). The baby doll in the seat is a CPR dummy filled with a 7properties. In the first image in figure 7-5, the infant is facing front, and the "Front Facing Infant" message is illuminated above. The other image shows the rear facing case.

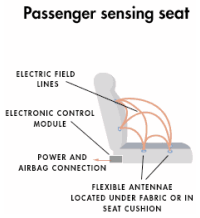

Figure 7-1: Schematic showing the components of the Passenger sensing system. The fabric electrodes in the seat generate and detect the sensing fields. The sensing circuitry is mounted in the seat.

Figure 7-2: A non-deployment situation: a rear facing infant seat.

7.1.1 Problems

In May 1996 in Detroit, half an hour before the 7:30 AM start of a press conference to introduce and promote the Passenger Sensing system within the automobile industry, the demo unit stopped working. Everyone became extremely nervous. When the team of NEC engineers turned the demo unit (shown in figure 7-5) upside down, I could see that the ground plane and sense electrodes were delaminating from the dielectric spacer separating them. When the electrodes are close to another conductor such as the ground plane, a small change in the spacing leads to a large change in the signal (recall the $\frac{1}{r}$ dependence of the parallel plate capacitor formula). I pressed the electrodes and the ground plane against the dielectric to reattach their adhesive. The demo then worked, and the press conference was able to begin on time. The only remaining hitch was a drunk reporter who wanted to know why anyone would want to detect a baby in their car—who worries about babies sneaking into cars, anyway?

This story illustrates the most significant source of electric field sensor "drift": changes in conductor geometry. It is essential when making electric field sensing systems that are designed to be robust to ensure that the electrodes cannot move, and especially that layered electrode structures cannot delaminate.

7.2 DosiPhone

A company that makes cellular phones is interested in performing an epidemiological study to determine whether cell phone usage has any adverse health effects, such as causing cancer. In order to make the study accurate, they need reliable dosimetry information: how close was the phone to the head, and how much power was the antenna transmitting? The answer to the second question is already contained in the control systems of a CDMA phone (for example), but the first question requires measuring the proximity of the phone to the head.

I mounted a LazyFish inside a cellphone housing to demonstrate the feasibility of using electric field sensing to solve the dosimetry problem. The prototype unit is shown in figure 7-6. Figure 7-7 shows the software I wrote for displaying the data. The application plots

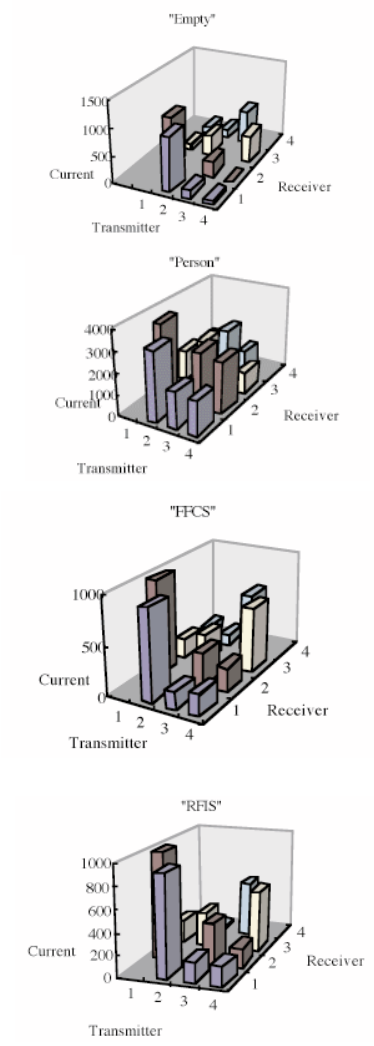

Figure 7-3: Sixteen measurements from four electrode array for four cases: empty seat, adult, forward facing child seat, rear facing child seat.

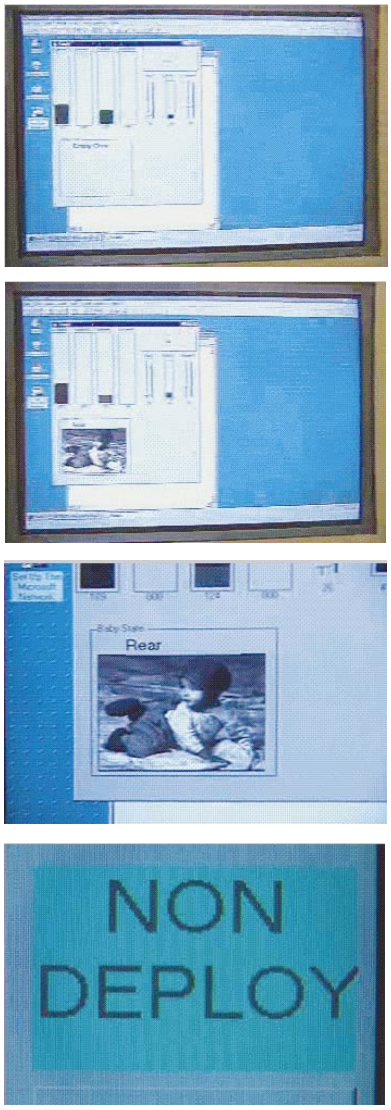

Figure 7-4: Screen shots illustrating the author's seat sensing application prototype. On screen sliders let the user interactively set thresholds to tune the decision boundaries. When the seat is empty, it displays the message "Empty," and does not display an image (top). When an infant is present, a picture of an infant is displayed, facing left if rear facing (as in the second image and the third closeup), and right in the forward facing case. When the rear facing case is detected, the "NON DEPLOY" message is also displayed (bottom image).

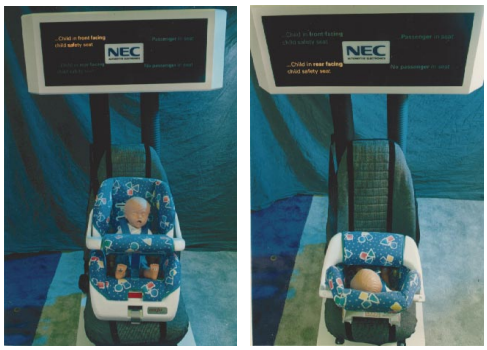

Figure 7-5: NEC's demonstration seat. The sign above reads either "Empty," "Adult," "Rear Facing Infant," or "Forward Facing Infant." Left: infant dummy in front facing configuration. Right: infant dummy in rear facing configuration.

the raw sensor value, and the total of all previously displayed values (the integrated dose).

7.3 Autoanswer Cellphone Prototype ("Phone Thing")

The goal of the Autoanswer cellphone prototype was to demonstrate that a phone can be "aware" of its physical state in relation to its user. Its ringing behavior, for example, might depend on its proximity to the user, or parts of the user. Clearly the phone should not ring loudly when it is already pressed against its user's ear. To answer the phone, it might only be necessary to place the phone against your ear.

The unit detects three conditions (states): not being held, being held aware from the head, being held close to the head. When the unit is not being held (state 0), it is silent. When it is picked up and held but not close to the head (transition from state 0 to state 1), it begins beeping regularly. When it moves close to the head (transition from state 1 to state 2), the beeps suddenly stop, and begin again when the unit moves away from the head (transition from state 2 to state 1). When the unit is released (transition from state 1 to state 0) the unit issues a single lower frequency beep and then is silent.

7.4 Alien Staff

This section describes a collaboration with Krzysztof Wodiczko. For several years, Wodiczko has been designing communication devices for strangers: Alien Staff and Mouthpiece are portable and wearable storytelling prostheses and legal communication instruments especially designed for immigrants. These instruments give the singular operator-immigrant a chance to address directly anyone in the city who may be attracted by the symbolic form of the equipment, by the character of the broadcast program, and by the live presence and performance of its owner. The stranger becomes an expert and a virtuoso in the technology of speech, better equipped than others, who have yet to overcome speechlessness in their encounters with strangers.

Prior to the collaboration, Wodiczko made several variants of each instrument. These instruments use analog video as the output mechanism and incorporate no sensing technology. The focus of the collaboration has been gesture technology: the goal is to create instruments of increased versatility and functionality, capable of sensing and responding

Figure 7-6: The Dosiphone demonstration unit in operation.

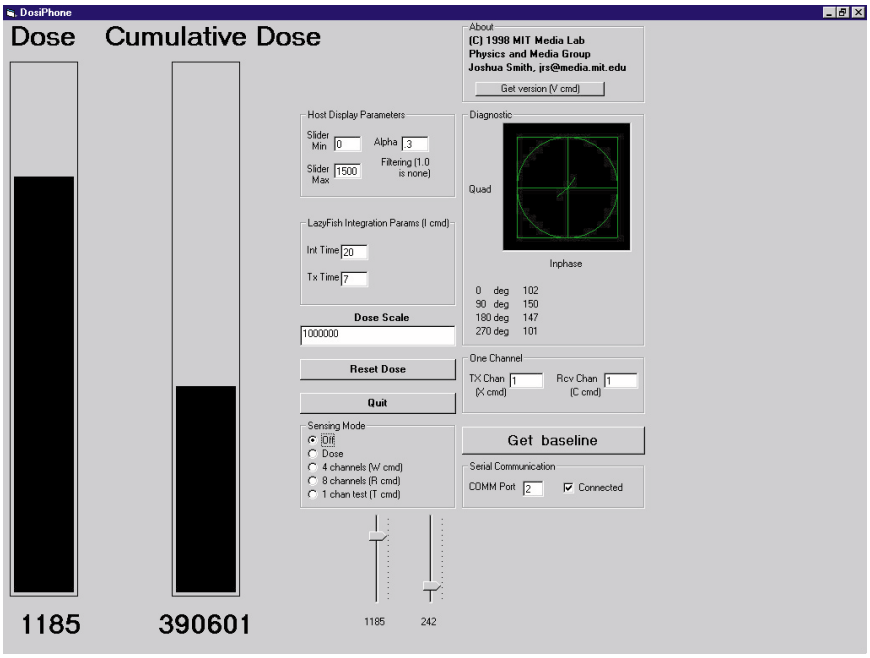

Figure 7-7: The Dosiphone software.

Figure 7-8: The Phone Thing demonstration unit. From a sensing point of view, this is quite similar to the DosiPhone application. However, this demo uses no PC—it's output is entirely aural. The microcontroller code in the Phone Thing is resolving the continuous data into one of three discrete states (distinguishing 3 states), like the car seat, rather than recording a continuous quantity, like the DosiPhone. Because the PhoneThing is not connected to a PC, the sensor system is definitely not making use of the PC's connection to ground. Thus it demonstrates the feasibility of field sensing for battery powered devices. Note that the more strongly the user's body is coupled to battery ground, the better the sensor works. In this case, a piece of copper tape placed where the user's hand should be provides a DC connection to battery ground.

to gestures of the owner and interlocutors. The new instruments use digital video as the output medium and for input incorporate the Physics and Media Groups Electric Field Sensing technology. At a coarse scale, the new systems must allow the operator to select the material being played and guide the course of the broadcast program; at fine scale, the desire is to transduce the most microscopic, nuanced bodily gestures into the output stream. The gesture-sensing instruments should provide a sense of the possibility of play, on the part of both the operator and the viewer, which should encourage viewers to approach the operator and speak.

7.4.1 Alien Staff Concept

The Alien Staff resembles a biblical shepherd's rod. It is equipped with a 3" LCD video monitor, loudspeaker, and the third variant is also fitted with electrodes for electric field sensing. Figure 2 shows the second Alien Staff variant. In the third version, a laptop is used as a digital video indexing and retrieval system that is controlled gesturally. The central part of the rod of the Alien Staff is made up of interchangeable cylindrical containers for the preservation and display of precious relics related to the various phases of the owners history, relics such as rejected visa applications, immigration and legal documents, apartment keys, old photographs and the various identity cards acquired by the owner. The early Alien Staff variants displayed the operators entire head or face. Only the operators eyes appear on the face of the latest staff. The small size of the monitor, its eye-level location and its closeness to the operator's face are important aspects of the design. Since the small image on the screen may attract attention and provoke observers to come very close to the monitor and therefore to the operator's face, the usual distance between the immigrant, the stranger, will decrease. Upon closer examination, it will become clear that the image of the eyes on the screen and the actual eyes of the person are the same. The double presence in "media" and "life" invites a new perception of a stranger as "imagined" (a character on the screen) and as "experienced" (an actor off-stage—a real life person). Since both the imagination and the experience of the view are increasing with the decreasing distance, while the program itself reveals unexpected aspects of the alien's experience, his or her presence become both legitimate and real. This change in distance and perception might provide the ground for greater respect and self-respect, and become an inspiration for crossing the boundary between a stranger and a non-stranger.

The third Alien Staff is capable of sensing and responding to hand and body gestures of the operator or interlocutors. For this staff, we worked with Noni, and immigrant from Peru living in Boston. As a hand approaches the top relic container, the eyes that appear on the small screen open and blink. A quick up and down motion of the hand when it is in the vicinity of the top relic container triggers several story fragments prerecorded by Noni, whose eyes appear on the Alien Staff screen. The fragments of the story repeat themselves as long as a hand or body remains near the central portion of the Alien Staff. This instrument allows the operator to re-play the audio-visual content, using gestures to navigate through the pre-recorded stories, for example recalling a particular episode or moment in response to a question from an interlocutor. Pointing at a particular container recalls an experience associated with the relic inside. In addition to the main audio-visual program, the operator is able to compose a "counterpoint" of several short, repeating audio lines played against the main narrative. These audio programs are associated with the relics and may emphasize, contradict, or provide other viewpoints on the main narrative. Because these repeating "audio relics" may differ in length from one another, the juxtapositions will shift over time.

Figure 7-9: Wodiczko's first model analog video Alien Staff.

Figure 7-10: Noni with the field sensing Alien Staff.

A pair of translucent plexiglass doors can be closed, partially obscuring the relics and sensors from view. The sensors function unimpeded by the closed doors, but a small receiver wire mounted on the bottom edge of one of the doors allows the computer to detect the configuration of the doors: open, closed, or anywhere in between. The state of the door in turn affects the audio-visual program and the meaning assigned to the operator's gestures. We are still experimenting with various uses of the doors. We have used the doors to muffle the relics; in this mode, when the doors are open, the relic tracks play at a volume comparable to the main program. When the doors are closed, the gestures no longer refer to individual relics but to the entire body of the staff and the main narrative. But the doors can also be used to conceal visually while revealing through audio. In this mode, when the doors are closed, sound fragments associated with the relics become more audible, as if the relics are asking to be viewed and explored.

7.4.2 Alien Staff hardware

The electric field sensing used in the staff is a descendant of one of the first gesture-sensing technologies (and one of the first electronic musical instruments), the Theremin. Neil Gershenfeld first began exploring electric field sensing for a collaboration with composer Tod Machover and Cellist Yo-Yo Ma, and under Gershenfelds guidance the Physics and Media Group has since developed it further. In addition to being incorporated in the Alien Staff, it has been used in a musical trick with magicians Penn and Teller, and is the basis of a commercially available product, NEC Automotive Electronics Passenger Sensing System. Other products are presently in development. Electric Field Sensing is a non-contact measurement of robust bulk properties of the human body (specifically conductivity and permittivity), not ephemeral surface properties, so it is not sensitive to variations in surface properties due to lighting conditions, clothing, skin color etc.

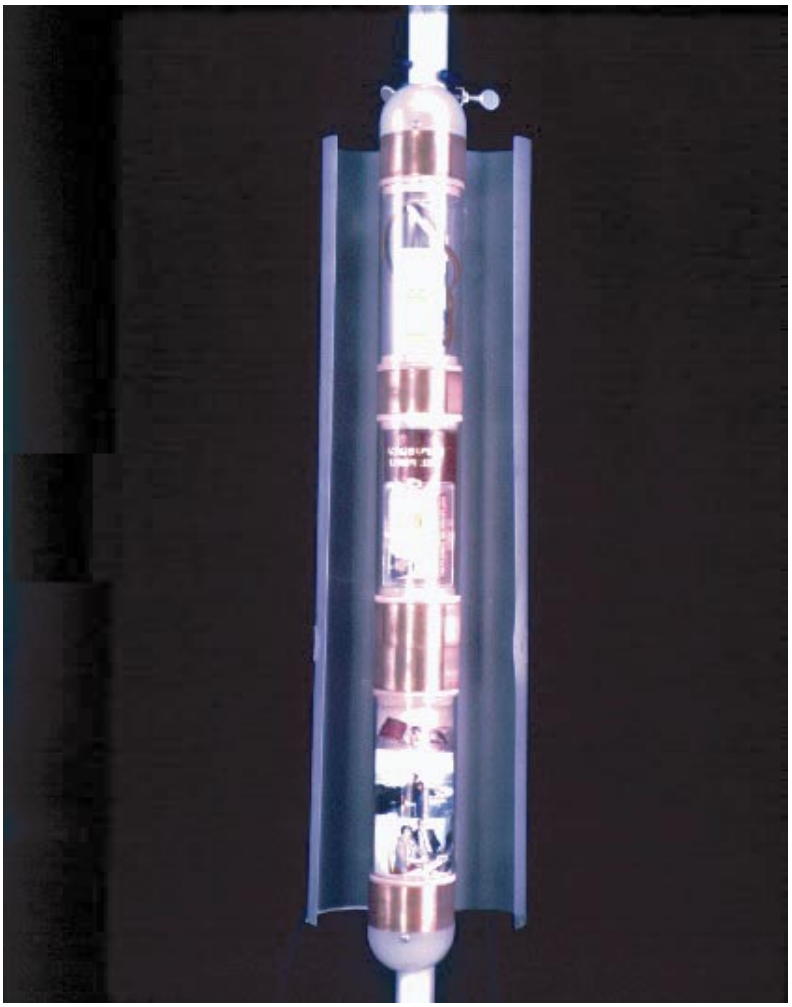

Figure 7-11: The body of the field sensing staff, with the doors open. The relic containers are bracketed above and below by field sensing electrodes.

Figure 7-12: Noni operating the staff.

In Electric Field Sensing, a low frequency (10kHz-100kHz) signal is applied to (typically one) transmit electrode, and the displacement current induced at several receive electrodes is measured. The human body is an excellent electrical conductor at these quasi-static frequencies, so when a hand or other body part approaches a transmit-receive pair, some of the displacement current is shunted to ground, and the current arriving at the receive electrode decreases. The plane halfway between a transmitter and receiver is the most sensitive. If transceiver circuitry is used (allowing the electrodes to be driven as either transmitters or receivers), then $O(n^2)$ measurements can be made with just n electrodes. From a few such measurements, it is possible to infer the position of a hand in 3 dimensions(ref jrs). An advantage of electric field sensing over video for gesture sensing is that it returns only a few numbers. With video, a lot of effort must be expended to reduce the size of the incoming data stream before meaningful interpretation can occur. In the case of EFS, the difficult data-reduction computation is effectively specified by the electrode geometry and performed by the fields themselves.

The first activated Alien Staff used the Physics and Media Group's 'Fish' board for sensing, and this variant of the Staff was designed specifically to work with the Fish. Because the most sensitive region is between a transmit-receive pair, the Staff was designed with a transmitter and receiver on either side of each relic container, so that gestures toward the relics can be registered best. Because there are three relic containers, we needed more than one transmit electrode, so we split the transmit line. These requirements, and the constraint that a transmitter cannot be directly adjacent to a receiver (because its strong signal would swamp that due to the other transmitter) led to the electrode geometry shown in Figure 5. If transceiver technology had been available, the repeated pair of electrodes would not have been necessary, since we could have quickly cycled through the electrodes, making each transmit one at a time and receiving on the nearby electrodes.

The electrodes consist of visible copper plates; the scale of the plates (both area and inter-plate spacing) is set by the human hand. The quantity that the design must optimize is contrast-to-noise, where contrast means the difference between the largest signal (which occurs when the hand is not present) and the smallest signal (which occurs when the hand is closest to the relic container, and shunting the most field). Taking a parallel plate capacitor as a crude model of a transmit-receive pair, both the largest and smallest signal increase linearly with plate area (and hence contrast and contrast-to-noise do also.)

Copper was chosen because of its good conductivity and important place in the history of electricity. Each electrode is soldered to the center conductor of a strand of RG-178 shielded co-axial cable. The sensor cables leave the body of the staff at one of two connectors mounted at the spine of the staff. Mini DIN connectors were chosen for their small size. This connector, and the other (associated with the output hardware) is the most fragile part of the Staff. A new sensor design, discussed in the section on future work, will improve this problem and allow the simplified electrode configuration mentioned above.

The head of the staff contains a 3" commercial LCD display and 3.5" speaker for audio output. The LCD power and ground, video signal and ground, and audio signal and ground leave the staff at a 6 pin mini DIN connector. The audio-visual cables and the sensing cables connect to an assortment of hardware in the operator's bag. The video connects to a (9v) power supply and a VGA to NTSC converter unit, which in turn has its own (6V) power supply. The audio signal is driven by an amplifier circuit (with 6V power supply) liberated from the powered computer-speaker enclosure that was the source of the speaker-cone in the head. The sensor cables plug into the fish board unit, which requires +12V, -12V, and +5V power supply (though regulators can be added so that it can be run single-supply).

The laptop, a Pentium P90 with 16MB of RAM running Windows 95, is connected to the sensing, audio, and video subsystems. The laptop communicates with the fish board through a DB-9 serial cable. The computer's audio mini-jack connects to the audio amplifier board; its VGA output connects to the VGA to NTSC converter unit.

7.4.3 Alien Staff Software

We had to develop the software infrastructure necessary for playing multiple audio tracks simultaneously with video. We hybridized the Microsoft AVIViewer program (which uses MMSystem calls to play a single audio track and synchs a single video to this audio track) with the audio facilities of the Microsoft Game Software Developers kit. The Game SDK allow multiple audio tracks to be mixed simultaneously, with individual control over frequency and volume. We replaced the MMsystem calls in the AVIViewer with Game SDK audio calls in order to be able to play multiple audio files simultaneously with a video.

The gesture interpretation performed by the Staff is still primitive, but because the sensors return such useful information, very simple gesture recognition goes a long way. Detecting a close approach of a hand to a relic is a simple thresholding operation. The position of the hand along the body of the staff can be approximated using the difference of the smallest sensor readings (i.e. the sensors to which the hand is closest). Because the sensor data stream is so sparse (compared with video, at least), it is easy to calculate hand velocity and in realtime recognize simple dynamic gestures such as sweeps upwards and downwards. It is also possible to estimate the radial distance of the hand from the staff (most crudely by considering the sum of the sensor values). But because of the staffs radial symmetry, essentially no azimuthal information is available.

7.4.4 Future work on the Alien Staff

Future improvements will replace the doors with illuminated relic containers. The relics will be displayed in dark plexiglass containers that will reflect outside light and conceal the contents within, but when illuminated from the inside will reveal the objects. The brightness of the illumination may be controlled by the sensors. The antenna atop the staff suggests communication and will eventually allow for the transmission and communication between other staff operators.

The present power supply is extremely unwieldy; we are interested in improving the situation. We hope to move to a better sensor design based on new Field Sensing Circuitry, the School of Fish, shown in Figures 9 and 10. The design of the School of Fish circuitry is in part a response to the challenges posed by the Alien Staff and Mouthpiece projects. The circuitry needed to drive each electrode would be mounted in the body of the staff next to the electrode. This would eliminate the present need for shielded co-ax cables carrying very small (mA), sensitive signals (and in particular would eliminate the fragile connector through which these sensitive signals leave the staff). Only power (+12V), ground, and the two RS-485 data lines would need to travel between the electrodes, and out of the staff. Because the School of Fish units are transceivers we would also be able to eliminate the "extra" 5th electrode.

7.5 MusicWear Dress

The MusicWear dress was a collaboration with fashion design students Ricardo Prado, Maria Ella Carrera, and Josefina Batres from the Domus school in Milan, Italy. The dress turns the wearer's movements into music. It was created for the MIT Media Lab Wearables fashion show in October 1997. Figure 7-13 shows the MusicWear dress, along with the Musical Jacket, in the February 1998 Vogue. Figure 7-14 is a closeup of the dress.

The dress incorporates conductive (and attractive!) fabric stripes that function as sense electrodes. The dress has a sturdy (but hot) interior fabric layer with pockets to hold School of Fish units, which connect to the fabric electrodes. A small backpack holds a MiniMidi synthesizerC, speaker, battery pack, and an extra School unit, which functions as the controller for the system. The School master in the backpack is connected to one of the other units (pupil?) by an IDC cable, which in turn connects to another unit in the dress. The IDC cable implements the usual School bus, which provides power and RS-485 differential serial data to the units.

7.5.1 Problems

The original concept for this project was to have a pair of outfits, one for a man and one for a woman, that would make music only when they came in proximity to one another, enabling a kind of joint collaborative musical performance (while hopefully avoiding an NC-17 rating). The fact that the School of Fish performs heterodyne detection (unlike the homodyne LazyFish) and therefore does not require a phase reference makes this possible without a hardwired connection between the costumes. However, the interior fabric layer in the man's pants, where the sensing equipment was housed, was so hot that the model sweated profusely in rehearsal, completely soaking the pants and all the electronics, which rendered the field sensors useless, and the man unable to perform music. The auto-musical woman's costume was a bit of last minute improvisation that did perform satisfactorily in the end.

7.6 FishFace

FishFace, a collaboration with Paul Yarin, is a platform for exploring the possibilities of coupling an inexpensive, fast, low resolution display with electric field sensing. The display is an 8x8 matrix of LEDs. Tom White developed several software applications for this platform, including "Dot," "Cross," "Zoom," "Radar," and "Pong."

Dot simply displays a single dot that (roughly) tracks the position of the user's finger. Cross displays a horizontal and vertical line whose intersection point tracks the user's finger. Zoom displays a square centered on the user's finger. The size of the square is determined by the distance of the finger, growing larger as the finger approaches. In the Radar mode, an LED "beacon" rotates around the display surface. In locations where the finger has been since the last sweep, the LEDs remain on until the next sweep. Pong mode is an implementation of the classic video game Pong (also known as Breakout). The horizontal position of the finger controls a paddle that is used to bounce a ball to the front wall, where it knocks out bricks. The player must not let the ball hit the back wall.

On several occasions, a FishFace unit, running one of these programs, was set up behind the glass window of a locked office. Passersby were able to play with the device from outside

WATCH OUT

Computer Couture

A Levi's jacket programmed to play tunes? An interactive digital dress? **Elizabeth Thrasher** checks out the new cyberstyles.

Power dressing once meant dressing for success. Well, tighten your Hermès belt and hold on to that Prada bag—an electrifying (literally) new line of clothing powered by microchips measured in megabytes is in the making at the Massachusetts Institute of Technology. It started several years ago with the MIT Media Lab's Things That Think consortium, a fashion-oriented project that focused on incorporating computers into all forms of clothing. But last October, Alex Pentland, Ph.D., academic head of the Media Lab, invited design students from schools all over the world to create accessories and clothes that would illustrate how technology could be incorporated into our daily lives through fashion. Students from Paris, Milan, Tokyo, and New York were invited to the Cambridge campus to brainstorm with MIT students on the marriage of couture and computer. The result? A far-out fashion show including an organza ball gown that lights up when the wearer moves, a T-shirt dress with silver stripes that plays music, thanks to electronic field sensors sewn into the fabric, and a musical jean jacket with a cloth keyboard stitched onto the shoulder (the speakers are in the pockets). Too Star Trek–y? Perhaps, but companies like Armani and Tommy Hilfiger have already sent scouts up to MIT to check it out. Others, such as Levi Strauss & Co., Swatch, and Nike, have pledged money to sponsor future projects at the Media Lab (a smart investment, considering how much this kind of research could run them in the future). Pentland predicts that the item most likely to make it onto the racks of mainstream stores such as Bloomingdale's will be the musical jean jacket because it's actually wearable. The real challenge for both techno and fashion designers is to create clothes people actually want to buy. The price is also more couture than mass: A fully wired get-up could run between $500 and $6,000. Other hiccups in the fusion of fashion and technology are the communication differences between scientists and seamstresses. While fashion designers may have difficulties understanding voltmeters, tech heads could have a hard time fathoming the importance of the bias cut. But what if we don't want to become bionic fashion cyborgs? Pentland argues that we already are—wearing glasses, carrying cellular telephones, and even, to some extent, using watches. After all, 100 years ago, wearing a watch was considered futuristic.

BATTERIES INCLUDED: MIT'S WIRED WARDROBE. FROM TOP: LEVI'S JACKET WITH SYNTHESIZER; ELECTRIC ORGANZA GOWN; MINIDRESS WITH MUSICAL STRIPES

Vogue's View ▸126

Figure 7-13: The MusicWear dress and the Musical Jacket (Vogue magazine, February 1997).

Figure 7-14: Detail of the MusicWear dress.

Figure 7-15: The interior of the MusicWear backpack. On the left is the controller School of Fish unit, on the right is the MiniMidi synthesizer, and below is the speaker. The battery pack (not visible) is beneath the speaker.

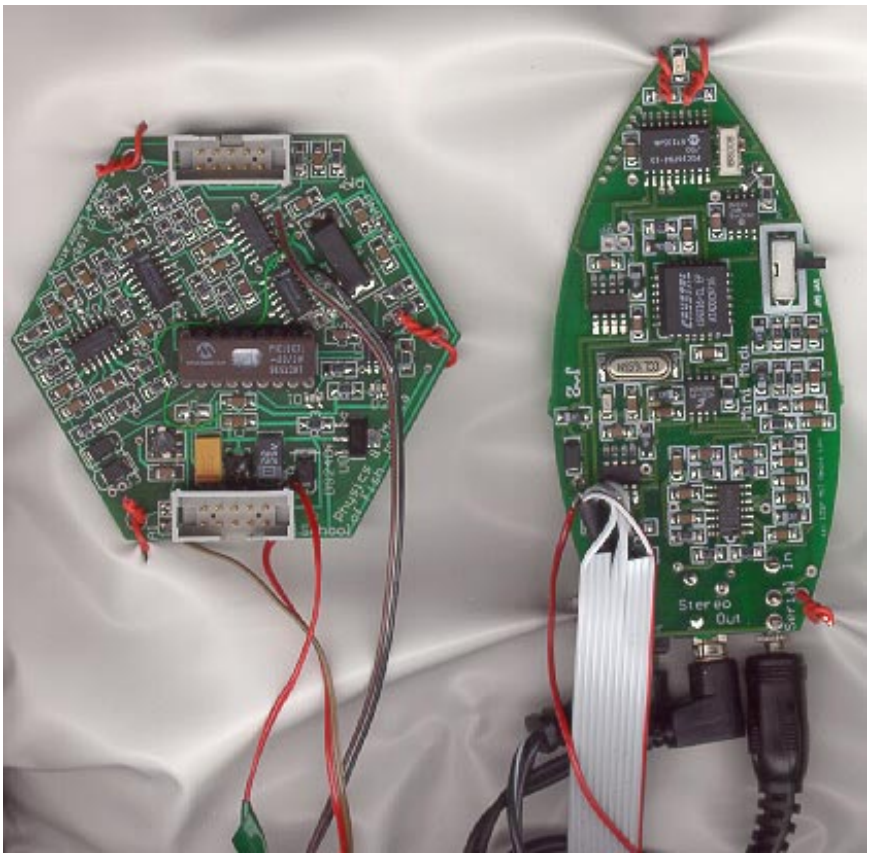

Figure 7-16: Detail of the School of Fish controller unit and MiniMidi synth.

the office, right through the glass, since the glass is transparent to both the sensing and display fields.

7.7 LaZmouse

The LaZmouse consists of a LazyFish (or parts of a LazyFish) mounted inside an ordinary 2d mouse. Electrodes at the back of the mouse measure the height of the heel of the hand above the case. The fingers remain in their usual position, so the mouse can be moved around, and the buttons in front can still be used. The 3 control degrees of freedom, combined with the buttons, make this a truly usable 3d input device.

Figure 7-17 shows the guts of the device. The board in the top half of the photo is the unmodified mouse circuitry. The circuit in the bottom half of the photo is the LazyFish, cut down so that only 1 resonator remains. The portion of the board embedded in the mouse supports 1 transmit and 2 receive channels. The reflective foil in the photo is a piece of metalized mylar used to shield the electric field sensing circuitry from noise produced by the digital electronics in the mouse, and from the effects of conductors beneath the mouse (for example, a metal desk). There are three electrodes: a transmitter is located in the center of the mouse, flanked by one receiver to the left, and one to the right. Though the device measures two channels, the 3-D mouse demo software only uses one of these.

7.8 David Small Talmud Browser

For David Small's Ph.D. thesis defense, he made a special physical user interface device, shown in figure 7-19, for "browsing" (in three dimensions) the Talmud, plus associated texts: the Torah, and a commentary on the Talmud by Emmanuel Levinas. The browser device incorporates a flat panel display, magnetic position and orientation sensors, and a variety of buttons, sliders, and knobs. Each of the three texts has its own set of physical controls. I helped Dave add a LazyFish to the browser, in order to imperceptibly detect the proximity of the hand to the various physical controls. When the hand comes near the Levinas controls, the Levinas text comes into focus in the foreground. When the hand moves to the Torah, that text comes into focus.

7.8.1 Problems

An unusual problem arose in integrating the LazyFish into the Talmud browser. The laser-cut black cardboard front plate, which hides the electrodes, turned out to be quite a good conductor, which disrupted the sensor performance. To avoid this problem, Dave had to switch to another color of cardboard.

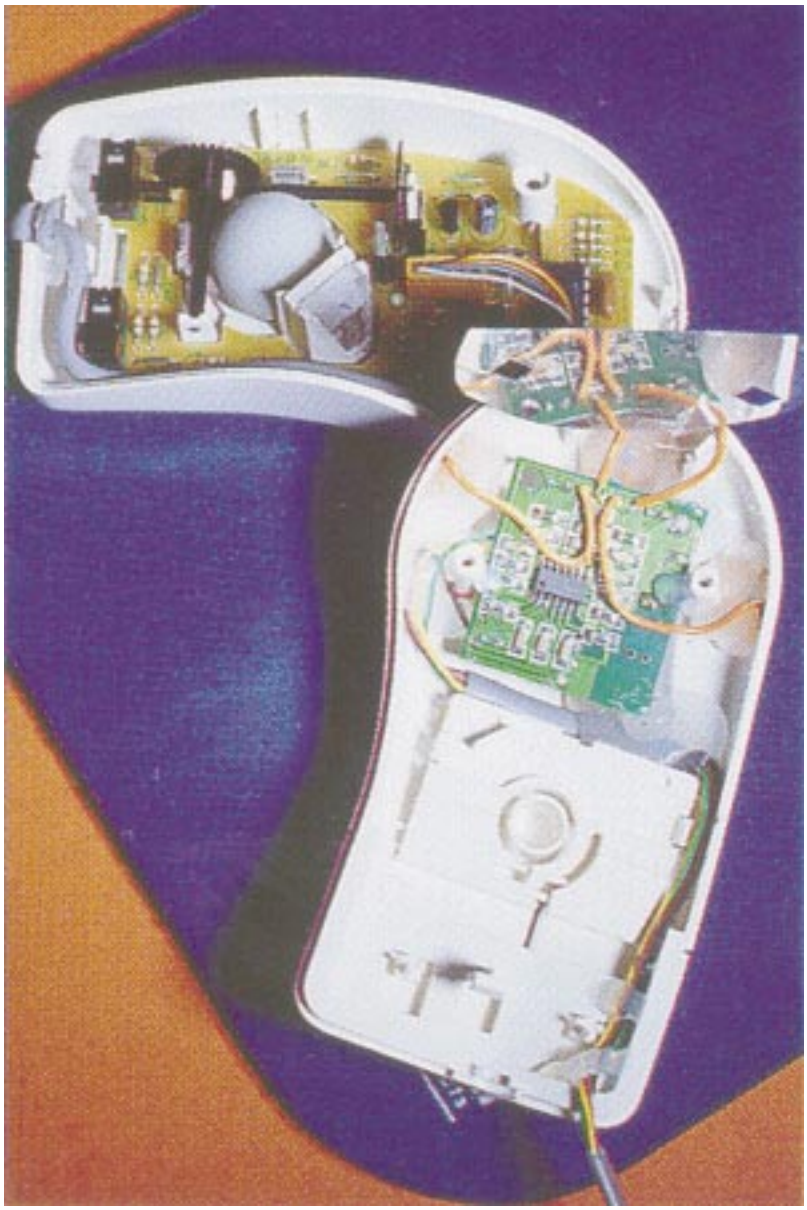

Figure 7-17: The guts of the LaZmouse.

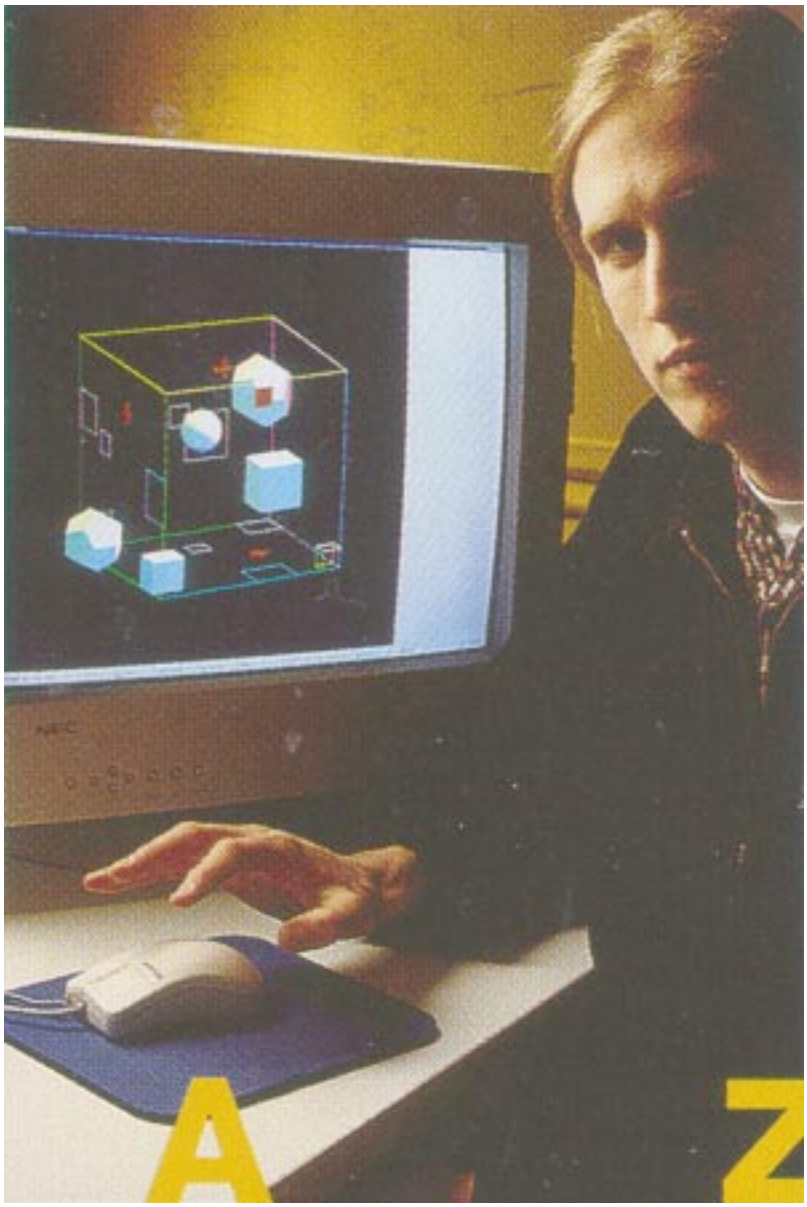

Figure 7-18: The author using the LaZmouse.

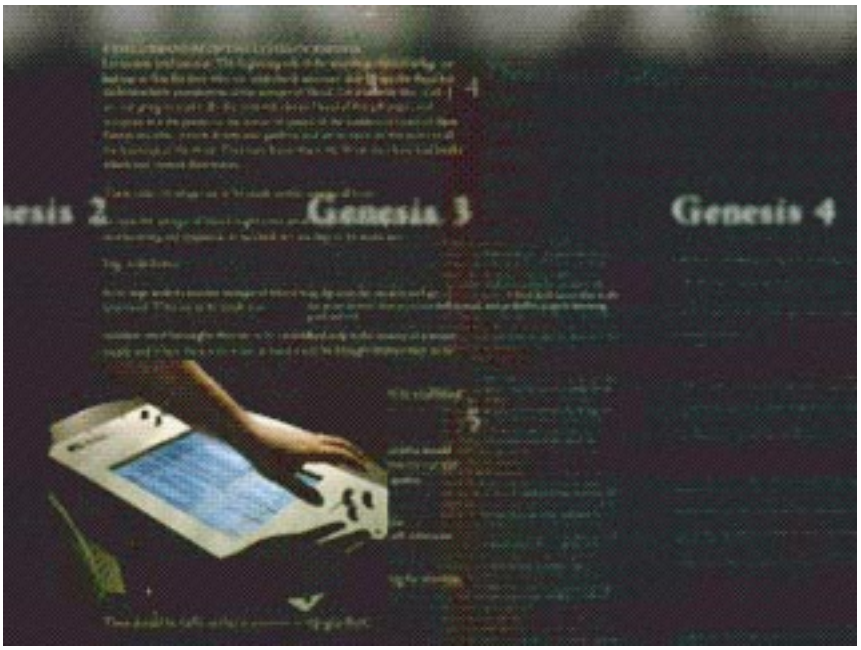

Figure 7-19: Dave Small's electric field sensing Talmud browsing workstation.

Chapter 8

Code Division Multiplexing of a Sensor Channel: A Software Implementation

The next three chapters present extensions and additional applications of some of the ideas presented earlier in the thesis. Chronologically, the work in this chapter is what lead to the LazyFish. This chapter describes my first experiments with software demodulation. I describe experiments with code division multiplexing of a sensor channel.

8.1 Introduction

This chapter demonstrates the use of software radio techniques in the context of sensing, rather than communications. It describes code division multiplexing (CDMA) and time division multiplexing (TDMA) of a receiver channel in an Electric Field Sensing system. The demodulation operation is performed entirely in software. Several sensor signals can simultaneously share the same analog front end hardware, including ADC, because software performs different processing operations on one set of samples to extract several distinct signals. This means that the number of channels that may simultaneously be received is not rigidly fixed by hardware. Additional receiver channels are extra processing steps on the data already being collected. Additional coded waveforms do appear as noise to the other channels (in the CDMA case), or require additional sensing time (for TDMA), so additional channels are not "free," but the number of channels is not fixed by the hardware. As long as one is able to pay the price in either signal-to-noise ratio (SNR) or measurement time, and also in processing time, additional channels may be added. Furthermore, the software receiver readily changes which particular channels it is demodulating.

The work is set in the context of Electric Field Sensing, which is introduced below, but the same principles could be applied in virtually any sensing or measurement system in which the quantity being sensed is modulated by a carrier. For example, these techniques could be applied to systems of ultrasound or infrared emitters and detectors.

8.2 Motivation: Electric Field Sensing

The term Electric Field Sensing refers to a family of noncontact methods for measuring the position and orientation of the human body, or parts of the human body such as a hand. Electric Field Sensing has been used for human-computer interface,[ZSP+95] to make a 3 dimensional non-contact mouse,[Smi96], for creating new musical instruments,[PG97] and in the automotive industry as a solution to the rear facing infant seat problem. [1]

In a typical implementation of electric field sensing, a low frequency (from 10-100KHz) voltage is applied to a transmit electrode (labeled T in figure 1-11), and the displacement current induced at a receiver R is synchronously detected. Figure 1-11 shows a lumped circuit model of the electrodes and the body.

The term capacitive sensing ordinarily refers to measuring the change in the loading of the transmitter as the hand approaches, increasing the value of C1. There is no distinct receiver in this measurement, so C1 is the only significant capacitance in the problem. This type of measurement is called a *loading mode* measurement.

But the other current pathways in the diagram suggest other measurement techniques. In *transmit mode*, the transmitter is coupled strongly to the body—C1 is very large—and the body is essentially at the potential of the transmitter. As the body approaches the receiver, the value of C2 (and C0—the two are not distinct in this mode) increase, and the received signal increases.

Shunt mode measurements are most relevant to this chapter. In the shunt mode regime, C0, C1, and C2 are of the same order of magnitude. As the hand approaches the transmitter and receiver, C1 increases and C0 decreases, leading to a drop in received current: the displacement current that had been flowing to the receiver is shunted by the hand to ground (hence the term shunt mode). The sensed signal is defined to be the magnitude of the decrease in received current as the hand moves in from infinity. In other words, one measures a baseline received current when the hand is at infinity, and then subtracts later readings from this baseline. Thus when the hand is at infinity, the signal is zero, and as the hand approaches the sensor, the signal increases.

With n ordinary capacitive sensors (loading mode), one can collect n numbers. These n numbers turn out to be the diagonal of the capacitance matrix for the system of electrodes. In shunt mode, one measures the $n(n-1)$ off diagonal elements.[2]

As the size of the electrode array grows, it becomes possible to infer more about the geometry of the hand or other body parts—as n gets large (say, above 10), a kind of fast, cheap, low resolution 3d imaging becomes possible.[Smi98] However, the large amounts of data are not without a cost. If one were to measure sequentially each value of the capacitance matrix, the time required would be $O(n^2)$. Fortunately, it is possible to make all the receive measurements for a single transmitter simultaneously. This brings the measurement time down to $O(n)$. One might expect the measurement time to be constant rather than linear since all the measurements are being made simultaneously. However, as will be explained in the section on resource scaling, the simultaneous measurements interfere with one another, leading to linear rather than constant scaling. But before the scaling is considered in more detail, the basics of code division multiplexing will be reviewed.

[1] Because the violent inflation of an airbag can injure infants in rear facing infant seats, it is desirable to sense the orientation of the child and disable the airbag as appropriate, as described in section 7.1.

[2] Because the capacitance matrix is symmetrical, there are ideally only $\frac{1}{2}n(n-1)$ distinct values. In practice, apparent deviations from symmetry can be used to calibrate the measurement system.

8.3 Direct Sequence Spread Spectrum and Code Division Multiplexing

In Direct Sequence Spread Spectrum[Dix94, SOSL94], the signal is modulated with a pseudo random carrier, usually generated by a maximum length Linear Feedback Shift Register (LFSR). With Code Division Multiple Access (CDMA), multiple users share the same physical channel by choosing different coded waveforms. Another common channel sharing technique is Time Division Multiple Access, in which transmitters avoid transmitting simultaneously.

The simple implementation of direct sequence spread spectrum for Electric Field sensing can be understood in terms of linear algebra. Each measurement is made using a single short pseudorandom measurement burst, during which the hand is assumed to be stationary.

If the pseudorandom carrier signal for transmitter i at time t is $\phi_i(t)$, then the signal received on electrode j, as modified by the capacitance matrix to be measured, is

$$R_{ij}(t) = C_{ij}\phi_i(t) + N_{ij}(t),$$

where C_{ij} is a constant because the hand geometry and therefore capacitance matrix is assumed to be static on the timescale of a sensing burst, and $N_{ij}(t)$ is noise. The noise is indexed by the transmitter and receiver because for each demodulation operation, the signal from the other transmitters appears as noise. Thus the noise depends not only on the receiver, but also on which transmitter is being demodulated.

In the ideal case, the transmitted waveforms would be orthogonal to one another, so that channels do not interfere:

$$< \phi_i, \phi_j >= \frac{1}{s}\sum_{t=1}^{s} \phi_i(t)\phi_j(t) = G^2\delta_{ij},$$

where s is the number of chips in a burst and G^2 is the average power per chip. One can view time division multiplexing as a case in which the carriers do not overlap, and thus satisfy this orthogonality condition exactly.

In CDMA systems, there are nonzero cross correlations between different code sequences. This can be a serious problem in a communications scenario in which receivers may spuriously lock on to the wrong code sequence. However, in the sensing application, there is no synchronization problem because the transmitter and receiver are on the same circuit board, so the only problem is a decease in SNR due to interference between the channels.

It would be best if the basis functions were also orthogonal to the ambient noise N. In reality, they are not completely orthogonal to N; if they were, one could sense using arbitrarily little energy or time. Since the noise is uncorrelated with the carrier, then by the law of large numbers, the fluctuations around zero of the inner product of the carrier and the noise are on the scale of $\frac{1}{\sqrt{s}}$, where s is the number of samples.

$$< \phi_i, N >= \frac{1}{s}\sum_{t=1}^{s} \phi_i(t)N(t) \approx 0 \pm \frac{1}{\sqrt{s}}$$

To measure an entry in the capacitance matrix, form the inner product of the received signal R_{ij} with the transmitted signal ϕ_i.

$$D_{ij} =< R_{ij}, \phi_i >= \frac{1}{s}\sum_{t=1}^{s} R_{ij}(t)\phi_i(t) = G^2 C_{ij} \pm \frac{1}{\sqrt{s}} \tag{8.1}$$

8.4 Resource Scaling

With the code division sensor multiplexing techniques explored in this chapter, one might argue that it should be possible in constant time to collect data that completely characterizes the n^2 entries in the capacitance matrix. Each transceiver unit would simultaneously transmit its coded carrier, measure the combined signal induced by the other transmitters, and store its measurement samples in memory. The vector of samples in unit $\hat{i}$ contains a complete, but encoded, representation of all of the $C_{\hat{i}j}$ values.

Of course, the "constant time" figure ignores deviations from orthogonality of the coded waveforms. A simple algebraic argument suggests that the number of samples necessary to maintain orthogonality grows linearly with n. If sinusoids are used instead of pseudorandom sequences, then the length of the measurement vector must be equal to (or greater than) the number of sensor channel amplitudes to be extracted, since the DFT matrix is square. Thus more realistically, one can form a complete but computationally encoded representation of the capacitance matrix using measurement time proportional to n, and storage of n values in each of the n units.

So far this discussion has ignored the computational operations necessary for unit $\hat{i}$ to do signal separation, transforming the raw measurement vector into an explicit representation of the n capacitance values. The naive separation algorithm (for a single receiver unit—multiple receivers can process in parallel) requires time mT, where T is the number of samples in the measurement vector and m is the number of channels being demodulated. By the argument from the previous paragraph, T cannot be less than n, and in practice should be some constant multiple of n. In the case in which one wishes to demodulate all n channels (so that $m = n$), then the naive demodulation algorithm requires n^2 time. (It is natural to wonder whether an $n \log n$ fast spread spectrum demodulation algorithm exists, analogous to the Fast Fourier Transform, that could be used with coded rather than sinusoidal carriers. A pseudorandom generator with hidden symmetry properties would probably needed.) The naive algorithm to extract just a small number m of the possible n sensor values would require time mn.

In the time division multiplexed case, the transmitted carriers are exactly orthogonal to one another, since they do not overlap at all. As in the CDMA case, the total measurement time for one unit is proportional to n. Because the carriers are exactly orthogonal, the time required for a single measurement is constant, rather than proportional to n as in the CDMA case. The amount of computation required to demodulate TDMA is also proportional to n. Thus in the limit in which all possible measured values are actually demodulated, TDMA appears to have an advantage, requiring n rather than n^2 or $n \log n$ time. In the limit in which just a few of the possible sensor values are extracted, the algorithms scale similarly, since TDMA still requires linear measurement time, and CDMA's processing time becomes linear instead of quadratic.

In the practical examples discussed in this chapter, there were 4 transmitters and 1 receiver (rather than measuring the full 5x5 capacitance matrix of this 5 electrode system), so m was 4. The theoretical discussion above considers the measurement and processing time required to achieve the same SNR using software TDMA and CDMA. In the examples,

the same total measurement and processing time is allotted to the two schemes, so that a meaningful comparison of the resulting SNR may be made.

The next sections describe the hardware and software used to implement the sensing schemes introduced above.

8.5 Hardware

The analog front end consists only of a MAX 474 dual opamp. It is configured as a transimpedance amplifier with a 1M resistor and 22pF capacitor in its feedback network, followed by an inverting voltage gain stage with a 10K input resistor and 100K feedback resistor, for a gain of 10. Because the opamp uses a single supply (5V), but the received signals are bipolar (positive and negative), 2.5V was used as analog ground. The amplified signals are read by the analog to digital converter on the PIC16C71 microcontroller. The PIC transmits data through a Maxim MAX233 RS-232 transceiver to a host computer for display and analysis. Figure 8-1 shows the hand wired prototype board used to make the measurements described in this chapter, along with the electrodes. The LazyFish board described in chapter 2 is effectively a printed circuit implementation of this handwired board, with two analog receive channels instead of one, and resonant transmitters. By short-circuiting the inductor and removing the capacitor that comprise the tuned transmit, the LazyFish hardware can easily be used to implement CDMA sensing.

8.6 Software Demodulation

If the gain variable G from section 8.3 is unity, then the multiplications in the inner product operation described in section 8.3 become additions and subtractions. Performing the demodulation in software requires an accumulation variable in which to store the value of each inner product as it is being calculated. When the first transmitter is high, the processor adds the current ADC value to the first accumulator; if the second transmitter is low, the processor subtracts that same ADC value from the second accumulator. The same ADC value is operated upon differently for each of the demodulation calculations. The addition operation can be thought of as a multiplication by $+1$ followed by addition to the accumulator; the subtraction is a multiplication by -1 followed by addition to the accumulator. Thus the operation just described effectively takes the inner product of a vector of samples with a vector representing a transmitted waveform.

An 8 bit LFSR was used, with taps at bits 3,4,5, and 7 (counting from zero). This set of taps is known to be maximal[HH90]. The C code fragment below calculates the four LFSRs, and then sets the states of the transmit pins accordingly.

```
for(k=0;k<3;k++) { // k indexes the 4 LFSRs
  low=0;
  if(lfsr[k]&8)  // tap at bit 3
    low++;  // each addition performs XOR on low bit of low
  if(lfsr[k]&16)  // tap at bit 4
    low++;
  if(lfsr[k]&32)  // tap at bit 5
    low++;
  if(lfsr[k]&128)  // tap at bit 7
```

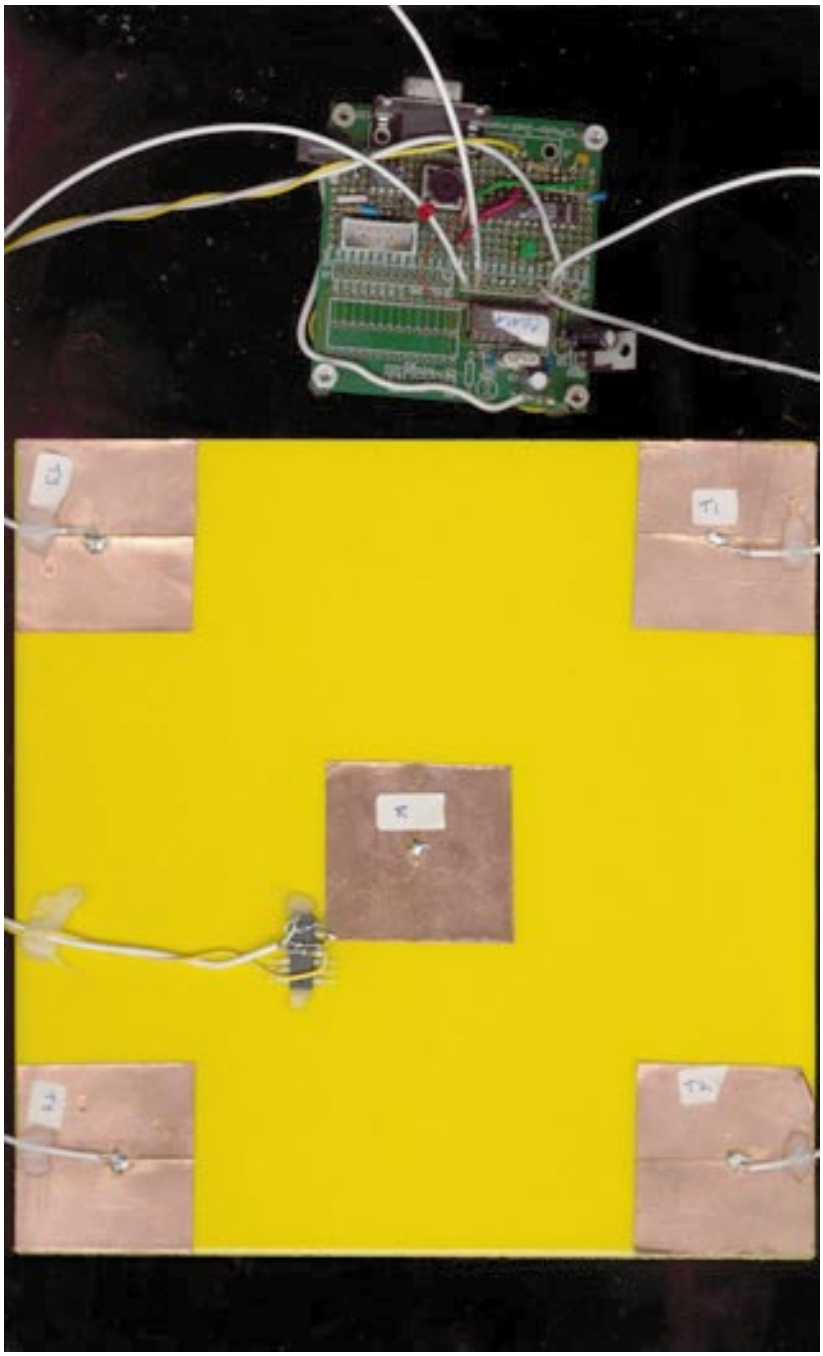

Figure 8-1: Hand wired prototype circuit board used to make measurements, together with electrode array. In the center is a receive electrode, with front end op-amp attached. In the corners are the transmit electrodes.

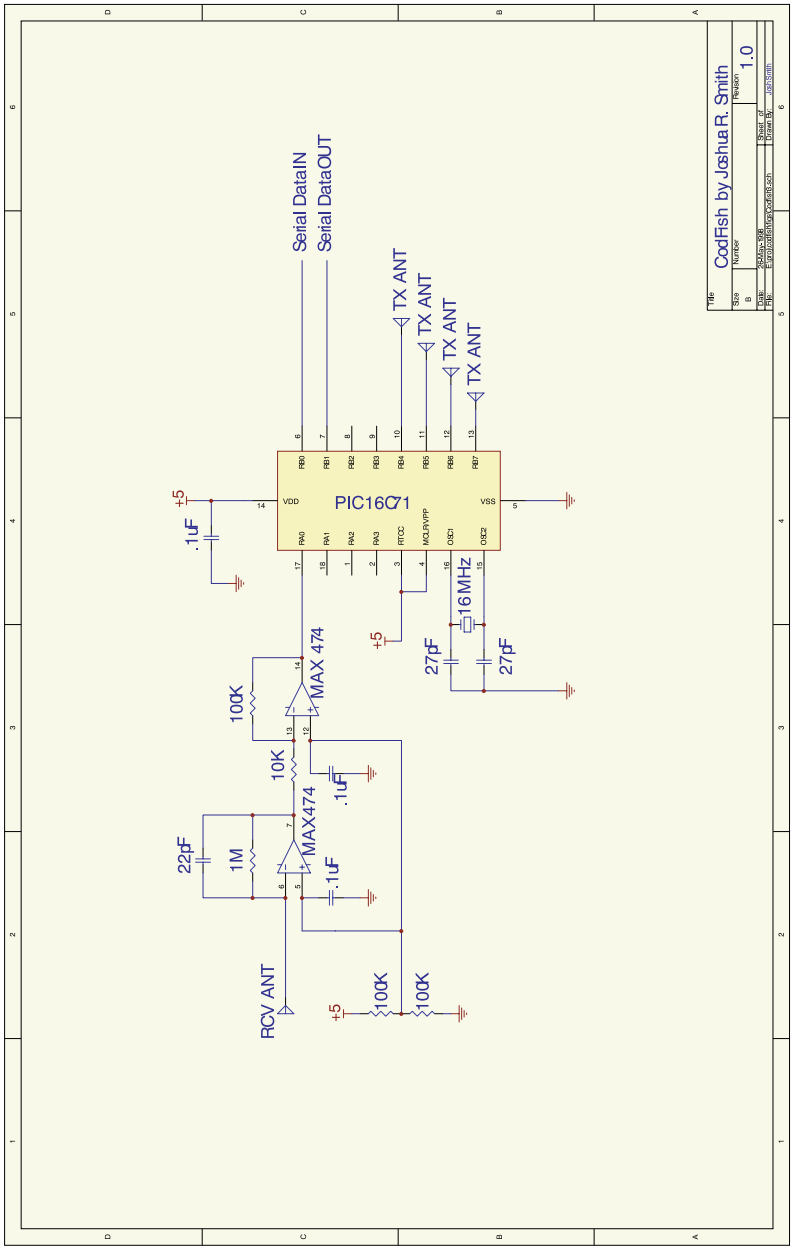

Figure 8-2: The analog front end consists of two op amp gain stages. These feed into the ADC built in to the PIC. Demodulated values are sent through a MAX233 RS232 transceiver to a host computer for display and analysis.

```
      low++;
    low&=1;  // keep only the low bit
    lfsr[k]<<=1;  // shift register up to make room for new bit
    lfsr[k]|=low;  // or new bit in
  }
  OUTPUT_BIT(TX0,lfsr[0]&1); // Transmit according to LFSR states
  OUTPUT_BIT(TX1,lfsr[1]&1);
  OUTPUT_BIT(TX2,lfsr[2]&1);
  OUTPUT_BIT(TX3,lfsr[3]&1);
```

The variable `lfsr` is an array of four bytes. Each byte in the array represents the state of one LFSR. The variable `low` also holds a byte. It is used as temporary storage to calculate the new value of the LFSR's low bit, which must be set to the sum modulo 2 of the current "tap" bits. The value of the least significant bit of `low` contains the appropriate modulo 2 sum after the compare and increment operations above. Its LSB is then masked off and ORed into the low bit of `lfsr`.

The next code fragment performs the demodulation operation. Each value returned by the ADC is operated upon differently for each channel, as specified by the appropriate LFSR.

```
  meas=READ_ADC(); // get sample
  for(k=0;k<3;k++) {
    if(lfsr[k]&1) { // check LFSR state
      next=accuml[k]; // keep backup of low byte of accum
      accuml[k]+=meas; // add measured val to accum
      if(accuml[k]<next) accumh[k]++; // if overflow, then carry
    }
    else {
      next=accuml[k];
      accuml[k]-=meas; // subtract measured val from accum
      if(accuml[k]>next) accumh[k]--; // carry if necessary
    }
  }
```

On the host computer, the final sensor value is found by dividing the accumulated value (`accumh*256 + accuml`) by s, the number of samples taken, as in equation 8.1. Because of the division by s, the fluctuations diminish as $\frac{1}{\sqrt{s}}$, but the typical magnitude of the sensed value (that is, the mean of the random variable describing the final sensed value) is independent of s.

8.7 Results: Comparison with Time Division

The performance of the CDMA and TDMA techniques was compared by measuring all four channels in 105 mS. The ADC on the current PIC16C71 can sample at about 40KHz. Calculating the 4 LFSRs, performing the ADC, and doing the 4 demodulation computations, the PIC's chip time was 136μS, or 7.4 kilochip per second. To examine the limit in which the processing time is negligible compared to the ADC time, one compares the performance of the CDMA algorithm with a TDMA algorithm generating a 3.7KHz square wave (7.4

kilochip per second). The total measurement time for the 4 time multiplexed channels was fixed at 105mS. An additional comparison was made in which the PIC generated a square wave at top speed (given the time required for analog to digital conversion and demodulation calculations), which was 13.5 KHz.

To measure the SNR on a particular channel requires calibration of the maximum and minimum observed values as a hand moved from outside the sensor's active region (defined to be zero signal), to the sensor's most sensitive point (maximum signal). With the hand outside the sensitive region, 100 samples were used to calculate the standard deviation. The SNR estimate is an average of 5 ratios of maximum signal to standard deviation.

When limited to the same chip rate, the SNR for the CDMA and TDMA approaches were as follows: for CDMA, the SNR was 340, or 51 dB, or 8.4 bits; for TDMA, the SNR was 320, or 50 dB, or 8.3 bits. When the TDMA system was operated at the maximum data rate, the SNR increased to 55 dB. This is not surprising. The total output signal power is higher in the higher chiprate case, and so the SNR should be higher. Thus, in the practical systems of today, the higher computational cost of generating the pseudo random sequence (which limits the chip rate) diminishes the performance. When the computation time becomes negligible compared to the ADC time, there is little difference between the performance of time division and code division multiplexing.

8.8 Conclusion

There are clear advantages for sensing applications of a software multiplexed analog front end over hardware multiplexing, or multiple analog channels. There are minimal requirements for special purpose hardware, and increased flexibility. Channels may be added without hardware modifications.

Comparing the two software channel sharing schemes, there are no significant performance differences between CDMA and TDMA. There is a slight practical advantage at present to TDMA. The advantage is that TDMA does not require the computation time to calculate the LFSRs. But even with present technology this difference could be eliminated by a more efficient algorithm.

Although the mean SNR values were the same for CDMA and TDMA, the variance of the SNRs for TDMA were higher. In the TDMA scheme, each single channel measurement is based on a smaller number of samples. Thus, even though the average channel properties are the same, there are larger fluctuations in the noise levels in the TDMA channel. In CDMA, the several transmit channels appear as noise to one another, but in this sensing application, with its fixed electrode geometry, the level of that self induced noise is consistent. Since each TDMA sensor value is formed from a smaller number of samples, the TDMA channels are more "bursty." Though the mean SNR is the same for the two modulation techniques, the distribution of SNR measurements are scattered more broadly about the mean for the TDMA technique than for the CDMA technique. For bulk communications purposes, the increased SNR variance does not matter, but for streamed data or sensor values, this is distinct disadvantage. The performance of a sensor system is only as good as the worst case SNR, measured in short windows of time. During a bulk communications operation like an FTP download, the short time performance of the channel is irrelevant; the relevant figure of merit is the global performance, calculated over the entire time required to complete the operation.

Apart from the subtle SNR variance advantage over TDMA, it is difficult to determine

whether the properties of CDMA sensing channels make them significantly more attractive than TDMA channels. The additional flexibility—such as the possibility of determining which sensor channels, or how many, to extract *after* the raw measurements have been made—might be useful in certain applications. For making multiple measurements with an array of sensors, the more consistent noise levels may make CDMA more attractive than TDMA. In either case, it is useful to process multiple channels of sensor data using a single analog front end, ADC, and DSP software. And while it is common practice to send multiple streams of communications data simultaneously through a single physical channel, it is historically less common (though fundamentally no different) to use a single analog front end and ADC to simultaneously process multiple sensor channels.

Despite the appeal of CDMA sensing may hold, the most practical implementation of Electric Field Sensing to date, the "LazyFish" board shown in figure 2, uses LC resonant circuits to transform the four 5V square wave microcontroller outputs into 80V sinusoidal carriers. These four transmit channels are time division multiplexed into two analog front ends. The resonators, which provide a significant increase in SNR, are useless with a spread spectrum excitation. For the practical system, the SNR advantage of resonant transmission tipped the balance decisively in favor of TDMA.

The practical system benefits from one additional application of software radio techniques. The LazyFish performs quadrature demodulation in software, which enables it to be invariant to phase shifts, such as those caused by cable capacitance. In a hardware implementation, quadrature demodulation would require almost doubling the amount of analog hardware—only the front end amplification could be shared by the inphase and quadrature channel associated with a particular receiver. Each receiver would require an additional analog multiplier and low pass filter for the additional quadrature channel. The LazyFish board fits sixteen sensing channels (counting quadrature channels) into a 1" x 2" footprint. Its small size would not have been possible without shifting the division of labor from hardware to software.

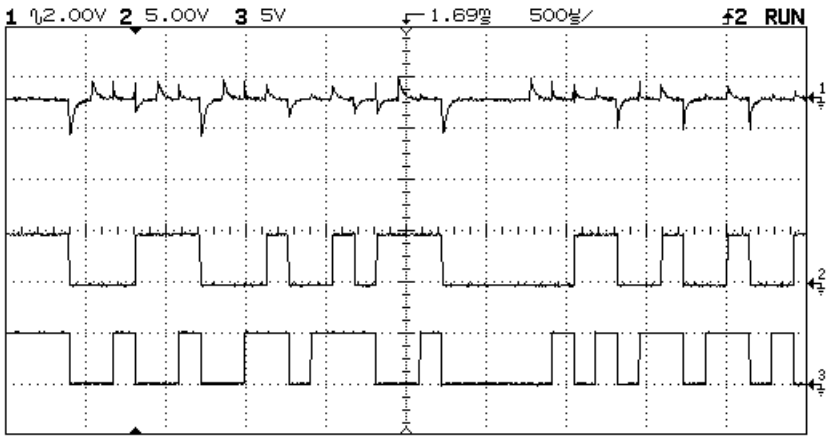

Figure 8-3: Top trace: received signal induced by 4 coded waveforms. Middle and bottom traces: two of the 4 transmit channels.

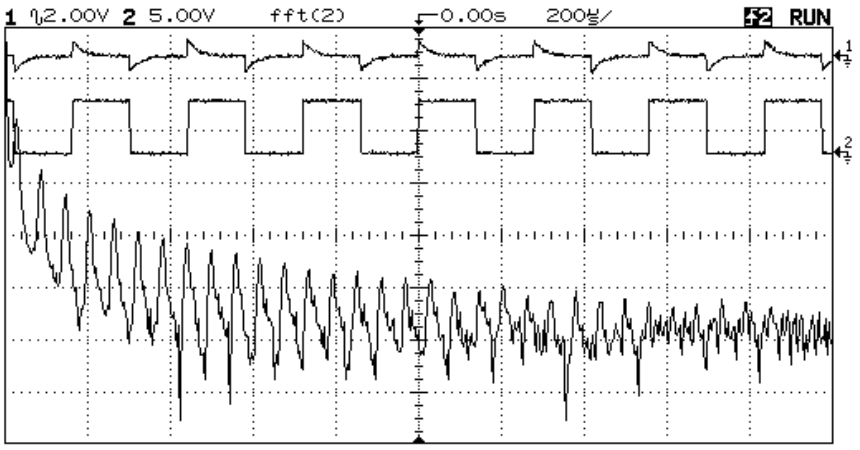

Figure 8-4: Top trace: received signal from one transmitted square wave. Middle trace: transmitted square wave. Bottom trace: Fourier transform of transmitted square wave (frequency span: 244.1KHz; 10 dB/division).

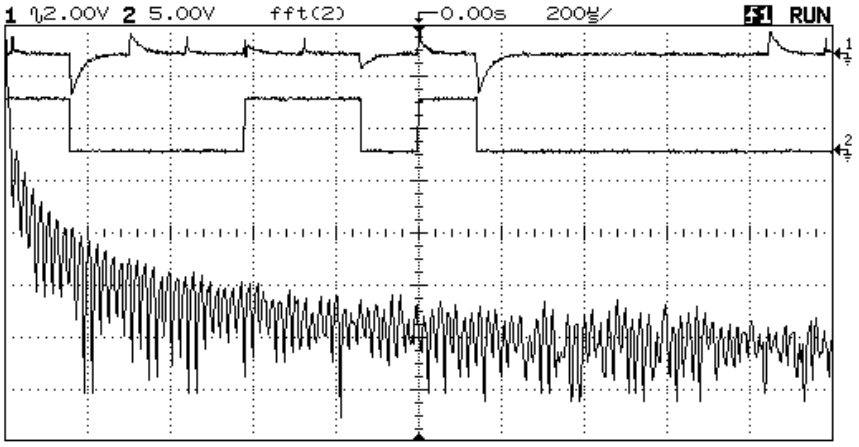

Figure 8-5: Top trace: signal received from 4 coded waveforms. Middle trace: one of the transmit channels. Bottom trace: Fourier transform of one transmit channel (frequency span: 244.1KHz; 10 dB/division). Notice how much more uniformly the LFSR generated carrier fills the bandwidth than the square wave.

Chapter 9

Modulation and Information Hiding in Images

The last chapter explored the use of spread spectrum techniques for sensing, rather than communication. The research in this chapter (which chronologically happened before the CDMA sensing work) is quite a different application of spread spectrum techniques. Here we apply spread spectrum methods to the problem of digital watermarking.

9.1 Introduction

In this paper, we discuss schemes for imperceptibly encoding extra information in an image by making small modifications to large numbers of its pixels. Potential applications include copyright protection, embedded or "in-band" captioning and indexing, and secret communication.

Ideally, one would like to find a representation that satisfies the conflicting goals of not being perceivable, and being difficult to remove, accidentally or otherwise. But because these goals *do* conflict, because it is *not* possible to simultaneously maximize robustness and imperceptibility, we will introduce a framework for quantifying the tradeoffs among three conflicting figures of merit useful for characterizing information hiding schemes: (1) capacity (the number of bits that may be hidden and then recovered) (2) robustness to accidental removal, and (3) imperceptibility. We will then present new information hiding schemes that can be tailored to trade off these figures of merit as needed in the particular application. For example, capacity may be more important in a captioning application, robustness may be most desired for copyright protection schemes, and imperceptibility might be favored in a secret communication scenario.

9.1.1 Information theoretic view of the problem

We view an image in which extra information has been embedded as an approximately continuous (in amplitude), two-dimensional, band-limited channel with large average noise power. The noise is the original unmodified image, which we will refer to as the *cover image,* and the signal is the set of small modifications introduced by the hider. The modifications encode the *embedded message*. We will refer to the modified, distribution image as the *stego-image,* following the convention suggested at the Information Hiding Workshop. From this point of view, any scheme for communicating over a continuous channel—that is, any

modulation scheme—is a potential information hiding scheme, and concepts used to analyze these schemes, such as channel capacity, ratio of signal power to noise power, and jamming margin can be invoked to quantify the trade-offs between the amount of information that can be hidden, the visibility of that information, and its robustness to removal.

9.1.2 Relationship to other approaches

In our framework, it becomes obvious why *cover image escrow* hiding schemes such as those presented in [CKLS96] and [BOD95] have high robustness to distortion. In cover image escrow schemes, the extractor is required to have the original unmodified cover image, so that the original cover image can be subtracted from the stego-image before extraction of the embedded message. Because the cover image is subtracted off before decoding, there is no noise due to the cover image itself; the only noise that must be resisted is the noise introduced by distortion such as compression, printing, and scanning. While the image escrow hiding schemes must respect the same information theoretic limits as ours, the noise in their case is very small, since it arises solely from distortions to the stego-image.

In our view, image escrow schemes are of limited interest because of their narrow range of practical applications. Since the embedded message can only be extracted by one who possesses the original, the embedded information cannot be accessed by the user. For example, it would not be possible for a user's web browser to extract and display a caption or "property of" warning embedded in a downloaded image. The need to identify the original image before extraction also precludes oblivious, batch extraction. One might desire a web crawler or search engine to automatically find all illegal copies of any one of the many images belonging to, say, a particular photo archive, or all images with a certain embedded caption, but this is not possible with cover image escrow schemes (at least not without invoking computer vision). Finally, even assuming that the cover image has been identified and subtracted out, the proof value of such a watermark is questionable at best, since an "original" can always be constructed a posteriori to make any image appear to contain any watermark. The only practical application of cover image escrow schemes we have been able to identify is fingerprinting or traitor tracing[Pfi96], in which many apparently identical copies of the cover image are distributed, but the owner wants to be able distinguish among them in order to identify users who have been giving away illegal copies.

The hiding methods presented in this paper are *oblivious*, meaning that the message can be read with no prior knowledge of the cover image. Other oblivious schemes have been proposed [BGM91, Cor95], but the information-theoretic limits on the problem have not been explicitly considered. We make comparisons between our hiding schemes and these other oblivious schemes later in the paper.

In the next section, we will estimate the amount of information that can be hidden (with minimal robustness) in an image as a function of signal-to-noise ratio. The bulk of the paper is a description of some new hiding schemes that fall short but are within a small constant factor of the theoretical hiding capacity. In the implementations of these schemes presented in this paper, we have chosen capacity over robustness, but we could have done otherwise. In the conclusion, we return to the discussion of modeling the trade offs between hiding capacity, perceptibility, and robustness using the quantities channel capacity, signal-to-noise, and process gain.

9.2 Channel Capacity

By Nyquist's theorem, the highest frequency that can be represented in our cover image is $\frac{1}{2}\frac{cycle}{pixel}$. The band of frequencies that may be represented in the image ranges from $-\frac{1}{2}\frac{cycle}{pixel}$ to $+\frac{1}{2}\frac{cycle}{pixel}$, and therefore the bandwidth W available for information hiding is $2 \times \frac{1}{2}\frac{cycle}{pixel} = 1\frac{cycle}{pixel}$.

For a channel subject to Gaussian noise, the channel capacity, which is an upper bound on the rate at which communication can reliably occur, is given by [SW49]

$$C = W \log_2(1 + \frac{S}{N})$$

Since the bandwidth W is given in units of pixel^{-1} and the base of the logarithm is 2, the channel capacity has units of bits per pixel. For some applications (particularly print) it might be desirable to specify the bandwidth in units of millimeters^{-1}, in which case the channel capacity would have units of bits per millimeter.

This formula can be rewritten to find a lower bound on the $\frac{S}{N}$ required to achieve a communication rate C given bandwidth W. Shannon proved that this lower bound is in principle tight, in the sense that there exist ideal systems capable of achieving communications rate C using only bandwidth W and signal-to-noise $\frac{S}{N}$. However, for practical systems, there is a tighter, empirically determined lower bound: given a desired communication rate C and an available bandwidth W, a message can be successfully received if the signal-to-noise ratio is at least some small *headroom factor* α above the Shannon lower bound. The headroom α is greater than 1 and typically around 3. [She95]

$$\frac{S}{N} \geq \alpha \left(2^{\frac{C}{W}} - 1\right)$$

In information hiding, $\frac{S}{N} < 1$, so $\log_2(1 + \frac{S}{N})$ may be approximated as $\frac{S/N}{\ln 2}$ or about $1.44\frac{S}{N}$.[She95] Thus $\frac{S}{N} \geq \frac{\alpha}{1.44}\frac{C}{W}$. So in the low signal-to-noise regime relevant to information hiding, channel capacity goes linearly with signal-to-noise.

The average noise power of our example cover image was measured to be 902 (in units of squared amplitude). For signal powers 1, 4, and 9 (amplitude^2), the channel capacity figures are 1.6×10^{-3} bits per pixel, 6.4×10^{-3} bits per pixel, and 1.4×10^{-2} bits per pixel. In an image of size 320×320, the upper bound on the number of bits that can be hidden and reliably recovered is then $320^2 C$. In our cover image of this size, then, using gain factors of 1, 2, and 3 (units of amplitude), the Shannon bound is 160 bits, 650 bits, and 1460 bits. With a headroom factor of $\alpha = 3$, we might realistically expect to hide 50, 210 or 490 bits using these signal levels.

9.3 Modulation Schemes

In the modulation schemes we discuss in this paper, each bit b_i is represented by some basis function ϕ_i multiplied by either positive or negative one, depending on the value of the bit. The modulated message $S(x, y)$ is added pixel-wise to the cover image $N(x, y)$ to create the stego-image $D(x, y) = S(x, y) + N(x, y)$. The modulated signal is given by

$$S(x, y) = \sum_i b_i \phi_i(x, y)$$

Our basis functions will always be chosen to be orthogonal to each other, so that embedded bits do not equivocate:

$$< \phi_i, \phi_j >= \sum_{x,y} \phi_i(x,y)\phi_j(x,y) = nG^2\delta_{ij}$$

where n is the number of pixels and G^2 is the average power per pixel of the carrier.

In the ideal case, the basis functions are also uncorrelated with (orthogonal to) the cover image N. In reality, they are not completely orthogonal to N; if they were, we could hide our signal using arbitrarily little energy, and still recover it later.

$$< \phi_i, N >= \sum_{x,y} \phi_i(x,y)N(x,y) \approx 0$$

For information hiding, basis functions that are orthogonal to typical images are needed; image coding has the opposite requirement: the ideal is a small set of basis functions that approximately spans image space. These requirements come in to conflict when an image holding hidden information is compressed: the ideal compression scheme would not be able to represent the carriers (bases) used for hiding at all.

The basis functions used in the various schemes may be organized and compared according to properties such as total power, degree of spatial spreading (or localization), and degree of spatial frequency spreading (or localization). We will now explain and compare several new image information hiding schemes, by describing the modulation functions ϕ_i used.

9.3.1 Spread Spectrum Techniques

In the spectrum-spreading techniques used in RF communications[Dix94, SOSL94], signal-to-noise is traded for bandwidth: the signal energy is spread over a wide frequency band at low SNR so that it is difficult to detect, intercept, or jam. Though the total signal power may be large, the signal to noise ratio in any band is small; this makes the signal whose spectrum has been spread difficult to detect in RF communications, and, in the context of information hiding, difficult for a human to perceive. It is the fact that the signal energy resides in all frequency bands that makes spread RF signals difficult to jam, and embedded information difficult to remove from a cover image. Compression and other degradation may remove signal energy from certain parts of the spectrum, but since the energy has been distributed everywhere, some of the signal should remain. Finally, if the key used to generate the carrier is kept secret, then in the context of either ordinary communications or data hiding, it is difficult for eavesdroppers to decode the message.

Three schemes are commonly used for spectrum spreading in RF communications: direct sequence, frequency hopping, and chirp. In the first, the signal is modulated by a function that alternates pseudo-randomly between $+G$ and $-G$, at multiples of a time constant called the chiprate. In our application, the chiprate is the pixel spacing. This pseudo-random carrier contains components of all frequencies, which is why it spreads the modulated signal's energy over a large frequency band. In frequency hopping spread spectrum, the transmitter rapidly hops from one frequency to another. The pseudo-random "key" in this case is the sequence of frequencies. As we will see, this technique can also be generalized to the spatial domain. In chirp spreading, the signal is modulated by a chirp, a function whose frequency changes with time. This technique could also be used in the spatial domain, though we

have not yet implemented it.

9.3.2 Direct-Sequence Spread Spectrum

In these schemes, the modulation function consists of a constant, integral-valued gain factor G multiplied by a pseudo-random block ϕ_i of $+1$ and -1 values. Each block ϕ_i has a distinct location in the (x, y) plane. In both versions of direct sequence spread spectrum we have considered, the blocks ϕ_i are non-overlapping (and therefore trivially orthogonal); they tile the (x, y) plane without gaps. Because distinct basis functions ϕ_i do not overlap in the x and y coordinates, we do not need to worry about interference and can write the total power

$$P \equiv \sum_{x,y}^{X,Y} (\sum_i G b_i \phi_i(x, y))^2 = \sum_i \sum_{x,y}^{X,Y} (G b_i \phi_i(x, y))^2 = G^2 XY = nG^2$$

The definition holds in general, but the first equation only holds if the ϕ_i tile the (x, y) plane without overlaps. Non-integral values of power can be implemented by "dithering": choosing step values

$$g \in (-G), (-G+1), \ldots, (-1), (0), (1), \ldots, (G-1), (G)$$

with probabilities $p(g)$ such that the average power $G^2 = \sum_g p(g) g^2$.

The embedded image is recovered by demodulating with the original modulating function. A TRUE $(+1)$ bit appears as a positive correlation value; a FALSE (-1) bit is indicated by a negative correlation value. We have found the median of the maximum and minimum correlation values to be an effective decision threshold, though it may not be optimal. For this scheme to work, at least one value of the embedded image must be TRUE and one FALSE. In the version of direct sequence data hiding presented in [Cor95], a similar problem is avoided by including 0101 at the beginning of each line.

A more sophisticated scheme would be to use a "dual-rail" representation in which each ϕ_i is broken in two pieces and modulated with $(-1)(1)$ to represent FALSE and $(1)(-1)$ to represent TRUE. Then to recover the message, each bit can be demodulated twice, once with $(-1)(1)$ and once with $(1)(-1)$. Whichever correlation value is higher gives the bit's value. This dual rail scheme also has advantages for carrier recovery.

Bender et al.'s Patchwork algorithm[BGM91] for data hiding in images can be viewed as a form of spread spectrum in which the pseudo-random carrier is sparse (is mostly 0s) and with the constraint that its integrated amplitude be zero enforced by explicit construction, rather than enforced statistically as in ordinary spread spectrum schemes.

In the Patchwork algorithm, a sequence of random pairs of pixels is chosen. The brightness value of one member of the pair is increased, and the other decreased by the same amount, G in our terminology. This leaves the total amplitude of the image (and therefore the average amplitude) unchanged. To demodulate, they find the sum $S = \sum_{i=1}^{n} a_i - b_i$, where a_i is the first pixel of pair i, and b_i is the second pixel of pair i. Notice that because addition is commutative, the order in which the pixel pairs were chosen is irrelevant. Thus the set of pixels at which single changes are made can be viewed as the non-zero entries in a single two-dimensional carrier $\phi(x, y)$. Bender et al. always modulate this carrier with a coefficient $b = 1$, but $b = -1$ could also be used. In this case, the recovered value of s would be negative. If the same pixel is chosen twice in the original formulation of the Patchwork algorithm, the result is still a carrier $\phi(x, y)$ with definite power and bandwidth.

Thus Patchwork can be viewed as a special form of spread spectrum (with extra constraints on the carrier), and evaluated quantitatively in our information-theoretic framework.

Fully Spread Version

We have implemented a "fully spread" version of direct sequence spread spectrum by choosing a different pseudo-random ϕ_i for each value of i. This fully spreads the spectrum, as the second figure in the second column of Figure 9-2 shows. The figure shows both space and spatial frequency representations of the cover image, the modulated pseudo-random carrier, and the sum of the two, the stego-image.

To extract the embedded message (to demodulate), we must first recover the carrier phase. If the image has only been cropped and translated, this can be accomplished by a two dimensional search, which is simple but effective. The point at which the cross-correlation of the stego-image and the carrier is maximized gives the relative carrier phase. We have implemented this brute force carrier phase recovery scheme, and found it to be effective. Rotation or scaling could also be overcome with more general searches.

Once the carrier has been recovered, we project the stego-image onto each basis vector ϕ_i:

$$o_i =< D, \phi_i >= \sum_{x,y} D(x,y)\phi_i(x,y)$$

and then threshold the o_i values. We have used the median of the maximum and minimum o_i value as the threshold value. Note that for this to work, there must be at least one $b_i = -1$ and one $b_i = +1$. Above we discussed more sophisticated schemes that avoid this problem. Figure 9-2 shows the original input to be embedded, the demodulated signal recovered from the stego-image, the threshold value, and the recovered original input.

Tiled Version

This scheme is identical to the "fully spread" scheme, except that the same pseudo-random sequence is used for each ϕ_i. The ϕ_i differ from one another only in their location in the (x, y) plane. Unlike the fully spread version, which is effectively a one-time pad, some information about the embedded icon is recoverable from the modulated carrier alone, without a priori knowledge of the unmodulated carrier. This information appears as the inhomogeneities in the spatial frequency plane of the modulated carrier visible in Figure 9-3. If a different icon were hidden, the inhomogeneity would look different. One advantage of the tiled scheme is that carrier recovery requires less computation, since the scale of the search is just the size of one of the ϕ_i tiles, instead of the entire (x, y) plane. Given identical transmit power, this scheme seems to be slightly more robust than the "fully spread" scheme.

These two spread spectrum techniques are resistant to JPEGing, if the modulated carrier is given enough power (or more generally, as long as the jamming margin is made high enough). With carrier recovery, the two direct sequence schemes are resistant to translation and some cropping. However, unlike the frequency hopping scheme that we will describe below, the direct sequence basis functions are fairly localized in space, so it is possible to lose some bits to cropping.

Predistortion

In addition to simply increasing the signal to improve compression immunity, Figure 9-4 illustrates a trick, called *predistortion,* for increasing the robustness of the embedded in-

formation when it is known that the image will be, for example, JPEG compressed. We generate the pseudo-random carrier, then JPEG compress the carrier by itself (before it has been modulated by the embedded information and added to the cover image), and uncompress it before modulating. The idea is to use the compression routine to filter out in advance all the power that would otherwise be lost later in the course of compression.[1] Then the gain can be increased if necessary to compensate for the power lost to compression. The once JPEGed carrier is invariant to further JPEGing using the same quality factor (except for small numerical artifacts).[2] Figure 9-4 shows both the space and spatial frequency representation of the JPEG compressed carrier. Note the suppression of high spatial frequencies. Using the same power levels, we achieved error-free decoding with this scheme, but had several errors using the usual fully spread scheme without the pre-distortion of the carrier. Tricks analogous to this are probably possible whenever the information hider has a model of the type of distortion that will be applied. Note that this version of predistortion cannot be applied to our next scheme, or to the version of direct sequence spread spectrum in [Cor95], because in these schemes carriers overlap in space and therefore interfere.

9.3.3 Frequency Hopping Spread Spectrum

This scheme produces perceptually nice results because it does not create hard edges in the space domain. However, its computational complexity, for both encoding and decoding, is higher than that of the direct sequence schemes.

Each bit is encoded in a particular spatial frequency; which bit of the embedded message is represented by which frequency is specified by the pseudo-random key. In our trial implementation of frequency hopping spread spectrum, however, we have skipped the pseudo random key, and instead chosen a fixed block of 10 by 10 spatial frequencies, one spatial frequency for each bit. One advantage of the frequency hopping scheme over the direct sequence techniques is that each bit is fully spread spatially: the bits are not spatially localized at all. This means that the scheme is robust to cropping and translation, which only induce phase shifts.

An apparent disadvantage of the frequency hopping scheme is that because the functions overlap in the space domain, the time to compute the modulated carrier appears to be kXY, where k is the number of bits, instead of just XY, the time required for the direct sequence schemes. However, the Fast Fourier Transform (more precisely, a Fast Discrete Cosine Transform) can be used to implement this scheme, reducing the time to $XY \log_2 XY$. This is a savings if $\log_2 XY < k$. In our example, $\log_2 320 \times 320 = 16.6$ and $k = 100$, so the FFT is indeed the faster implementation.

Figure 9-5 illustrates the frequency hopping modulation scheme. The results, shown in figure 9-6, are superior to the direct sequence schemes both perceptually and in terms of robustness to accidental removal. There is little need to threshold the output of the demodulator in this case. However, encoding and decoding require significantly more computation time.

[1] By compressing the carrier separately from the image, we are treating the JPEG algorithm as an operator that obeys a superposition principle, which it does in an approximate sense defined in this chapter's Appendix.

[2] It should be apparent from the description of JPEG compression in the Appendix that the output of the JPEG operator (or more precisely, the operator consisting of JPEG followed by inverse JPEG, which maps an image to an image) is an eigenfunction and in fact a fixed point of that operator, ignoring small numerical artifacts.

This scheme survived gentle JPEGing[3] with no predistortion, as illustrated in figure 9-7.[4]

A disadvantage of this scheme for some purposes is that it would be relatively easy to intentionally remove the embedded message, by applying a spatial filter of the appropriate frequency. A more secure implementation of the scheme would disperse the frequencies from one another, to make this sort of filtering operation more difficult. The main disadvantage of this scheme relative to the direct sequence schemes is that, even using the FFT, its computational complexity for encoding and decoding is greater ($XY \log XY$ rather than XY).

9.4 Discussion

We have suggested that information and communication theory are useful tools both for analyzing information hiding, and for creating new information hiding schemes. We showed how to estimate the signal-to-noise needed to hide a certain number of bits given bandwidth W. A shortcoming of our channel capacity estimate is that we used the capacity formula for a Gaussian channel, which is not the best model of the "noise" in a single image, as a glance at any of the frequency domain plots in the figures will reveal. The Gaussian channel has the same power at each frequency, but clearly these images do not, especially after compression. A more refined theory would use a better statistical model of the image channel, and would therefore be able to make better estimates of the signal-to-noise needed to hide a certain number of bits. This would also lead to better hiding schemes, since the signal energy could be distributed more effectively.

The scheme we have called "frequency hopping" is superior perceptually, and in terms of robustness to accidental removal, to the direct sequence schemes with which we experimented. Direct sequence may be less vulnerable to intentional removal, and wins in terms of computational complexity.

Assuming that the Gaussian channel approximation discussed above is not too misleading, our capacity estimates suggest that there exist significantly better schemes than we have presented, capable of hiding several hundred bits in an image in which we hid one hundred. Hybrid modulation/coding schemes such as trellis coding are a promising route toward higher hiding densities. But better models of channel noise (the noise due to cover images themselves, plus distortion) would lead immediately to better capacity estimates, and better hiding schemes.

In all the practical examples in this paper, we have tried to hide as much information as possible using a given signal-to-noise. However, keeping signal-to-noise and bandwidth fixed, communication rate can instead be traded for robustness to jamming. The quantities known as jamming margin and processing gain in spread spectrum communication theory are helpful in capturing this notion of robustness.

Processing gain is the ratio $\frac{W}{M}$ of available bandwidth W to the bandwidth M actually

[3] All the JPEG compression reported here was done in Photoshop using the "high quality" setting.

[4] In fact, it is not possible to predistort in the frequency hopping scheme: because the basis functions overlap, the resulting interference pattern depends strongly on the particular values of the bits being encoded. There is no single pattern onto which we can project the stego-image to recover the embedded data; we must (naively) project it onto a sequence of vectors, or (more sophisticated) use the FFT. In either case the idea of predistortion does not apply, at least not in the same way it did in the non-overlapping direct sequence schemes.

needed to represent the message. Jamming margin, the useful measure of robustness, is the product of signal-to-noise and processing gain. If the actual signal-to-noise ratio is $\frac{S}{N}$, then the jamming margin or effective signal-to-noise ratio $\frac{E}{J}$ after demodulation is given by $\frac{E}{J} = \frac{W}{M}\frac{S}{N}$. So robustness may be increased either by increasing signal-to-noise (at the cost of perceptibility, as we will explain in more detail below), or by decreasing the size of the embedded message (the capacity), which increases the processing gain. For example, in the case of our direct sequence schemes, the processing gain increases when we hide fewer bits because each bit can be represented by a larger block. The Patchwork hiding scheme referred to earlier sacrifices communication rate entirely (hiding just one bit) in order to buy as much robustness as possible.

Signal-to-noise ratio provides a rough estimate of perceptibility, because, all other things being equal, the higher the signal-to-noise, the more visible the modulated carrier will be. However, keeping signal-to-noise constant, some carriers—particularly those with mid-range spatial frequencies, our experience so far suggests—will be more perceptible than others. So the crudest model of perceptibilty is simply signal-to-noise ratio; a plausible refinement might be the integral over all spatial frequencies of the signal-to-noise as a function of frequency weighted by a model of the frequency response of the human visual system. Methods for quantifying visibility to humans might be a new theoretical avenue to explore, and developing systematic methods for minimizing the visibility of hidden signals is certainly a challenge to information hiding practice. The pre-distortion technique demonstrated in this paper can be viewed as a first step in this direction, in the sense that successful compression schemes comprise implicit, algorithmic models of the human visual system (the ideal compression scheme would encompass a complete model of the human visual system). It will be interesting to watch the development of information hiding schemes and their co-evolutionary "arms race" with compression methods in the challenging environment of the human visual system.

9.5 Appendix: Approximate superposition property for JPEG operator

An operator O obeys superposition if $O\{f+g\}-(O\{f\}+O\{g\})=0$. Each coefficient generated by the JPEG operator J satisfies $-1 \leq J\{f+g\}-(J\{f\}+J\{g\}) \leq 1$. In other words, JPEGing a pair of images separately and then adding them yields a set of coefficients each of which differs by no more than one quantization level from the corresponding coefficient found by adding the images first and then JPEGing them (using the same compression parameters in both cases).

The proof is simple. For a gray scale image, the unquantized JPEG coefficients S_{ij} are found by expanding each 8×8 block in a cosine basis. The final quantized coefficients a_{ij} are found by dividing each S_{ij} by a quantization factor q_{ij} (where each q_{ij} is greater than one, since the purpose of the JPEG representation is to decrease the file size), and rounding toward zero[BH93]:

$$a_{ij} = \lfloor \frac{S_{ij}}{q_{ij}} \rfloor$$

The cosine expansion is a linear operation, and therefore obeys superposition, so (as long as $q_{ij} > 1$) we need only show that for any real numbers f and g, $-1 \leq \lfloor f+g \rfloor - \lfloor f \rfloor - \lfloor g \rfloor \leq 1$. Without loss of generality, we may take f and g to be non-negative and less than one, since the integer parts F and G of f and g satisfy $\lfloor F+G \rfloor - \lfloor F \rfloor - \lfloor G \rfloor = 0$. So, for such an f and g, $0 \leq f+g < 2$. There are now two cases to consider. If $0 \leq f+g < 1$, then $\lfloor f+g \rfloor - \lfloor f \rfloor - \lfloor g \rfloor = 0-0-0 = 0$. If $1 \leq f+g < 2$ then $\lfloor f+g \rfloor - \lfloor f \rfloor - \lfloor g \rfloor = 1-0-0 = 1$. Since $f+g < 2$, these are the only two cases. The case of f and g negative is analogous, yielding a discrepancy of either -1 or 0. The discrepancy in the case that f and g have opposite sign is less than in the same sign case. Therefore each a_{ij} coefficient produced by the JPEG operator satisfies our approximate superposition principle, $-1 \leq J\{f+g\}-(J\{f\}+J\{g\}) \leq 1$. Since each a_{ij} coefficient has a discrepancy of $+1$, 0, or -1, each S_{ij} has a discrepancy of $+q_{ij}$, 0, or $-q_{ij}$. Thus the total power of the deviation from superposition (in either the spatial frequency or pixel representation, by Parseval's theorem) is bounded above by $\sum_{ij} q_{ij}^2$. This explains why JPEGing the carrier separately from the cover image is a reasonable predistortion tactic.

Note that the more aggressive the compression (the larger the q_{ij} values), the larger the discrepancies, or deviations from superposition.

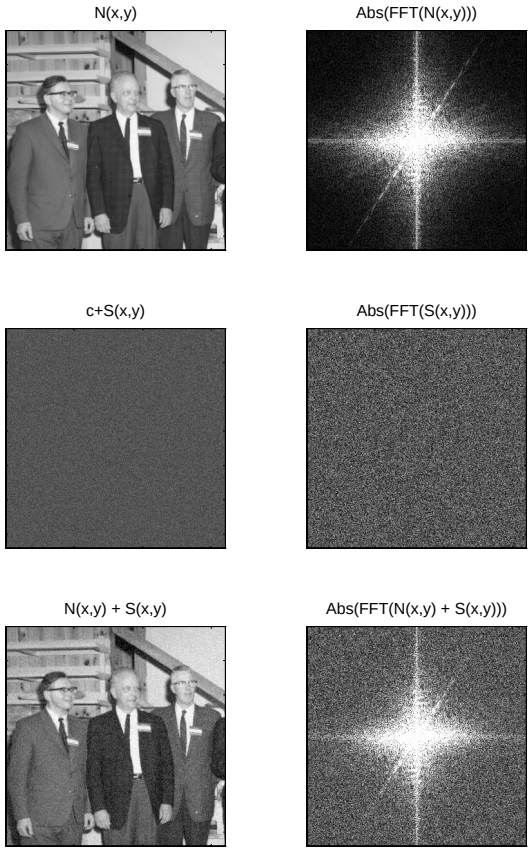

Figure 9-1: "Fully Spread" version of direct sequence spread spectrum. The left column shows (from top to bottom) the space representation of the cover image, the modulated carrier, and the stego-image. The right column is the spatial frequency representation of the same three functions. The cover image has six bits of gray scale $(0-63)$, and the power per pixel of this particular cover image, that is, the noise power per pixel, is $902 \approx 30^2$. The carrier alternates between $+2$ and -2 in this figure, so the signal power per pixel is $2^2 = 4$. We have added a constant c to the carrier to map the values into a positive gray scale.

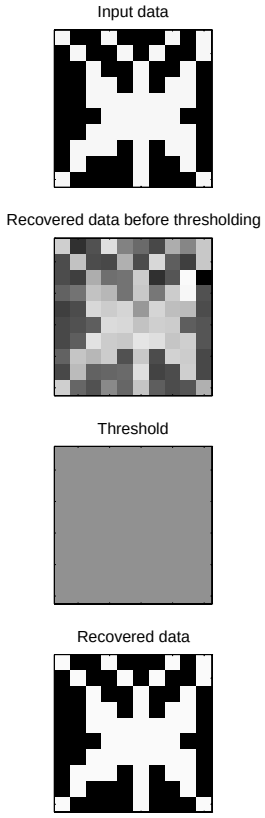

Figure 9-2: Demodulation of Fully Spread Scheme. Top: 100 bit input data icon to be embedded. Second: normalized values after demodulation. Third: threshold value. Bottom: Original input recovered by comparing demodulated values to threshold.

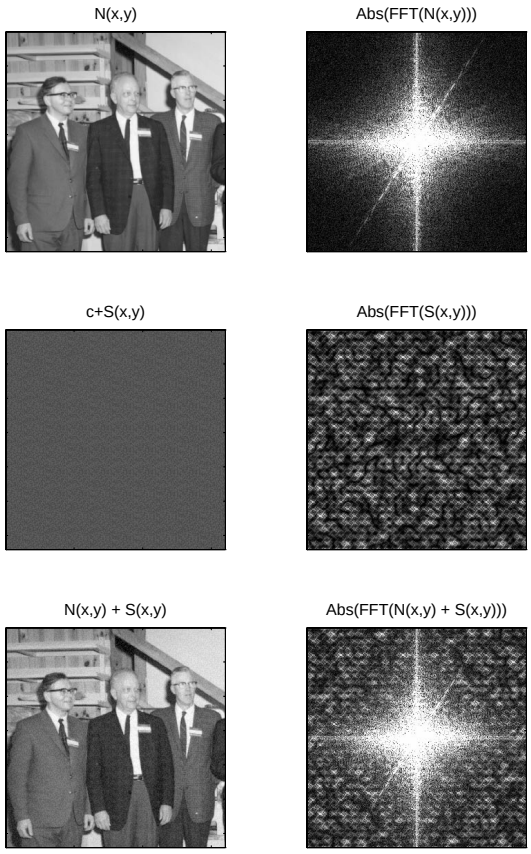

Figure 9-3: Tiled version of spread spectrum modulation scheme. Note the inhomogeneities in the spatial frequency view of the modulated carrier. As in the fully spread scheme, the noise power per pixel (the average power of the cover image) is 902, and the carrier ranges between $+2$ and -2, for a signal power of 4 per pixel.

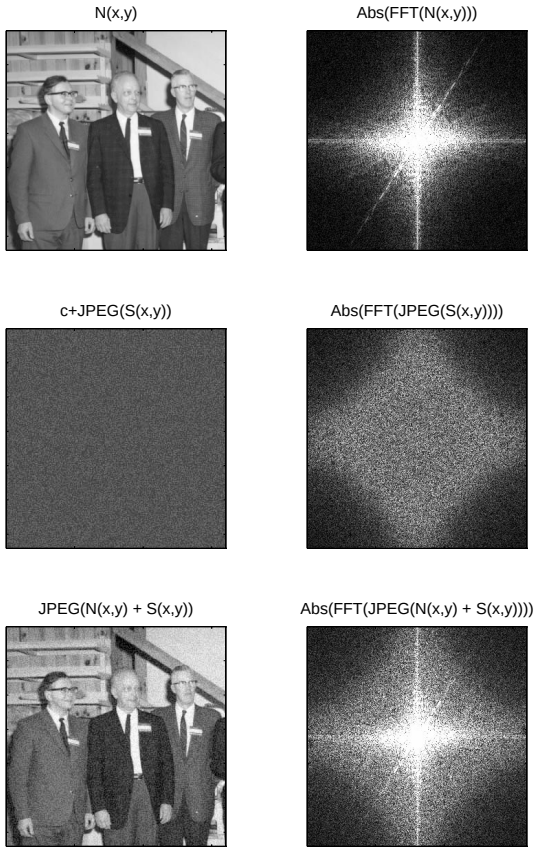

Figure 9-4: Predistortion of carrier by JPEG compression to compensate for distortion from anticipated JPEG compression. The usual direct sequence carrier has been compressed and uncompressed before being used to modulate and demodulate. JPEG compression of the same quality factor will not alter the carrier further. The original average carrier power was 16; after JPEGing the carrier by itself, the average carrier power dropped to 8.8.

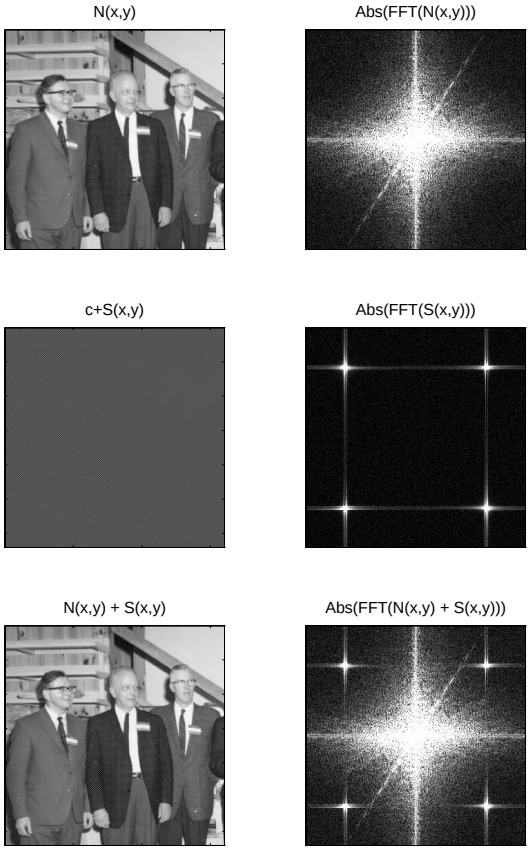

Figure 9-5: Frequency Hopping spread spectrum. Average signal power = 9.1 (units of amplitude squared), and average noise power = 902.

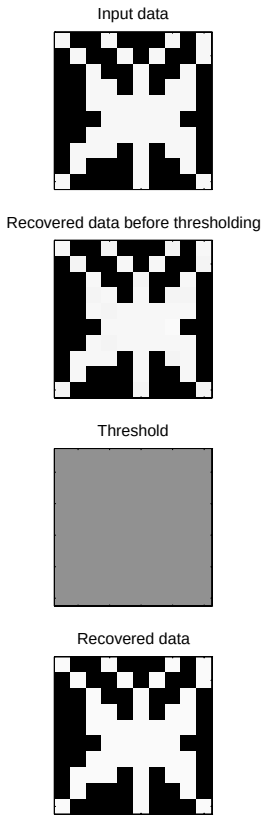

Figure 9-6: Demodulation of Frequency Hopping spread spectrum.

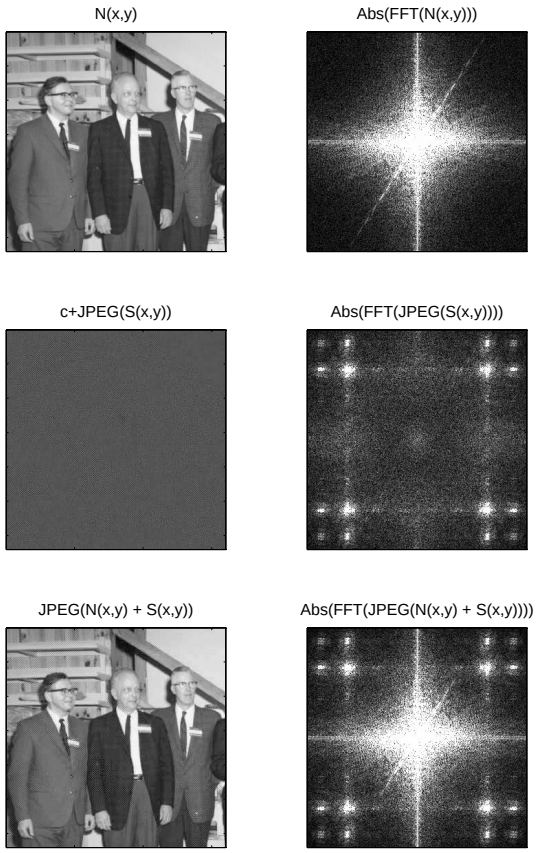

Figure 9-7: Frequency Hopping spread spectrum, with JPEGed stego-image. The stego-image D was created, JPEGed at high quality, uncompressed, and then demodulated. To estimate the amount of signal lost to compression, we measured the average power of jpeg$(N + S) - N$ and found its value to be 5.6; the power in the carrier S was 9.1, as Figure 5 showed. The carrier shown for illustration purposes in the figure, labeled $c + JPEG(S(x, y))$, is in fact JPEG$(N + S) - N$. The carrier used to create the stego-image was in fact $S(x, y)$.

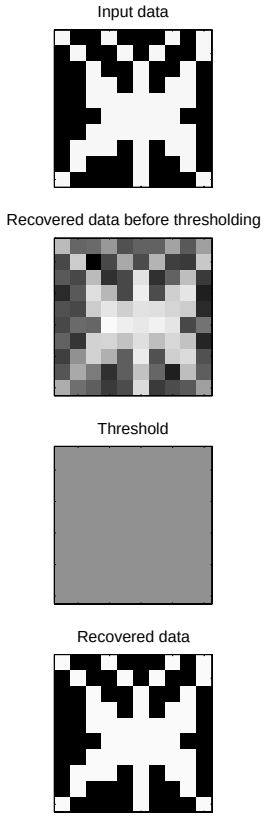

Figure 9-8: Demodulation of Frequency Hopping spread spectrum, with JPEGed stego-image. The compression took its toll: contrast this output figure with the one from figure 6, which was so robust it needed no thresholding.

Chapter 10

Distributed Protocols for ID Assignment

The work presented in this chapter grows out of the School of Fish. Each of the School of Fish units has a unique ID that typically is manually programmed in. This chapter considers distributed protocols for allowing initially indistinguishable computational units to make use of hardware random number generation and inter-unit communication to formulate a unique assignment of IDs to units.

10.1 Introduction and Motivation

In the next several years, computational, sensory, and communication capabilities will diffuse out of their present home in beige boxes on desktops and into everyday objects such as furniture, clothing, and other "non-technological" objects. As the cost of electronics continues to drop, the number of activated, networked devices will grow rapidly. Each device sharing a particular multiaccess channel will need a unique identifier for that channel. Present communication protocols such as Ethernet specify that the manufacturers must coordinate with one another in order to avoid assigning the same ID twice.[Tan89] This paper explores methods by which the *devices* could coordinate with one another to manage ID assignment dynamically and automatically.

In particular, this paper is an investigation of distributed protocols that utilize physical sources of symmetry breaking to enable a network of initially identical units to acquire the unique identities required for point-to-point communication over a shared resource such as a bus or common RF band (multiaccess channel). I present several protocols, compare their resource use (time, random bits, space, communication), and note a trade-off among these measures of algorithm complexity.

The goal, once again, is that each initially identical unit have a unique identifier by the end of the protocol. Unique identifiers, or at least sources of symmetry breaking, are necessary for point-to-point communication over a shared channel: if two units have duplicate IDs, both will respond to the same set of messages, and when both try to respond simultaneously, there is no way for them to break the deadlock, since the two units begin in the same state and proceed deterministically, in particular pausing and retransmitting at the same time. The "random exponential backoff" used to recover from collisions in CSMA/CD networks (e.g. Ethernet) is impossible without either a source of thermodynamic randomness, or an identifier such as the so-called MAC address, a unique number associated

with each Ethernet adapter, that can seed a pseudo-random number generator.

Thus the protocols must use uncorrelated noise sources, together with communication, to move the system from a perfectly correlated state (all units with ID 0, say) to create a kind of perfect anti-correlation, in which each unit is in a unique state.

An obvious method for generating a unique ID is to hardcode it into the unit's ROM. In this solution, the symmetry is broken before use, in the course of the manufacturing process. Effectively this means that the manufacturing process for each unit is slightly different. Clearly the process of manufacturing a large number of unique units is more complex than that of manufacturing a large number of identical units, and this complexity is undesirable because it translates to increased cost. For an item as expensive as an Ethernet card (around $50), the cost of the coordination activity required to ensure that each receives a unique ID is small compared to the cost of the unit. For a $5 sensor unit, this coordination cost is relatively higher, and for active RFID ($1) and passive tags ($.10), the coordination cost begins to dominate the other manufacturing costs.

Dallas Semiconductor in fact sells a "silicon serial number" (for a around $1 in large quantities) that is essentially a 56-bit unique ID that can be read by a one-wire serial protocol. Each unit made is guaranteed to have a different ID than all others.[Sem96] Effectively this manufacturer is mass producing broken symmetry, centralizing the manufacturing burden of producing unique items into a single production line.

Since the number of "smart" devices with unique IDs will be growing rapidly in the next several years as their cost drops, the idea of building the *capability* to break symmetry into each device, of moving the symmetry breaking step from the production line to the device itself, may become increasingly attractive. An interesting practical question is whether dynamic symmetry breaking schemes such as the ones we will present can be more efficient economically (i.e. cheaper) than the hard-wired, "mass produced broken symmetry" represented by Dallas' silicon serial number.

In addition to the possible cost savings, there is another practical motivation for understanding dynamic symmetry breaking protocols: they may enable anonymous communication between units, or between a unit and a public server. Since the identities are determined after the units have been deployed, there is no possibility that the manufacturer or other entity could maintain a list associating IDs with owners. However, in order to evaluate the protocols presented here from the perspective of preserving anonymity, we would have to consider the effect of adversaries who may not obey the protocol, which is beyond the scope of this paper.

10.1.1 Biological Examples

There are several well known biological examples of distributed symmetry breaking protocols. The "jamming avoidance" behavior of the South American weakly electric fish *Eigenmannia* is one such example.[Hei77] Members of this species generate continuous sinusoidal electric fields of constant frequency for both sensing and communication purposes. For a fish to sense its environment effectively, it needs its own electrolocation frequency. When the sensing frequencies of two fish overlap enough to interfere with electrolocation, both fish can sense this collision, and the resulting beat frequency. The fish with the higher frequency raises its frequency still higher, and the lower frequency fish drops its frequency, resolving the conflict. Of course, this may create new frequency conflicts, which are then resolved by the same algorithm. In this way, the frequency spectrum used by the fish can be adaptively re-allocated when a new individual is introduced into a habitat.

Another natural example that has been discussed by Rabin in the computer science literature is the case of mites of genus *Myrmoyssus*, which feed on the eardrum of *Phaenidae* moths which in turn are fed on by bats that use sonar. Since the moths use their hearing to avoid the bats, it is in the mites' interest to eat one of the moths' eardrums at a time. What protocol do they use to chose an eardrum? Each mite emits the same concentration of a pheromone. The mites move toward the eardrum with higher pheromone concentration. Thus any initial asymmetry in the mite distribution is quickly magnified until all the mites are on one eardrum. In the computer science literature, this is referred to as the Choice Coordination problem. We can take these biological examples as existence proofs of distributed coordination schemes that make use of natural symmetry breaking.

Both of these examples share certain features with the problems we discuss: the initial part of the process involves symmetry breaking, and an increase in entropy, and the final part of the process involves a decrease in entropy, and results in some coordinated behavior (correlated in the case of the mites, and anticorrelated in the case of the fish). In the mite example, the symmetry breaking consists of deciding which of two identical ears to occupy first; the coordination portion occurs when all the mites move to this ear.

10.2 The problem: Symmetry breaking and ID assignment

Like the examples mentioned above, the self-organizing ID assignment problem has two logically distinct subparts. The first part is to make all the units different in some way (to break symmetry), and the second part is to assign the unique IDs to the units, assuming that the symmetry has already been broken. (The two parts may not actually occur as two sequential phases, but they are logically separate.) In the first part, the entropy of the units' memory increases; in the second part it decreases. Self organizing processes are characterized by (local) lowering of entropy; but for this to occur in the context of our deterministic digital devices, a source of entropy, the thermal noise source, must be built in. The electric fish and the mites, by contrast, are "analog," and thus have ready access to fluctuations.

10.2.1 Related work on symmetry breaking problems

There are several problems that have been studied by computer scientists that have a symmetry breaking component to them. In all these problems, if symmetry is not broken in advance by pre-assigning a unique ID to each processor, then it must be broken in the course of the algorithm by use of independent sources of random bits.

The first well known symmetry breaking problem is the Dining Philosophers problem, posed in 1965 by Dijkstra[Dij65]. In this problem, processes with overlapping resource requirements must be scheduled. The symmetry breaking problem arises when two processors try simultaneously to access the same resource: to break or avoid deadlock, one must yield. Dijkstra proposed a randomized algorithm in which each processor waits for a random interval before trying to use the resource again.

Around 1971 a similar randomized protocol was used to avoid deadlocks after collisions in Alohanet, an early packet radio communication network, and a substantially improved, adaptive version of this scheme was developed in 1973 for Ethernet. [Tan89]

In a 1982 paper, Rabin explored randomized algorithms for the Choice Coordination Problem, in which n processors must agree on a common choice of one out of k

outcomes.[Rab82] Awerbuch *et al.* present randomized algorithms for the maximal independent set and Dining Philosophers problem for an asynchronous distributed network of processors with arbitrary network topology.[ACS94] Almost all the computer science work on symmetry breaking problems differs from ours in that it considers specific computational problems that have a symmetry breaking component. In this paper, we propose isolating the symmetry breaking problem, and solving it once and for all using an algorithm whose sole function is to assign unique identities. These identities can then be used to break symmetry if desired, and for additional non-computational purposes such as communication.

The one example in the computer science literature of using a symmetry breaking protocol to assign unique identities is a paper of Lipton and Park[LP90], in which they pose what they call the Processor Identity Problem—fundamentally the same problem discussed in this paper, but set in the context of a shared memory multiprocessing system. The protocol they describe is of no use in the multiaccess-channel scenario we are considering. Like most of the other work relating to symmetry breaking in computer science, the interactions between the processors (i.e. the communication model) is very different than ours.

10.2.2 Preliminaries

In this example, we will assume there are n units with T bits of storage each. Thus each unit may be in one of $m = 2^T$ states. The three schemes we will present each assume a different model of communication. The non-interacting algorithm doesn't require any communication infrastructure to assign the IDs. The sequential algorithm assumes a "wired-or" bus, and the parallel algorithm uses what I will call an Amplitude Modulation (AM) bus. On a wired-or bus, if any unit writes a one, all see a one. On the AM bus, all the units operating at a particular frequency can write a modulated carrier (modulated by zero or one, say), and can determine the amplitude of the sum of the (up to) n signals that were written on the bus at that frequency. So, the AM bus allows many simultaneous channels (different frequencies), and in each channel a sum of the input amplitudes is performed. Thus all the units in a communicating subgroup (cell) can learn the sum of the values they wrote onto the bus, typically the sum of their random bits.

In analyzing the proposed schemes, we will consider the scaling of three quantities as a function of n and p_f: the amount of storage, time needed to assign IDs, and the number of random bits used. We will also be mindful of the amount of communication used in each scheme, though it does not appear explicitly in any formulae.

10.2.3 Non-interacting solution and the "birthday" problem

Perhaps the most obvious solution is only to have a symmetry breaking phase: the random bits chosen by a unit in the symmetry breaking phase are its ID. This seems appealing intuitively, since the probability of a particular unit choosing a particular ID is very small, 2^{-T}. However, the probability of that some collision will occur is much higher. The famous birthday paradox asserts that given a set of more than 22 people (with randomly distributed birthdays), the probability that two or more of them will share a birthday is greater then one half. In particular, a simple combinatorial argument shows that the probability of failure p_f of the non-interacting protocol is

$$p_f = 1 - \frac{m!}{(m-n)!m^n} = 1 - \frac{2^T!}{(2^T-n)!2^{Tn}}$$

There is a well known formula that bounds this probability; see for example Motwani [MR95] for a derivation.

$$p_f \leq 1 - e^{-n(n-1)/2m} = 1 - e^{-n(n-1)/2^{T+1}}$$

Solving this inequality for T, we find

$$T \geq \log_2 \left(\frac{-n^2 + n}{\ln(1 - p_f)} \right) - 1$$

Assuming that we are interested in a small probability of failure, we can make the approximation $\ln(1 - p_f) \approx -p_f$, so $T \geq \log_2 \left(\frac{n^2 - n}{p_f} \right) - 1$. We now drop the additive constant, allowing ourselves one extra bit, and also drop the linear n term, which is negligible, to find the following useful expression for the number of bits needed in the symmetry breaking ID as a function of the number of units n and the desired probability of failure p_f:

$$T \geq \log_2 \frac{n^2}{p_f}$$

Since the algorithm has no other significant storage requirements, the space requirement is also given by this expression for T. For $n = 10^6$ and $p_f = 10^-15$, $T = 90$. Since, with high probability, the symmetry has been broken (each unit has a unique identifier), a compact set of working IDs in $0, \ldots, n-1$ can now be assigned.

The working IDs can be assigned by a binary search procedure. Assume for the moment that one unit can become "bus master" (below we will present a scheme by which this could happen). The bus master can broadcast a message asking whether there are any units with ID less than $\frac{1}{2}2^T$ and wait for a response. If there is a response, it cuts its pivot point in half and asks again. We do not need to assume that the bus master can detect communication collisions (multiple responses to its queries). We have chosen T large enough that the IDs are all distinct. Instead of stopping its search as soon as it has found a range of IDs containing just a single unit, the bus master can continue its binary search until it has discovered the precise "symmetry breaking ID" of each unit. Thus it can find an all the occupied IDs.

Finding a single occupied ID takes time $\log_2 m = \log_2 2^T = T$, and since this must occur n times, the total time for this procedure is nT. Each unit can then be assigned the working ID in the range 0 to $n-1$. The time needed to break symmetry is T and time nT is needed to assign the short IDs. Thus the total time needed is $L = (n+1)T$. Using the expression for T derived above, we can write the total time

$$L = (n+1) \log_2 \frac{n^2}{p_f}$$

The total number of random bits needed is $R = nT$; this can be written

$$R = n \log_2 \frac{n^2}{p_f}$$

We will see in the next section that allowing some communication decreases the ID size requirements.

10.2.4 Sequential symmetry breaking and ID assignment

For this scheme, the communication model is assumed to be a wired-or bus, a term that was defined in section 10.2.2. The n units engage in a tournament consisting of a series of n rounds. Each round ends with one winner, who takes the next unallocated ID, and then doesn't participate in later rounds. Thus the first tournament has n participants, and the last has just 1. In the terminology of distributed algorithms, this could be described as repeated leader election.

In each round, all the participating units chose a random bit, write it on to the shared bus, and then check the value of the shared bus (I will call this a trial). Because of the wired-or bus, if any unit writes a 1, all will read a 1. So if a unit writes 0 and reads 1, it knows that at least one other unit is sharing the bus and drops out of the round. Eventually, only one unit will remain; when it has read a long enough sequence of bits that are identical to what it wrote, it can be satisfied that it is alone, so it takes the next ID and broadcasts a message for the others to begin the next tournament.

The total space required by the algorithm is simply the number of bits T needed to represent n IDs:

$$T = \log n$$

A rough analysis of the running time is as follows. There are two parts to the running time: $L = L_1 + L_2$. The first, L_1, is the expected time to produce a winner; it depends only on the number of units n. The second, L_2, is the time it takes the winner to realize that it has won. It depends only on the desired probability of failure p_f. First we will estimate L_1. Round i, which has i participating units, consists of approximately $\log_2 i$ trials, since roughly half drop out of the round after each trial. Thus the total number of trials (total time) is approximated by

$$L_1 = \sum_{i=1}^{n} \log_2 i = \log_2 \prod_{i=1}^{n} i = \log_2 n! = \frac{\ln n!}{\ln 2} \leq \frac{n \ln n}{\ln 2}$$

To find L_2, we will evaluate our our "birthday failure" expression from the previous section with $n = 2$ to find the probability of 2 or more units choosing the same sequence of L_2 random bits. The result is $L_2 = \log_2 \frac{1}{p_f}$. Thus

$$L = L_1 + L_2 \leq \frac{n \ln n}{\ln 2} + \log_2 \frac{1}{p_f}$$

The total number of random bits needed is

$$R = \sum_{i=1}^{n} i \log_2 i$$

For n sufficiently large, this sum can be approximated by an integral:

$$R \approx \int_{i=1}^{n} i \log_2 i di = \frac{1}{4}\left(n^2(\log_2 n - 1) + 1\right) \leq n^2 \log_2 2n$$

Allowing some communication—the wired OR bus—decreased the ID size needed. The next protocol significantly decreases the time needed to assign the IDs, though its bandwidth requirements are much higher.

10.2.5 Parallel symmetry breaking and ID assignment

The modest communications needed for the previous scheme allowed the shortest possible IDs to be used. The scheme that will be presented in this section brings to bear more communication resources (bandwidth), and is thereby able to assign a set of short IDs much more quickly.

The idea is to split the n units into two equal groups after they have chosen T random bits, and then repeat the process recursively. Each of these iterations assigns one ID bit to each unit. If additional communication bandwidth is available, then the units can divide into smaller and smaller communication cells: each unit's current ID specifies the frequency at which it should communicate. Initially, all the units have the same ID (0), so all share the same communication channel. The additional communications bandwidth allows the IDs to be assigned simultaneously, in parallel.

In this scheme, each unit first chooses a random bit, which splits the group roughly in half; then the deviation from a perfect split is measured using the AM bus, and then further, corrective flips occur, with probabilities chosen so as to reduce the deviation. If a proposed random step ever increases the deviation, it is rejected, so the deviation converges monotonically to zero. For the corrective flip at time t, we chose the flipping probability p_t such that the expected value of the sum after the flip is $\frac{n}{2}$. We will denote the sum of the random variables ξ, and the expected value of the sum $\mu = \frac{n}{2}$. It will be convenient to define the "excess" $s = \xi - \mu$. Without loss of generality, we can always take s to be positive; s is negative indicates an excess of 0s, and in this case we can deterministically flip all the units, so that $s' = |s|$.

The correction step at time t is a biased random walk starting from ξ_{t-1} and moving in the direction of μ. The probability $p_t(\xi_t|\xi_{t-1})$ is a Gaussian with mean $\xi_{t-1} - p_t\xi_{t-1}$, because the initial value is ξ_{t-1}, and ξ_{t-1} is also the number of units eligible to flip from 1 to 0. At every time step we must chose p_t so that the mean of $p_t(\xi_t|\xi_{t-1})$ is $\mu = \frac{n}{2}$.

Thus we solve $\xi_{t-1} - p_t\xi_{t-1} = \frac{n}{2}$ for p_t, which gives $p_t = 1 - \frac{n}{2\xi_{t-1}}$. The mean of the distribution describing the new value of a single unit that is eligible to flip is given by $\mu_1 = (0)p_t + (1)(1-p_t) = 1 - p_t$. The variance of this distribution is given by $\sigma_1^2 = (0-\mu_1)^2 p_t + (1-\mu_1)^2(1-p_t)$, which simplifies to $\sigma_1^2 = p_t(1-p_t)$. Thus if the sum before the correction flip is ξ_t, so that ξ_t units are eligible to participate in this flip, the variance of the $p_t(\xi_t|\xi_{t-1})$ distribution is $\sigma_t^2 = p_t(1-p_t)\xi_{t-1}$.

We can study the algorithm's rate of convergence by considering the deterministic dynamics of a "typical value" of the sum ξ_t. For this typical value, we will use $\xi_t = \mu + \sigma_t = \mu + (p_t(1-p_t)\xi_{t-1})^{\frac{1}{2}}$. We have an explicit expression for p_t in terms of ξ_{t-1}, which gives us an explicit difference equation for ξ_t in terms of ξ_{t-1} and n:

$$\xi_t = \frac{n}{2} + \left(\frac{n}{2} - \frac{n^2}{4\xi_{t-1}}\right)^{\frac{1}{2}} \tag{10.1}$$

This difference equation is non-linear, however, so it cannot be solved for ξ as a function of t. We could model the algorithm's dynamics by brute-force iteration of the equation, but this would not give insight into the scaling properties.

However, it may be verified that $\xi_t < \mu + \xi_0^{\frac{1}{2^t}}$, or equivalently, $s_t < n^{\frac{1}{2^t}}$. We can use this as an approximate (upper bound) solution of the non-linear difference equation. The approximate solution does give some insight into the algorithm's scaling properties. Figure 10-1 shows the second expression plotted alongside data from 35 simulated runs of the

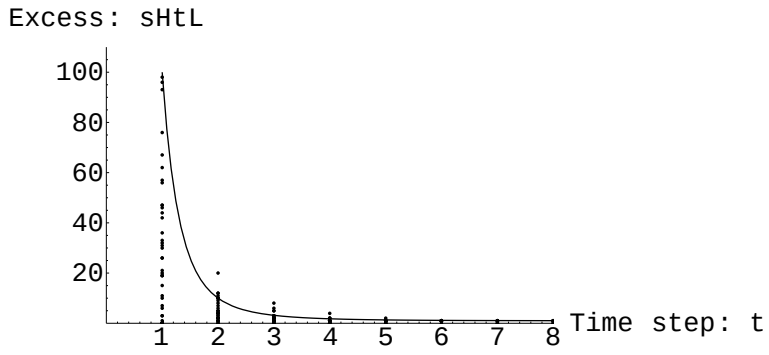

Figure 10-1: Measured and theoretical excess s as a function of t for $n = 10^4$ units. The scattered points represent 35 runs of the algorithm, and the solid line is the analytical model.

algorithm with $n = 10^4$: the s_t values are plotted against t, and the data from all these runs is overlayed. The simulations were performed using Mathematica's default (real) pseudo-random number generator. In each iteration of the algorithm, the units with value 1 flip to 0 with probability p_t, where $p_t = 1 - \frac{n}{2\xi_{t-1}}$, as specified earlier. (Our simulation assumes that the units have access to a real valued random number generator. A more detailed simulation could model the process units would employ if they only had access to binary random number generators.)

Both the non-linear difference equation for s and the upper bound approximation for s_t converge to 1 as t grows, though in the actual algorithm this value eventually converges to zero. The algorithm's dynamics enters a completely new regime when $s = 1$; presumably there is a continuous rather than discrete change to this new regime, and our analysis would ideally model that. We will assume there are two distinct epochs in the algorithm dynamics, one in which $s > 1$ (the period analyzed above), and $s = 1$, the endgame.

We will now find the convergence time of the first phase of the algorithm. We will consider the time required for s to converge to 2, since our analytical expressions can never actually reach 1. Solving $s_t = n^{\frac{1}{2^t}} = 2$ for t yields $t = \log_2 \log_2 n$.

Thus L_1, the time required for the first phase of the algorithm, is given by

$$L_1 = \log_2 \log_2 n$$

Endgame

Once the excess has been reduced to one, the algorithm's dynamics enters a new regime, since s_t cannot become any smaller than one without the algorithm terminating. Thus once the algorithm has reached this "endgame," equation 10.1 is no longer a good model (and the upper bound formula is even worse), so we must analyze it differently. The probability of exactly one of the $\xi = \frac{n}{2} + 1$ high units flipping in one time step (i.e. $\frac{n}{2} + 1$ trials) is given by a Poisson distribution: $w = \frac{\lambda^1}{1!} e^{-\lambda}$, where $\lambda = (\frac{n}{2} + 1)p$, with $p = \frac{s}{n/2+s} = \frac{1}{n/2+1} = \frac{2}{n+2}$. Writing out the expression for λ, one finds that it simplifies to one. Thus the probability of exactly one unit flipping in a single parallel trial is $\frac{1}{e}$.

In each iteration of the algorithm, either one unit flips, or some other number flips, so each iterate is a Bernouli trial with $p = \frac{1}{e}$. (If more than one unit flips, the units all revert to their previous state; thus there really are only two cases to worry about.) The probability

of a certain number of trials L_2 occurring before success is given by a geometric distribution $p(L_2) = p(1-p)^{L_2-1}$. The probability of winning at any time l up to and including L_2 is therefore given by

$$p_s = \sum_{l=1}^{L_2} \frac{1}{e}(1-\frac{1}{e})^{l-1} = 1 - (\frac{e-1}{e})^{L_2}$$

We have summed the geometric series to find the final expression. The probability of the algorithm failing to terminate prior to time L_2 is $p_f = 1 - p_s$. Solving for L_2, we find

$$L_2 = \frac{\ln p_f}{\ln \frac{e-1}{e}} = -2.18 \ln p_f$$

In order to make p_f less than 10^{-15}, we must chose $L_2 = 75$. The total running time L is given by

$$L = L_1 + L_2 = \log_2 \log_2 n + 2.18 \ln \frac{1}{p_f}$$

Under this algorithm, each unit needs $\log n$ bits of storage to hold its own ID, $\log n$ more to represent the random variable that it uses in deciding whether to take part in the corrective flip, and another $\log n$ to represent the ξ_t value that it reads from the AM bus. Thus the storage T needed for each unit is

$$T = \log n^3$$

Finally, the total number of random bits R used is the product of the number of trials L times $\log n^2$ (the number of storage bits per unit devoted to random variables; each of these gets randomized once per trial) times n, the number of units. $R = (\log_2 \log_2 n + 2.18 \ln \frac{1}{p_f})(\log n^2)n$ which can be rewritten

$$R = (n \log n^2)(\log_2 \log_2 n + 2.18 \ln \frac{1}{p_f})$$

This algorithm assigns a set of short IDs much more quickly than the sequential algorithm. However, its bandwidth requirement is correspondingly higher.

10.3 Conclusion

The tables below summarize and compare the three algorithms. A point not illustrated by the tables is the fact that the three algorithms use different amounts of communication: in the first, symmetry breaking phase of the non-interacting algorithm, the units don't do any distributed communication at all (the second phase of this algorithm, which does involve communication with the elected leader, may be viewed as optional). In the sequential algorithm, the units learn the OR of the other units' random bits. In the parallel algorithm, units communicate simultaneously using multiple frequencies, and the units sharing a frequency learn the sum of the other units' random bits. Clearly the third algorithm requires a channel with more bandwidth (for the multiple frequencies) and better signal-to-noise ratio (to make accurate measurements of the sums). These quantities, bandwidth and signal-to-noise, define the theoretical capacity of the channel, so it is clear that a higher capacity channel is required for the third algorithm. It would not be difficult to estimate the number

of bits actually exchanged in the course of the various protocols.

What is notable in examining the tables is that the algorithms that use more communication require less time. The parallel algorithm in particular requires far less running time than the others, and requires a much fatter communication pipe. One can imagine a framework analogous to the Shannon's noisy coding theorem for understanding the fundamental limits of symmetry breaking protocols: it would relate the number of units, the number of random bits required, the time to convergence, the channel capacity of the bus, and probability of failure.

The first step would be to calculate a lower bound on the number of bits that must be exchanged in order for each unit to be (reasonably) certain that no one else is sharing its ID. Next, one might attempt to budget the entropy flow for each of the algorithms: entropy enters the system at the random number generator, then, through the communication mechanism, each unit learns some amount of information about the random variables in the other units. Finally, there is a stage in which unnecessary entropy is rejected and IDs are assigned. It would be interesting to know how closely the algorithms presented here approach these hypothetical lower bounds.

Another interesting question is how efficiently each protocol uses its bus. For example, using the bandwidth and SNR requirements of the parallel algorithm, one could calculate a required (peak) channel capacity, in bits per second. By calculating the total number of bits exchanged and the total running time, one could find the average communication rate, again in bits per second. Does the algorithm use the bus efficiently, utilizing its full capacity most of the time? Or, despite the need for high peak capacity, is the bus idle for much of the algorithm?

We have proposed and analyzed three distributed protocols for using a source of physical symmetry breaking to assign unique identities to a group of n processors. The communication requirements, and associated hardware complexity, of the third, parallel protocol are probably too demanding for it to be practical, despite its speed. The second, sequential protocol is faster than the non-interacting protocol, uses less storage, and does not have burdensome hardware requirements. Thus in practice, the sequential protocol is likely to be most useful. There is probably an even more practical, hybrid protocol that combines the minimal hardware requirements of the sequential protocol with some of the speed advantages of the parallel protocol. Its running time would be intermediate between the two, would require somewhat more storage, and would be difficult to study analytically. Such a hybrid protocol is likely to be most useful in practice.

As the number of networked processors explodes in the next few years, distributed protocols that allow the processors to automatically and dynamically manage their own ID assignment may become increasingly appealing. Our investigation of the ID assignment problem suggests that there is interesting theoretical structure lurking beneath this potentially very practical problem.

T (storage/unit)	
non-interacting	$\log_2 \frac{n^2}{p_f}$
sequential	$\log n$
parallel	$\log n^3$

L (time)	
non-interacting	$(n+1)\log_2 \frac{n^2}{p_f}$
sequential	$\frac{n \ln n}{\ln 2} + \log_2 \frac{1}{p_f}$
parallel	$\log_2 \log_2 n + 2.18 \ln \frac{1}{p_f}$

R (random bits)	
non-interacting	$n \log_2 \frac{n^2}{p_f}$
sequential	$n^2 \log 2n$
parallel	$(n \log n^2)(\log_2 \log_2 n + 2.18 \ln \frac{1}{p_f})$

T (storage/unit)	$n = 10^6$	$n = 320$
non-interacting	90	67
sequential	20	9
parallel	60	25

L (time)	$n = 10^6$	$n = 320$
non-interacting	9.0×10^7	2.1×10^4
sequential	2.0×10^7	2.7×10^3
parallel	80	78

R (random bits)	$n = 10^6$	$n = 320$
non-interacting	9.0×10^7	2.1×10^4
sequential	1.4×10^{13}	6.6×10^5
parallel	3.2×10^9	4.2×10^5

Chapter 11

Conclusion

Electric Field Imaging is a practical and useful new mechanism for machines to perceive their human users, and thus for people to interact with technology.

In order to make the field measurements necessary for Electric Field Imaging, I developed inexpensive hardware and signal processing techniques for measuring, at millisecond update rates, femtofarad changes in the off-diagonal entries of the sensing capacitance matrix. I have also developed a method for solving the inverse problem of inferring geometrical information about the conductivity distribution from these capacitance measurements. Using my hardware and algorithms, it is now possible for a machine to accurately track the three dimensional position and orientation of one or two hands at high update rates.

The next important milestone from the practical user interface perspective is extracting some sort of "click" gesture from the measurements. Doing so should be a matter of development and implementation, rather than research. The framework presented here can be extended in a straightforward fashion to solve this problem, simply by searching a forward model with more degrees of freedom, such as a thumb.

11.1 Contributions

The research contributions presented in this thesis may be grouped into four categories: signal processing and algorithms, hardware, applications, and the inverse problem. The signal processing contribution includes synchronous undersampling, a narrowband, phase sensitive detection technique that is well matched to the capabilities of contemporary microcontrollers. In hardware, the primary contributions are the School of Fish, the scalable network of microcontroller-based transceive electrodes, and the LazyFish, the small footprint integrated sensing board. The inverse electrostatics portion of the thesis presents a fast, general method for extracting geometrical information about the configuration and motion of the human body from field measurements. The method is based on the Sphere Expansion, a novel fast method for generating approximate solutions to the Laplace equation. Finally, the thesis describes a variety of applications of electric field sensing, many enabled by the small footprint of the LazyFish. To demonstrate the School of Fish hardware and the Sphere Expansion inversion method, the thesis presented 3 dimensional position and orientation tracking of one or two hands. Additional signal processing contributions include an investigation of code division multiplexing of a sensor channel, and application of channel level coding and information theory to the problem of digital watermarking. My work on distributed protocols for ID assignment is an additional algorithm contribution

motivated by problems that arose in the course of developing the field sensing hardware.

11.2 Future Work

This thesis poses, rather than answers, questions about the user interface. It may take a long time to understand how best to exploit Electric Field Imaging for user interface purposes. The answers will probably emerge organically through applications such as those presented in chapter 7.

It also poses questions about machine perception. The School of Fish hints at what a "post-empiricist" sensing system would look like. Information flows bidirectionally in the School of Fish, rather than just from the sensor to the processor. It is still unclear how best to take advantage of the bidirectional information flow that occurs in the School of Fish. None of the algorithms presented in this thesis propagate information from the high level representation down to the low level sensing process, even though the hardware is capable of supporting this. Since each School of Fish unit contains a processor, one could even imagine omnidirectional information flows, in which the sensing and high level processing occur in a distributed fashion throughout the system. Such a system would presumably be much more similar in architecture to biological systems. Perhaps we will ultimately have to reinvent the whole fish, not just appropriate its sensing mechanism.

11.3 Coda

Although fish were "early adopters" of electric field imaging, the sensing mechanism has not been extensively exploited for technological purposes. The Theremin is the one notable exception to this rule. If Electric Field Imaging is truly successful and becomes a pervasive feature of our technological environment, perhaps the Theremin will someday seem like a fairly ordinary piece of twenty first century technology that anachronistically made its appearance in the early twentieth century.

Appendix A

LazyFish Technical Documentation

A.1 LEDs

The LazyFish has two sets indicator of LEDs, one set on the sensor subsection, and one set for the RS-232 interface. These can be very helpful in debugging Lazyfish applications. On both boards, the amber LED shows that power is present, the green indicates incoming data from the host, and the red LED is for outgoing data. The data LEDs are tied directly to the communication lines and flash on when a one bit is present on the line. When the board is being polled by a computer, the red LED will not typically flash as brightly as the green, because the poll command is a single byte, while the Lazyfish's response is many bytes. Thus the resulting duty cycle for the red LED is much lower.

If either portion of the Lazyfish is being mounted in a case, the surface mount indicator LEDs may be removed, and panel or case-mounted LEDs used instead. Underneath each LED is a pair of holes for soldering wires leading to off-board panel-mounted indicator LEDs.

A.2 Connectors

A.2.1 Interface board

The RS-232 portion of the Lazyfish has a DB-9 serial connector, power jack, and stereo audio jack. The stereo audio jack carries two TTL level data lines (for data headed to the host and from the host), plus ground. When the sensing portion of the board is split from the RS-232 interface, a stereo cable can be used to connect the two portions. There are six holes (with standard .1" inter-hole spacing) on the RS-232 portion and 6 corresponding holes on the sensing portion that can be stuffed with a 6 pin header. When the two boards are mounted in the "sandwich" configuration, the header can be used to join the two boards. When the sensing board is in the "remote" configuration, rather than using the stereo cable to connect the two boards, a ribbon cable connecting these headers could be used. The advantage over the stereo jack is that the interface board can also supply power to the sensing board this way, for cases in which there is no local power supply for the sensing board. The power switch on the interface board cuts the power to both boards when the sensor board is getting its power from the interface unit. If the sensor board is being powered from its own battery, you will probably want to add an additional, external power switch for the sensor board.

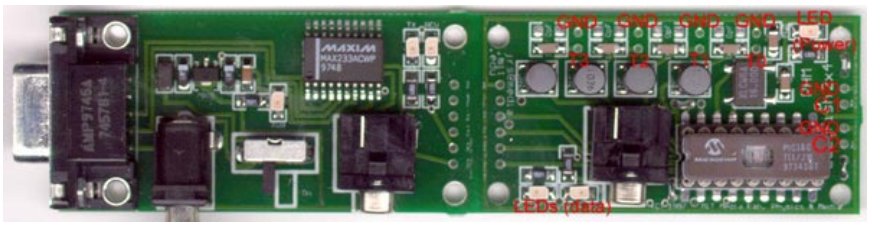

Figure A-1: This figure shows the board connectors.

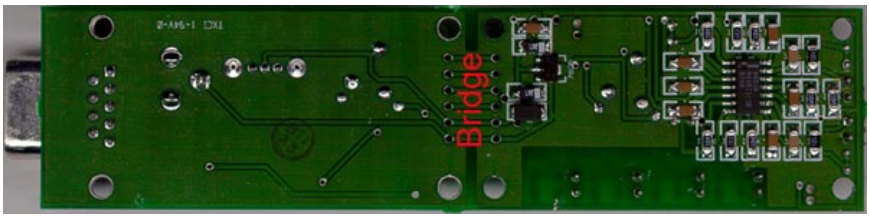

Figure A-2: The bottom of the Lazyfish board (integrated configuration), showing the bridge connecting the two halves.

A.2.2 Bridge

The bridge is the set of six lines connecting the interface board to the sensing board. There is a hole on each side of each bridge line, for soldering wires or headers. When the bridge is broken to separate the boards, these holes may be used to connect the two halves. The six bridge lines are: power (unregulated), ground, data TX, data RCV, DIG1, and DIG2. The power line is the raw, unregulated power supplied to the RS-232 board. The sensing board has its own voltage regulation and power conditioning so that it can be run from a local battery if desired. The labels TX and RCV are from the computer's perspective...if you consider the RS-232 interface to be a part of the computer, you could argue that this is a feature. Even if you disagree, that's the way it is. Two unused bi-directional PIC digital lines are broken out for easy access on the DIG1 and DIG2 lines. These lines do not connect to anything on the RS-232 board, but the holes on the RS-232 side of the bridge were included for mechanical purposes: a six pin header can be used to connect the boards in the sandwich configuration.

A.2.3 Sensing board

Two of the PIC pins connected to ADC channels are unused, and these pins have also been broken out to their own holes. These two pins can also be configured as configured as bi-directional digital lines if desired. See figure [] for the location of these holes.

Four wire loops, intended for attaching scope probes to monitor the behavior of the analog front end, are also included. There are two front end channels, each with two gain stages, and a wire loop for each gain stage. The plated-through screw holes in the corners of the board are all grounded and may be used as attachment points for the oscilloscope probe ground. Ordinarily, one would check the output of the second gain stage to determine whether the front end is clipping. Checking the output of the first gain stage would be rarer, probably only for investigating possible hardware problems.

There are a variety of holes for attaching transmit and receive electrodes. Each electrode hole has an associated ground hole spaced .1" away for connecting a shield. If the Lazyfish

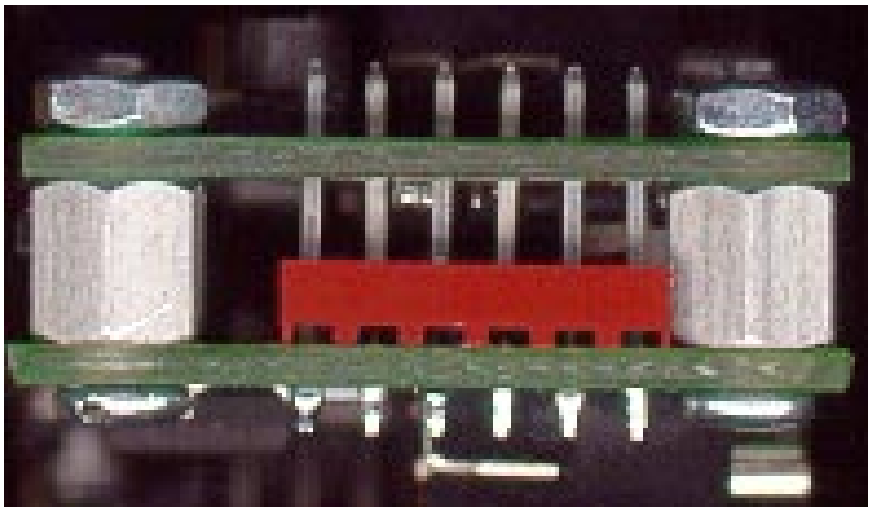

Figure A-3: The two halves of the Lazyfish connected by a header in the stacked configuration.

is being deployed in such a way that the electrode cables may move, or need to be changed occaisionally, I recommend soldering in a two-pin molex header in each, as shown in figure []. The transmit and receive electrode locations are labeled in figure []. There is enough clearance on the board to use the locking Molex connectors for the receive electrodes. The locking connectors can be used for the transmit electrodes too, though the connectors will sit on top of some surface mount components.

A.3 PIC pin assignments

Pin 1 PIN A2 R2—front end RCV chan 2
Pin 2 PIN A3 R3—uncommitted ADC channel 3
Pin 3 VREF
Pin 4 MCLR
Pin 5 VSS (GND)
Pin 6 PIN B0 DATARCV—serial data RCV
Pin 7 PIN B1 DATATX—serial data TX
Pin 8 PIN B2 DIG2—uncommitted digital pin
Pin 9 PIN B3 DIG1—uncommitted digital pin
Pin 10 PIN B4 TX3—resonant TX 3
Pin 11 PIN B5 TX2—resonant TX 2
Pin 12 PIN B6 TX1—resonant TX 1
Pin 13 PIN B7 TX0—resonant TX 0
Pin 14 VDD (+5V)
Pin 15 OSC2
Pin 16 OSC1
Pin 17 PIN A0 R0—uncommitted rcv chan 0
Pin 18 PIN A1 R1—front end RCV chan 1

A.4 Communications protocol and commands (code version LZ401)

The default communication protocol shipped with the LazyFish is 38400 baud, 8 bits, no parity, 1 stop bit (38.4K baud, 8N1). To poll the LazyFish, send either an ASCII W or ASCII R. After an R, the LazyFish will respond with a 48-byte string representing 8 decimal numbers between 0 and 65535. These 8 numbers are all the measurements the LazyFish can make: there are 4 transmitters and 2 receivers, and the eight numbers are all the transmit-receive pairs. The W command returns 4 numbers (24 bytes), representing the signal on the currently selected receive channel due to each of the four transmitters.
Protocol: 38.4K baud, 8N1

Your Command	Meaning	LazyFish Response (e.g.)
R	Read all 8 chans	00000 11111 22222 33333[CR] 44444 55555 66666 77777[CR]
W	Read 4 chans	00000 11111 22222 33333[CR]
S	Read Single Chan	65535[CR]
T	Test command...get raw samples	00255 00255 00255 00255[CR]
C0	Read from unused ADC chan 0	None
C1	Change to RCV chan 1	None
C2	Change to RCV chan 2	None
C3	Read from unused ADC chan 3	None
I*yz*	Change integration parameters	None
X0	Change to TX chan 0	None
X1	Change to TX chan 1	None
X2	Change to TX chan 2	None
X3	Change to TX chan 3	None
U	Read 4 chans, return binary result on DIG2	*aabbccdd* (4 two-byte values)
Y	Read 4 chans, return binary result	*aabbccdd* (4 two-byte values)
V	Get code version	LZ401

A.4.1 R command

Read all eight channels (all measurements made by pairing the 4 transmitters with 2 receivers). This command is not affected by the C or X commands: it automatically manages assignment of the transmit and receive channels. The values are 5 byte decimal ASCII representations of 16 bit numbers, so the range of values is 0 to 65535. Each value is followed by a space, except for the 4th and 8th value, which are followed by carriage returns.

A.4.2 W command, C command

Read 4 channels (each of the 4 transmitters paired with the currently selected receive channel). This command is not affected by the X command since it automatically choses the transmitters, but is affected by the C command, which choses the receive channel. The LazyFish has two analog front end receive channels, which can be selected using C1 or C2. The commands C0 and C3 select two uncommitted ADC channels (with no op-amp front

ends). To use the LazyFish as a data acquisition board (to measure a DC voltage to and return it to the computer), you'd use C0 or C3, followed by the T command.

A.4.3 S command

Read a single channel. Uses the currently selected transmitter or transmit mask, and the currently selected receive channel. The transmit channel (or channels) is selected by the X command (whichever was issued last). The receive channel is determined by the C command.

A.4.4 T commnd

The main purpose of the T command is to check whether the front end is clipping. This command returns 4 raw ADC samples (0, $\frac{\pi}{2}$, π, $\frac{3\pi}{2}$). The range of values is 0 to 255. For consistency with other commands, these are 5 byte decimal ASCII numbers (so the first two digits of each number are 0). If any of the 4 values returned by this command are 0 or 255, the front end is clipping. To eliminate clipping, the transmit burst length parameter z can be decreased, or the electrode geometry can be changed. To eliminate clipping by changing the electrode geometry, the electrode sizes can be decreased, the distance between the electrodes can be increased, or a grounded shield can be placed in the vicinity of the electrodes.

A.4.5 I command

The parameters y and z are each single bytes. The default values of y and z are 20 and 7. The parameter y specifies the number of measurements to be integrated each time the LazyFish receives a poll command. The z value is the length (in periods) of the transmit burst used to excite the resonant transmitter. If the receive front end is clipping (which can be determined by connecting an oscilloscope to test points on the board, or using the T command), this value should be decreased. The shorter excitation burst will prevent the transmitter from ringing all the way up to its maximum 80V peak-to-peak swing, and thus should prevent the receive front end from clipping.

The total measurement time is proportional to the product $y * z$. Increasing y will decrease the measurement noise, as will increasing z (up to the point that the front end clips). But the increased measurement time means that the sensor update rate is lower. The default parameter values represent a good tradeoff between update rate and measurement noise.

A.4.6 X command

This selects a transmitter. X0 picks transmitter 0, X1 picks transmitter 1, and so on. Since the R and W commands reset the transmit channel automatically, X does not affect R or W. Modifying the selected transmit channel does affect S and T, however.

A.4.7 Y command

Same as the W command, but the data is returned in binary rather than ASCII form. Since it returns a 16 bit value for each of the 4 channels, it returns 8 bytes (high byte, low byte, high byte, low byte, high byte, low byte, high byte, low byte).

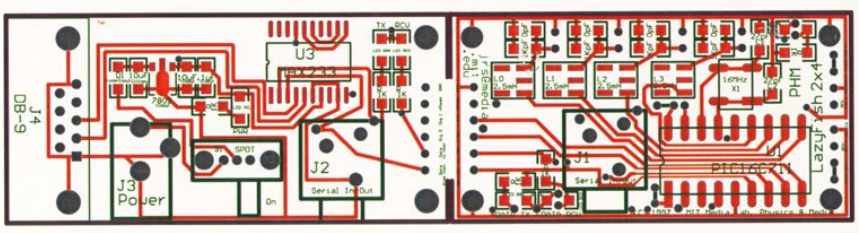

Figure A-4: Top of LazyFish.

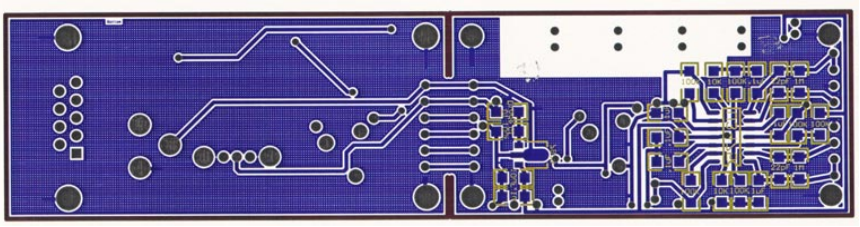

Figure A-5: Bottom of LazyFish

A.4.8 U command

Same as the Y command (4 channels returned in binary format), but the data is returned on the DIG2 line rather than the usual line used for serial communication. This function can be useful when the LazyFish is attached to both a PC and another microcontroller. The binary format is typically easier for another microcontroller to interpret (since it doesn't have to bother converting from ASCII). With the DIG2 line, the PC doesn't have to "hear" the data that the LazyFish is sending back to the other device.

A.4.9 V command

Returns code version. Current code version is LZ401.

A.5 LazyFish Front End Application

The LazyFish front end application can be used exercise most of the LazyFish commands, demonstrate the basic capabilities of the device, and debug electrode geometries. The main interface elements are the sliders, which are used to display sensor values. Each slider is labelled with the combination of transmitter and receiver it represents. For example, the first slider is labelled "T0R1," for transmit 0 and receive 1.[1]

The controls are grouped into various logically related frames. Interface objects that correspond closely to a LazyFish command are labelled with the command name, so that one can use the front end program to learn the commands. The frame labelled "Sensing Mode," for example, contains a set of radio buttons that determine which one of the various LazyFish Sensing commands should be used to update the display. In "Off" mode, the

[1] The LazyFish transmit channels start at 0, while the receive channels start at 1. This numbering system reflects the PIC's hardware (receive channel 1 corresponds to analog input 1). Having forgotten entirely about the hardware and confused myself many times while writing software, I consider this inconsistency of numbering schemes to be a dreadful mistake.

LazyFish is not being operated at all, and the display is static. In "1 Channel" mode, the S command is used to poll a single channel. Which channel is polled is determined by the parameters in the frame labelled "One Channel:" the contents of the "TX Chan" text box determine the transmit channel, and the "RCV Chan" box choose the receive channel. The sensor value is plotted on the appropriate slider. The "4 Channels" mode uses the W command to read out the bank of 4 channels associated with the currently selected receiver (as determined by the "RCV Chan" text box). The "8 Channels" mode makes no reference to the "TX Chan" or "RCV Chan" text boxes, because it reads out all the channels.

The "1 chan test" command is somewhat different than the others. It uses the "T" command, which returns raw in-phase and quadrature samples, with no integration. In the usual sensing modes, the LazyFish takes the magnitude in software. Also in these modes, multiple samples are taken and summed before returning the value. (The number of samples taken is specified by the "Int Time" text box in the "LazyFish Integration Parameters" frame.) In the T mode, the magnitude is not taken in the firmware, and the measurement is not repeated. The 4 raw values (0, 90, 180, and 270 degrees) are displayed as text in the Diagnostic frame, and also plotted as an XY plot. One purpose of the XY plot (and of the T mode) is to allow the user to see whether the sensor values are clipping, without using a scope. Basically it lets the LazyFish function as a scope. If any of the values are clipped (which occurs when a displayed vector touches the outer borders of the box), then the electrode geometry may be changed, or the transmit amplitude may be reduced by shortening the transmit square wave burst, as explained in Chapter 2.

The "Tx time" text box in the "LazyFish Integration Paramters" frame determines the length of the transmit burst. Decreasing this value should decrease the magnitude of the vectors plotted in the Diagnostic scope display, which can be used to eliminate clipping. The other text box in the Integration frame, "Int Time," has no effect on the T command, only on the usual commands that operate the sliders because these other commands integrate the results of multiple measurements before returning a sensed value.

On startup, and whenever the integration parameters change, one should measure baseline values for all the channels by clicking the "Get Baseline" button several times. Nothing should be in the sensing field when the baseline is taken. The Get Baseline button measures all the channels once using the current integration parameters, and subtracts future sensed values from this baseline. The reason this button should be clicked several times is that the VB front end performs some low-pass filtering of the sensed values (including those used to take the baseline), so a few samples are needed for the baseline to converge correctly. Once a baseline has been established, the sensor value should be zero for each channel when nothing is in the vicinity of the sensors.

The filtering mentioned above is controlled by the Alpha text box in the "Host Display Parameters" frame. When Alpha is 1, the sensed values are not low-pass filtered at all. When Alpha is 0, new data does not affect the displayed values at all. The filter is of the form $y = \alpha x + (1 - \alpha)y'$, where x is the latest data, y is the new displayed value, and y' is the previous displayed value. The default value of Alpha is .5.

The "Host Display Parameters" frame also contains the "Slider Min" and "Slider Max" text boxes, which allow the lowest and highest value on the slider to be set arbitrarily (with certain restrictions, like the Max has to be greater than the Min). Typically, Slider Min is left at its default value of 0, since if a baseline has been taken, the minimum sensor value is 0. It is usally necessary to adjust Slider Max when the integration parameters are changed, because increasing the integration time increases the size of the maximum possible deviation from the baseline (since each measurement is the sum of more numbers).

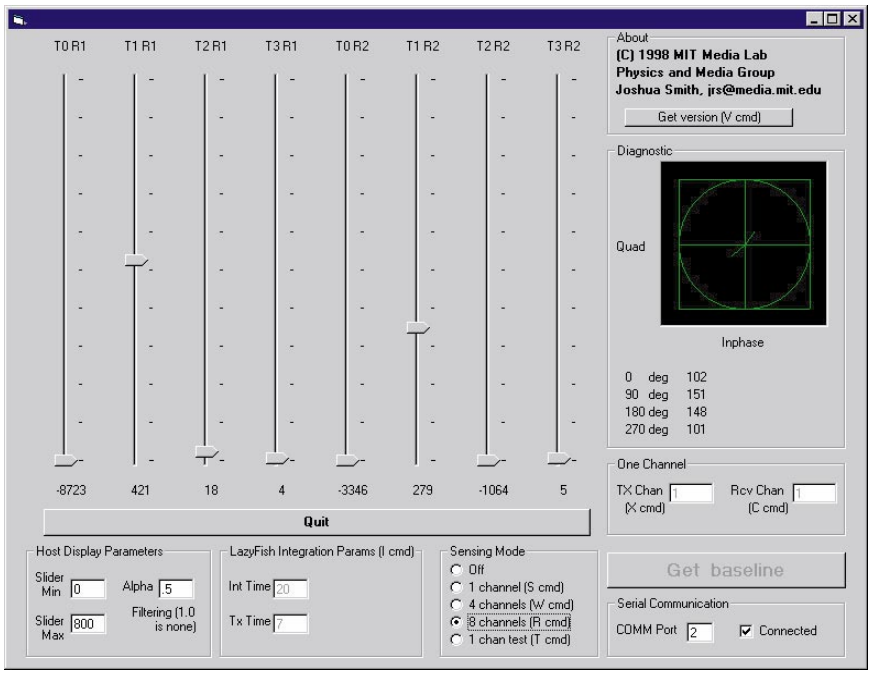

Figure A-6: The LazyFish front end application, oct.exe.

A.6 LazyFish Firmware (version LZ401)

```
// Joshua R. Smith
// MIT Media Lab (c) 1998
//
// LZ401       Some opts to shrink code
// LZ39C711    Fixed TRIS on ADC indicator pin
// LZ38C711    Try waiting for TX to die down. Got rid of outer int loop.
// LZ37C711    Use sense to implement test;
//             do abs(sum(i))+abs(sum(q)) instead of
//                sum(abs(i))+sum(abs(q)) [faster, OK if is phase const]
// LZ36C711    Added binary output mode
// LZ35C711    Fixed bad sensing problem.  Made sense() consistent
//             with test()
// LZ34C711    Added U (url) and V (version) cmds.  Changed naming convention.
// lazy33      Added single channel read cmd;
//             Improved timing in main sense command as lazy32 did for test cmd.
// lazy32      In test mode, added spikes on dig1 to indicate when sampling occurs
//             Improved timing in test cmd by using DELAY_CYCLES instead of DELAY_US
// lazy31      switch to C711
// lazy30      added code to switch TX, and code to give user direct control over TXMASK
// lazy29      added test code to detect saturation / measure phase
// lazy28      added cmd to read all 8 chans; moved sense code from include into this file
// lazy27      Use all 4 TXs, by compressing code, by not using OUTPUT_HI on tx
// lazy26      Fix multiple rcv channels
// lazy25      Delayed powerup message so it doesn't get stomped on by FishFace
// lazy24      Change default ADC chan; reverse order of txs
// lazy23      Fixed bug in 20: clear registers before sensing
// lazy20      Don't use U as cmd
// lazy19      Made tx burst length a variable
//             Switched to 711
//             Made sense routine an include; took out software filtering (not enough ram)
// lazy18      Switch to using longs; do software filtering
// lazy17      Add inphase & quad together
// lazy16      Made mode w/ no spike on TX2;  added delay
//             between bursts 2 and 3 to keep phase consistent;
//             added hex output; changes sign of hex quad output to\
//             make it positive
//     100kHz  This is debug version: makes spike on TX2,
//                   outputs raw measured values
// lazy15    four channels
// lazy14    quadrature
// lazy13    Separate inner and outer integration loop
// lazy12
// lazy3.c   Data out serial port
// lazy2.c
// synch3.c  Made integration time adjustable param
// synch2.c  Split fish read into separate routine
// based on
```

```
// synch1.c  First working Laz-i-fish code

#byte portb = 6

#include <16C711.h>
//#include <stdio.h>
#include <math.c>

#fuses HS,NOWDT,NOPROTECT,BROWNOUT,PUT
#USE Delay(Clock=16000000)
//#USE FAST_IO(a)
//#USE FAST_IO(b)

// pin 17
#define R0  PIN_A0
// pin 18
#define R1  PIN_A1
// pin 1
#define R2  PIN_A2
// pin 2
#define R3  PIN_A3

// Pin 6
#define RX  PIN_B0
// Pin 7
#define TX  PIN_B1
//Pin 8  binary data to eg FishFace
#define DIG2 PIN_B2
//Pin 9  ADC indicator pin
#define DIG1 PIN_B3

// pin 10
#define TX3 PIN_B4
// pin 11
#define TX2 PIN_B5
// pin12
#define TX1 PIN_B6
// pin 13
#define TX0 PIN_B7

#define TX3MASK   0x10
#define TX2MASK   0x20
#define TX1MASK   0x40
#define TX0MASK   0x80

#USE RS232 (Baud=38400, Parity=N, Xmit=TX,Rcv=RX)

#define hi(x)  (*(&x+1))
#define lo(x)  (*(&x))

byte G_inttime,G_txtime;
byte G_it_old;
long i_adc;
long q_adc;
byte cmd;
byte txmask;
byte chanoff;
//byte binmode;

long i2_adc;
long q2_adc;
long i1_adc;
long q1_adc;

long chan[8];

void initialize() {
   OUTPUT_HIGH(TX);

   SET_TRIS_A(0xFF);// WAS 0X0F
   //SET_TRIS_B(0xF6); // TXs, RX, TX and DIGIN are tristated
   //SET_TRIS_B(0xFF); // everything tristated
   SET_TRIS_B(0xFB); // everything tristated but DIG2
   SETUP_PORT_A(ALL_ANALOG);
   SETUP_ADC(ADC_CLOCK_DIV_32);
   SET_ADC_CHANNEL(1);

   i_adc = 0;
   q_adc = 0;
   chanoff = 0;
   G_inttime = 20;
   G_txtime = 7;
   OUTPUT_HIGH(DIG2);
//   binmode = 0;
}



void putdecl(long n) {
putchar( (n/10000) +'0');
putchar( ((n/1000) % 10) +'0');
putchar( ((n/100) % 10) +'0');
```

```
putchar( ((n/10) % 10) +'0');
putchar( (n % 10) + '0');
}


outd4() {
byte c;
   for (c=chanoff; c<chanoff+3; c++) {
     putdec1(chan[c]);
     putc(' ');
   }
   putdec1(chan[c]); // no space
   putc('\r');
}

outb4d2() {
byte c;
#USE RS232 (Baud=38400, Parity=N, Xmit=DIG2,Rcv=RX)
   for (c=chanoff; c<chanoff+4; c++) {
     putc(hi(chan[c]));
     putc(lo(chan[c]));
   }
#USE RS232 (Baud=38400, Parity=N, Xmit=TX,Rcv=RX)
}

outb4() {
byte c;
   for (c=chanoff; c<chanoff+4; c++) {
     putc(hi(chan[c]));
     putc(lo(chan[c]));
   }
}

sense() {
byte i, j, k;
byte adcval;
    i_adc = 0;    // Erase old value of adc...cut if we're doing low pass filter
    q_adc = 0;
      for (j=0; j < G_inttime; j++) {
       SET_TRIS_B(0x02);     // UNTRIS TXs, UNTRIS DIG2, TRIS TX
       for (k=0; k < G_txtime; k++) {
         portb = txmask;
         DELAY_CYCLES(18);
         portb = 6;               // 4+2
         DELAY_CYCLES(10);
       }
       //DELAY_CYCLES(10);
       DELAY_CYCLES(6);          // reduced by 4 to allow for ADC indicator
       portb = 14;               // ADC indicator 8+4+2
       portb = 6;                // 4+2
    adcval=READ_ADC();                      // phase pi/2
       SET_TRIS_B(0xF3);         // RE-TRIS TXs to discharge tank
       q1_adc = adcval;
       q_adc = q_adc+q1_adc;
       DELAY_US(15);              // time for tank to discharge


       SET_TRIS_B(0x02);         // UNTRIS TXs, UNTRIS DIG2, TRIS TX
       for (k=0; k < G_txtime; k++) {
        portb = txmask;
        DELAY_CYCLES(18);
        portb = 6;               // 4+2
        DELAY_CYCLES(10);
       }
       //DELAY_CYCLES(20);
       DELAY_CYCLES(16);         // reduced by 6 to allow for ADC indicator
       portb = 14;               // ADC indicator 8+4+2
       portb = 6;                // 4+2
    adcval=READ_ADC();        // phase pi
       SET_TRIS_B(0xF3);         // RE-TRIS TXs to discharge tank
       i2_adc = adcval;
       i_adc = i_adc-i2_adc;
       DELAY_US(15);              // time for tank to discharge

       SET_TRIS_B(0x02);         // UNTRIS TXs, UNTRIS DIG2, TRIS TX
       for (k=0; k < G_txtime; k++) {
        portb = txmask;
        DELAY_CYCLES(18);
        portb = 6;               // 4+2
        DELAY_CYCLES(10);
       }
       //DELAY_CYCLES(30);
       DELAY_CYCLES(26);         // reduced by 4 to allow for ADC indicator
       portb = 14;               // ADC indicator 8+4+2
       portb = 6;                // 4+2
    adcval=READ_ADC();        // phase 3pi/2
       SET_TRIS_B(0xF3);         // RE-TRIS TXs to discharge tank
       q2_adc = adcval;
       q_adc = q_adc-q2_adc;
       DELAY_US(15);              // time for tank to discharge

       SET_TRIS_B(0x02);         // UNTRIS TXs, UNTRIS DIG2, TRIS TX
```

```
       for (k=0; k < G_txtime; k++) {
        portb = txmask;
        DELAY_CYCLES(18);
        portb = 6;                 // 4+2
        DELAY_CYCLES(10);
       }
       //DELAY_CYCLES(40);
       DELAY_CYCLES(36);          // reduced by 4 to allow for ADC indicator
       portb = 14;                // ADC indicator 8+4+2
       portb = 6;                 // 4+2
    adcval=READ_ADC();         // phase 2pi
       SET_TRIS_B(0xF3);          // RE-TRIS TXs to discharge tank
       i1_adc = adcval;
       i_adc = i_adc+i1_adc;
       //DELAY_US(5);
      }
    //SET_TRIS_B(0xF6);   // RE-TRIS TXs
    SET_TRIS_B(0xFB);   // RE-TRIS TXs, but don't TRIS DIG2

    if (i1_adc < i2_adc) {// need to invert to take abs?
      i_adc = 65535-i_adc;
    }
    if (q1_adc < q2_adc) {// need to invert to take abs?
      q_adc = 65535-q_adc;
    }

    i_adc = i_adc+q_adc;
}



read4_ef() {
byte i;
byte c;
byte txmaskind;
   txmaskind = TX0MASK;
   txmask = txmaskind+6; //4+2

   for (c=chanoff; c<chanoff+4; c++) {
     sense();
     chan[c] = i_adc;
     txmaskind = (txmaskind >> 1);
     txmask = txmaskind+6; //4+2
   }
}



readoutd4() {
   read4_ef();
   outd4();
}

rc1() {
      SET_ADC_CHANNEL(1);
      chanoff = 0;
}

rc2(){
      SET_ADC_CHANNEL(2);
      chanoff = 4;
}


void main () {

  initialize();
  while(1) {
    cmd = getc();
    if (cmd == 'U') {      // Read current bank of 4 in binary on DIG2
      read4_ef();
      outb4d2();
    }
    if (cmd == 'Y') {      // Read current bank of 4 in binary
      read4_ef();
      outb4();
    }
    if (cmd == 'W') {      // Read current bank of 4
      readoutd4();
      //printf("\r");
    }
    if (cmd == 'R') {      // Read all 8
      rc1();
      readoutd4();
      //putc(' ');
      rc2();
      readoutd4();
      //putc('\r');
      chanoff = 0;
    }
    if (cmd == 'S') {
      sense();
```

```
      putdec1(i_adc);
      putc('\r');
    }
    if (cmd == 'C') {
      cmd = getc();
      if (cmd == '0') {
         SET_ADC_CHANNEL(0);
      }
      if (cmd == '1') {
         rc1();
      }
      if (cmd == '2') {
         rc2();
      }
      //if (cmd == '3') {
      //   SET_ADC_CHANNEL(3);
      //}
    }
    if (cmd == 'X') {
      cmd = getc();
      if (cmd == '0') {
         txmask = TX0MASK+6; //4+2
      }
      if (cmd == '1') {
         txmask = TX1MASK+6; //4+2
      }
      if (cmd == '2') {
         txmask = TX2MASK+6; //4+2
      }
      if (cmd == '3') {
         txmask = TX3MASK+6; //4+2
      }
    }
    //if (cmd == 'K') {
    //  txmask = getc();
    //}
    if (cmd == 'T') {      // Test to check for ADC saturation
      G_it_old = G_inttime;
      G_inttime = 1;
      sense();
      chan[chanoff+0] = q1_adc;
      chan[chanoff+1] = i2_adc;
      chan[chanoff+2] = q2_adc;
      chan[chanoff+3] = i1_adc;
      outd4();
      //putc('\r');
      G_inttime = G_it_old;
    }
    if (cmd == 'I') {
   G_inttime = getc();
   G_txtime = getc();
    }
    if (cmd == 'V') {
      printf("LZ401\r");
    }
    //if (cmd == 'B') {
    //  binmode = 1;
    //}
    //if (cmd == 'A') {
    //  binmode = 0;
    //}
    //if (cmd == 'U') {
    //  printf("http://jrs.www.media.mit.edu/~jrs/lazydrv\r");
    //}
   SET_TRIS_B(0xFB);
  }
}
```

Appendix B

School of Fish Technical Documentation

B.1 Description of Schematic

B.1.1 Power Supply

School of Fish units receive +12V and ground from the bus. The 12V signal passes through a protection diode to an LM7805CT regulator (U7) which puts out +5V. Large bypass capacitors (C10 and C11, 22uF) are located before and after the regulator to compensate for any slow fluctuations in the power supply (shunt low frequency fluctuations to ground). A smaller bypass cap (C12, .1uF) is also situated at the output of the regulator. Its purpose is to shunt high frequency fluctuations to ground. (Ideally the large capacitors would attenuate the fast fluctuations even more effectively than slow fluctuations, but because of resonances associated with lead inductance, certain bands will not be attenuated. Thus a range of capacitances are needed to attenuate all frequencies effectively.)

With the exception of the hardware noise generator, which takes +12V, all School of Fish circuitry runs on +5V supply. The MAX475, which is a single supply quad op amp, was chosen because it does not require an additional negative supply. However, since the boards must process both positive and negative signals, we generate a 2.5V reference for the analog electronics, which functions as analog ground, with 0V functioning as a -2.5V negative supply, and +5V functioning as the +2.5V positive supply.

To generate the +2.5V reference, we use two 100K resistors (R6 and R7) to divide the +5V supply. This +2.5V reference is buffered by an op-amp follower (U3B). The low-impedance output of this follower is our +2.5V reference signal. In the schematic, we use the signal ground symbol (open triangle) to denote this +2.5V reference voltage. We placed a .1uF bypass capacitor (C15) between the output of the follower and ground to filter fluctuations in the reference.

B.1.2 Front End Transceiver

The obvious way to make a transceiver would be to connect the electrode through a SPDT CMOS switch (2-1 multiplexer) which could connect the electrode either to transmit or to receive circuitry. The problem with this straightforward approach is that the switch adds an unacceptable amount of noise to the very sensitive, unamplified signal. (I know because this was the first thing I tried.) Instead, I used a circuit which allows the same opamp to

function as either a transmitter or receiver, depending on whether a PIC pin (RA2) is put in a high impedance mode (tristated) or driven as an output.

When the unit is in receive mode, RA2, which is connected to the non-inverting input of the front end opamp, is tristated, and thus has very little effect on the op-amp. In this condition, the non-inverting input is pulled (somewhat weakly) to 2.5V (analog ground) by the 10K "pulldown" resistor R7. The reason for the high value of this pull down resistor (10k would be more usual) is that when transmitting, the PIC drives RA2 with 0V and +5V. When this happens, R7 acts as a shunt, weakening the transmitted signal unacceptably. The disadvantages of using such a high value for the pulldown are that it takes somewhat longer (uS) for the front end to return to the 2.5V state after transmission ends, and the non-inverting input would be more prone to independent voltage fluctuations that would show up as noise in the output. (On the other hand, it would tend to reject common mode fluctuations somewhat...)

In receive mode, small currents from the electrode flow to the inverting opamp's inverting input, and the opamp generates a voltage on its output such that the current through the feedback resistor cancels the current from the electrode. (Almost no current is sunk into the inverting input, since it is high impedance).

For the small signals that occur in receive mode, the crossed diodes in the feedback network are essentially an open circuit, so the effective feedback network in receive mode consists of the 1M resistor R9 and the 22pF capacitor C4. At our typical 75kHz operating frequency, this network presents an impedance of about $100k\Omega$. The purpose of the capacitor is to balance the phase shift that appears on the input due to the capacitive coupling.

The transceiver front end uses a series inductor-capacitor resonant "tank" (L1 = 2.0mH and C3 = 2400pF) tuned to 75kHz (the resonant frequency $f_0 = \frac{1}{2\pi\sqrt{LC}}$). The role of the tank is to increase the amplitude of the transmit voltage (by a factor of $Q = \sqrt{\frac{L}{C}}$), which increases the transmitted signal—and therefore the received signal—without increasing receiver noise (unlike increasing receiver gain, which increases both). The electrode is connected to the node between the inductor and capacitor. When the tank is driven at its resonant frequency, there is a large voltage swing on this node. In receive mode, the tank functions as a bandpass filter.

The output of the front end gain stage is capacitively coupled by C9 to the next gain stage. On the far side of the coupling capacitor is a 100k "pull-middle" (pull-to-analog-ground) resistor (R22) that ensures that the second gain stage doesn't see any dc offset with respect to analog ground. This second stage has a gain of 10. It is followed by a unity gain inverter that is used for the synchronous detection.

The second (non-inverted) and third (inverted) gain stages feed into two SPDT switches in the 4053. The first switch is for the in phase demodulation, and the second is for the quadrature demodulation. The in phase switch alternately connects the non-inverted or the inverted input signal to its output. It switches at the transmit frequency. The second (quadrature) switch does the same 2-1 multiplexing operation, and at the same frequency as the in phase channel, but is phase shifted by $\pi/2$ with respect to the in phase switch. The pair of switches thus have four possible states, and two lines from the PIC (one for each switch) are used to control them.

The in phase and quadrature demodulated outputs are then integrated in separate op-amp integrators. There is a 10k resistor (R14 / R15) between the output of the switch and the integrator's inverting input, and a $.01uF$ capacitor (C7 / C8) and $1M$ leakage resistor (R20 / R21) in the feedback network. Because it must be possible to reset a receiver very

quickly and make a new measurement (from another transmitter), there is also a (quad) 4066 switch with a very low on resistance (U6A and U6C) across the feedback network that can be used to discharge the integration capacitors very quickly.

B.1.3 Digital Communication

The school of fish units communicate over an RS-485 differential serial bus. We use a DS75176 RS-485 transceiver chip to drive the bus. Each unit pulls the A line of the bus high with a 10k resistor (R2), and pulls the B line of the bus low, also with a 10k resistor (R3). In receive mode, the transceiver puts the difference between the A and B lines (thresholded to 0V or 5V) on its RO line, which feeds into an input on the PIC. The cathode of a red LED D1 is also connected to the RO pin (through a 10k resistor R4); the LED's anode is connected to +5V. When no data is on the bus, the RO line is high, so both leads of the LED are at +5V, and it doesn't light. When a bit appears on the bus, RO drops, the LED's cathode is pulled low, and the LED lights. Thus the red LED gives a visual indication of bus activity. It is important that the LED loads only the output of the RS 485 transceiver, and not the bus itself, because the current drawn by the LEDs would start to interfere with bus operation as the number of units increased.

To transmit, the PIC raises its RB2 output, which raises the transmit enable pin (labeled DE on the schematic) of the 485 transceiver, allowing it to seize the bus. Raising RB2 also lights the green LED D2, indicating that the unit is transmitting data.

B.1.4 Silicon Serial Number

A Dallas Semiconductor Silicon Serial Number was included on the School of Fish for experimenting with automatic identification strategies. The Silicon Serial Number communicates with PIC pin RB1 over a one-wire serial bus that is pulled high by a 5k resistor (R23).

B.1.5 Noise Circuit

The avalanche breakdown of a reverse-biased NPN transistor is our noise source. The base of the transistor is held at analog ground, and the emitter is pulled to +12V. The noise that appears on the emitter is capacitively coupled through C23 (.01uF) to the non-inverting input of on opamp. A 1M feedback resistor between the inverting input and the output provides gain. Additional high frequency gain is provided by the R17 (5k) and C22 (.1uF) shunt to ground. Higher frequencies are coupled more strongly to ground, so the opamp must work harder to make the inverting input follow the non-inverting input at higher frequencies.

B.2 School of Fish Interface Unit

The school of fish interface unit has a MAX233 RS232 level shifter and a DS75176 RS-485 transceiver chip, as well as a PIC. The MAX233 is wired directly to the DS75176: the PIC does not have to interpret and regenerate the signals passing between the host computer and the school. The PIC does however monitor activity on the RS232 TX line from the computer. When it sees a start bit from the computer, it asserts the transmit enable (TDRE) line on the DS75176 and holds it for one byte time, at the current baud rate. Initially I tried having the computer drive the TDRE line itself using the DTE line of

the serial port, but because timing under Windows 95 is so variable, this was not feasible. Sometimes the computer would keep transmit enable asserted for so long that it blocked the response from the school. In any case, it was more desirable for the school of fish to be accessible by ordinary 3 wire RS232, instead of a specialized protocol. This way it can be operated and debugged from an ordinary terminal program. It is also potentially backward compatible with existing Fish software.

Since the PIC has no operating system, it was straightforward to have it manage the transmit enable line—there are no timing issues. I could have used something simpler than a PIC, such as a one shot, to manage the transmit enable, but with the PIC, the delay is software controllable. So when I switched the school from 9600 baud to 38400 baud, I only had to make a small change to the code running on the interface unit.

An additional advantage of having a PIC on the interface unit is that it can be used for low level management of the school. For example, it can handle multiplexing to reduce the compute and communications burden on the host PC. The orientation sensing FieldMouse demonstration described in section 6.4 relies on this strategy to improve update rate. The PC issues a single high level command that the PIC on the interface unit translates into detailed commands that it broadcasts to the rest of the school. The PC can devote itself entirely to the inverse problem and graphics until all the data has been collected. Since there is a latency of up to 5mS associated with each serial port event (read or write) under Windows, using the PIC to eliminate many serial events altogether and "batch process" the rest improved performance substantially.

B.3 School of Fish Communications Protocol

B.3.1 I Command

The command to set the two burst parameters is I<byte1><byte2> (these two bytes are one of the few present exceptions to the ASCII rule). This command applies globally: it does not specify a fish ID, and all the units change their parameters when it is issued.

B.3.2 O Command

The O command reads out the burst parameters from a single unit. For example typing "O9" at a terminal causes unit 9 to return its burst parameters, for example the default "050 013<cr>" (meaning 50 bursts of 13 periods). We often use the O command as a "ping" to test if a unit is online. These default values work, but most applications change them to a value better suited for the application. Thus one can usually tell whether the unit is in a properly configured state, rather than a newly reset state, by examining this value.

B.3.3 T Command

The T command causes one unit to become the transmitter, and all other units to become receivers. For example, "T2" causes unit 2 to become the transmitter, and any other unit that had been a transmitter to become a receiver. When unit 2 has become the transmitter, it responds with an acknowledgement character (an X). This way the application software can be sure that the operation has finished before issuing further commands.

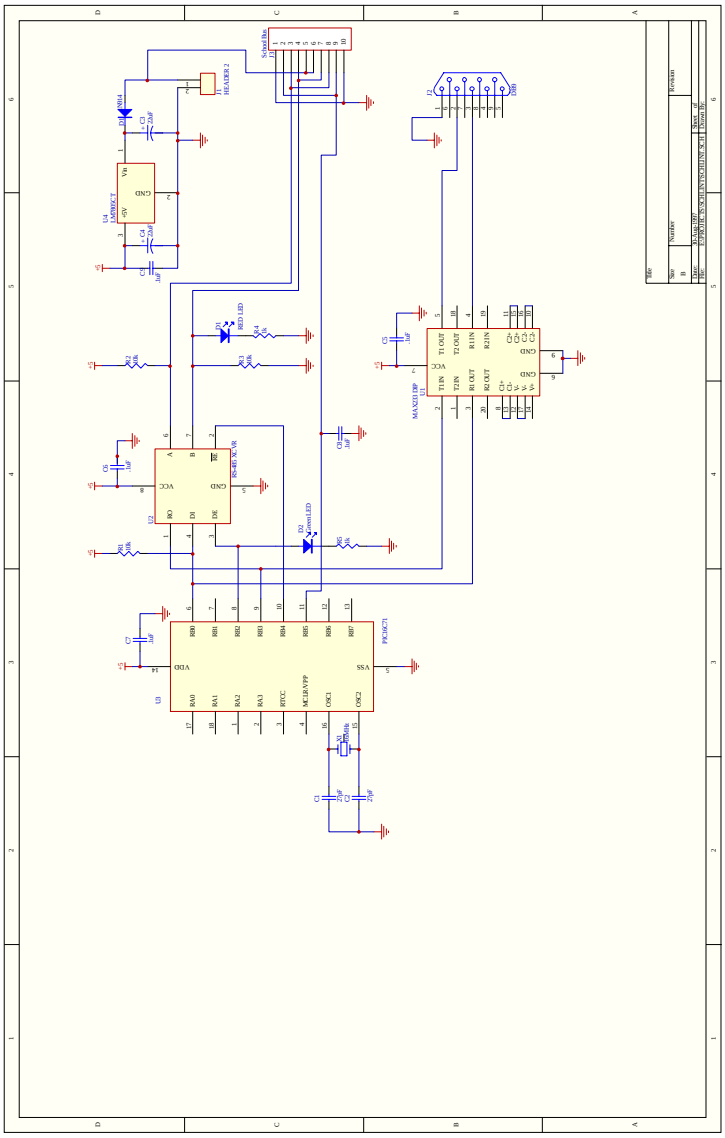

Figure B-1: Schematic of the School of Fish interface unit. The MAX233 chip translates between RS-232 and RS485 line levels and polarities. The PIC watches the RS-232 line from the host computer for a start bit and, when one is detected, asserts the enable line of the RS485 interface chip (the DS75176) for one byte's worth of time at the current baud rate. Thus from the host computer's point of view, the school of fish is a 3 wire RS232 device—the host does not have to handle the extra handshaking required by RS485. The PIC could also be used as a controller for the school if desired. The host could issue a high level command (such as "scan entire array") to the interface unit, which would then issue more detailed instructions to the members of the school in order to carry out the scan.

B.3.4 R Command

With the transmitter selected, an R command can be issued. This causes the transmitter to transmit and all the recievers to receive, using the current values of the burst parameters. When the measurement is complete, the transmitter sends and acknowledgement character (a Y), so that the host will not issue further commands until the read has concluded. This acknowledgement scheme is not optimal from the point of view of communications bandwidth, and one obvious improvement would be to have the interface unit manage the high level aspects of the process. In fact, the final version of the orientation sensing FieldMouse described in section 6.4 relies on this strategy to achieve a better update rate. More details are given in the subsection below on the Q command, and in section B.2.

B.3.5 P Command

This command causes a specified unit to return the value measured by the most recently issued R command.

B.3.6 Q Command

When the PC issues this command, the interface PIC makes a sequence of 19 measurements, automatically issuing all the necessary T, R, and P commands. The particular sequence of measurements is hardcoded for the electrode geometry used in the orientation sensing 3D mouse demo described in chapter 6.4. This command should be replaced by a more general one that allows the PC to download a sequence of measurements. The particular sequence of commands it issues is
T0 R P1 P5 P6
T1 R P5 P4 P2
T2 R P4 P3
T3 R P4 P9
T4 R P9 P8 P5
T5 R P8 P7 P6
T6 R P7
T7 R P8
T8 R P9

For the 10 electrode geometry described in chapter 6.4, this sequence makes one measurement for each of the $2n-1 = 19$ nearest neighbor electrode pairs. Figure B-2 illustrates the measurement sequence. The arrows indicate transmit-receive pairs, with the arrow directed from transmitter to receiver.

B.3.7 S Command

This command causes the unit to read and return the ID string stored in its Silicon Serial number. This feature could be used to associate a "digital shadow"—that is, a database record containing history or state information—with a particular School of Fish unit. Currently, one must use the unit's ordinary hardcoded unique ID to access the Silicon Serial Number ID. Eventually, it would be preferable to use just the Silicon Serial Number or IDs generated dynamically from noise, rather than the ID burned in to EEPROM, but doing so

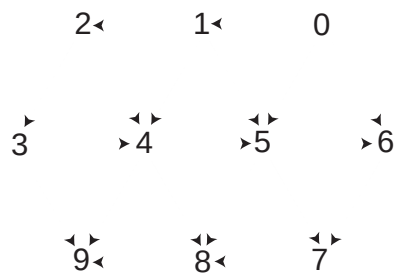

Figure B-2: The sequence of measurements taken by the Q command. The arrows connecting two units indicate a transmit-receive pair. The arrow is oriented from the transmitter to the receiver.

requires implementing more complex identity assignment protocols, such as those explored in Chapter 10.

Another improvement to the communications protocol that might seem attractive is actually very problematic. One could imagine doing the serial I/O in an interrupt driven fashion; then the units could buffer commands received during a transmit or receive operation, and process them afterwards. The problem with this idea is that it is crucial to maintain "radio silence"—or rather, digital silence—during any measurement operation. The hard digital edges used in communication generate spikes in the front end that are as large as the measured signals.

Protocol: 38.4K baud, 8N1

Your Command	Meaning	School Response (e.g.)
Iyz	Change integration parameters	None
O1	Verify integration parameters	050 013[CR]
T2	Transmit	X
R	Sense	Y
P3	Report measured value	255 255[CR]
Q	Make 19 measurements	135 bytes of data
S4	Read Silicon Serial Number	255 255 255 255 255 255 255 255[CR]

B.4 School of Fish Unit Firmware (version `synch371`)

```
#define ID 1
// Joshua R. Smith
// MIT Media Lab (c) 1997
//
// synch37   Added the working SSN code
// synch36   Using txrx35 w/ first LED cmd in, and second moved back into this file
// synch35   Added code to light LED when TXing; switched to txrx35
// synch34   changed to 38400
// synch33   xmit ack char after tx change operation
// synch32   problem: hi byte repeated in low byte
// synch31
//           TX cmd makes this unit the TX, all others the rcv
//           Got rid of G_talkenableflag
//           Changed TX initial state to hi (this is default state)
//           Make cmd to poll for data distinct from cmd to make measurement
//           Get rid of unused qd_hi, qd_lo;
//           replace ip_hi and ip_lo with global mag_hi, mag_lo
//
// synch30   TX and RX set in software...IDs added
//
//           Got working school of fish boards back
// synch29
//           Swapped pins 6 and 9 (to make board layout easier)
```

```
//           Changed from variable G_tanktrisval to const
//           TANKTRISVAL, which takes less time to set.
//
// synch28   First working transceiver!!!
//           Problem with this design: when in RCV mode, INPHASE
//           signal feeds through switch to the front end.
//           Current design: use INPHASE signal for both
//           TX and demod, but interrupt path to tank with switch
//           when we are in RCV mode
//           Added debugging code because couldn't set
//           inner and outer loop values anymore
// synch27   Made A3 TANK; use B4 as inphase demod
// synch26
//           Made pin B6 xcv select (switch choses whether to apply 2.5V [rcv]
//           or 0V / 5V [xmit] for noninverting input to opamp
//           Made pin A2 xmit dipswitch
//
// synch25
//           After cutting integrating cap value, we were able to discharge it quickly
//           enough (switch has on resistance of 100-200 ohms, MOSFET only had 7.5 ohms)
//           This version of code does correct averaging.
// synch21
//           Wait longer for cap to discharge
// synch20
//           (Try intentional dephasing)
//           Only talk on 485 bus if G_talkenableflag (set by jumper IDJUMP) is true
//           Converted to PCW environment
// synch19
//           Increased gain (cut integ cap); make inner loop adjustable
//           Take average magnitude in software
// synch18
//           Add command to measure offset for each channel
//           Send out inphase and quad separately (so we can
//           take sqrt in VB)
// synch17
//           Re-instated quadrature subtraction code.
//           Changed fuses...use high speed crystal oscillator setting
// synch16
//           Reverted to inphase only
//           Tried to correct quadrature-finding...replaced sum with difference
//           Changed shunt from RA1 to RB7
//           Sample second channel too
//           Reverted to no external voltage ref since ADC became
//           imprecise near voltage ref (saw jumps from 144 -> 160).
// synch15
//           Changed to external voltage ref for ADC.
//           Leave integ cap in shunted state all the time, except
//           when we're actually reading.
//           Leave integ cap in shunted state for 50us (1us did nothing)
//           Try clearing integ cap just after reading.
//           Try MOSFET short across integ cap, because there was...
//           problem with shunting cap: op-amp 0v is 2.5V.  Thus
//           our shunt puts a -ve dip into opamp.
//           Clear accum cap by grounding both sides of it before
//           any read.  ***IF THIS DOESN'T WORK, FIND OUT ABOUT ALL_ANALOG
//           Damp Tx osc after we're done driving it by
//           tristating TANK line.
//           No more un-tristating after read.
//           Cut delay from 2uS to 1uS (with extra toggle, freq went
//           down to 50 kHz)
//           Use separate line to toggle switch.  Changed phase of
//           switch to make nice looking demodulated signal.
//           Move to 100kHz (74kHz really)...just sample once at end
//           (do accum in analog)
// synch14   Change to assume demodulation has already been done, ie
//           just add together all samples.
//
//           REMARK on synch13:  May not have eliminated it since
//           behavior of RX depends on measured signal values; thus
//           it will behave differently (out of synch) with TX.
// synch13   Try to eliminate software phase locked loop.
//           This causes strange (non-linear?) behavior: increasing
//           integration time *increases* noise.  Tried setting
//           phase synch absolutely by moving to one pic for both
//           xmit & rcv.  This resulted in expected behavior: longer
//           integration times gave less noise.
// synch12   because we added gain stage, cut outer
//           integration loop
// synch11   experiment: try shorting ADC to see if
//           we can smooth out "steps"
//           un-tristating ADC pin after each read lead to no signal
//           un-tristating after each poll
//           Next: consider integrator (and rectifier demodulator?)
// synch10   converted to 485; back to ordinary
//           added cmd = 0 to see if watchdog failures stop
// synch8    Try to isolate software PLL accidentally
// discovered in synch7: removed all dummy code; phase locking
// stopped.  Replaced dummy code
// synch7.c  Do quadrature sum in pic...this lets us overcome
//           slow phase drift...
// based on
// synch5.c  Quadrature
```

```
// synch4.c  Test for non-infinite input impedance
// synch3.c  Made integration time adjustable param
// synch2.c  Split fish read into separate routine
// based on
// synch1.c  First working Laz-i-fish code

#include <16C71.H>
// #include <stdio.h>
#include <math.c>


#fuses HS,NOWDT,NOPROTECT
#use Delay(Clock=16000000)

/*Pin Declarations*/

// pin 6
#define TX      PIN_B0
// pin 7
#define SSN     PIN_B1
// pin 8
#define DE      PIN_B2
// pin 9
#define RX      PIN_B3


// pin 10
#define IDMOD   PIN_B4
// pin 11
#define QDMOD   PIN_B5
// pin12
#define NOISECTL  PIN_B6
#define TXLED  PIN_B6
// pin 13
#define SHUNT   PIN_B7


// pin 17
#define ADC_IN  PIN_A0
// pin 18
#define ADC_QUAD PIN_A1
// pin 1
#define NOISE_IN PIN_A2
// pin 2
#define TANK    PIN_A3

#use RS232(Baud=38400,Parity=N,Xmit=TX,Rcv=RX)

byte G_inttime_outer;
byte G_inttime_inner;
byte G_iamxmitflag;
byte G_rcvdelay;
static byte G_mag_hi;
static byte G_mag_lo;
byte zero;

void initialize() {
  SET_TRIS_A(0x0F);     // make all of PORTA (RA0-RA3) inputs
  SET_TRIS_B(0x0A);     // RX, SSN are data input
  OUTPUT_LOW(DE);       // Get off RS-485 bus

  OUTPUT_HIGH(TX);
  OUTPUT_HIGH(SHUNT);   // Shunt the integ cap

  SETUP_PORT_A(RA0_RA1_ANALOG);
  SETUP_ADC(ADC_CLOCK_DIV_32);
  SET_ADC_CHANNEL(ADC_IN); // Set ADC channel to inphase
  printf("Initializing");  //for one time diagnostic should be removed if more than one electrode
                           //is being used
  G_iamxmitflag = 0;
  G_rcvdelay = 1;

  zero = 0;
  G_inttime_outer = 50;
  G_inttime_inner = 13;

}



void putdec(byte n) {
putchar( (n/100) +'0');
putchar( ((n/10) % 10) +'0');
putchar( (n % 10) + '0');
}


// RCV
void read_ef() {
#define TANKTRISVAL 0x0F
// don't tristate since we are RCV
```

```
#include <txrx35.c>

}


// TX
void send_ef() {
#define TXCODE
#define TANKTRISVAL 0x07

#include <txrx35.c>

}



void sense() {
    if (G_iamxmitflag) {
      send_ef();
      OUTPUT_HIGH(DE);
    }
    else {
      read_ef();
      OUTPUT_LOW(DE);
    }
    printf("X");
    OUTPUT_LOW(DE);
}


// PRE: 485 has been enabled or not enabled as nec
// compiler bug?  when trying to pass globals directly, we
// get weird problems
// another weird problem: the space (ASC 32) between #s seems to be getting
// turned into 255 or something weird
void ef_out() {
byte lo, hi;

   lo = G_mag_lo;
   hi = G_mag_hi;
   putdec(hi);
   printf(" ");  //***DBG  ??  does one space get dropped?
   putdec(lo);
   printf("\r");
   OUTPUT_LOW(DE);
}


void debug_out() {
  DELAY_US(70);
  OUTPUT_HIGH(DE);          // Send out data
  putdec(G_inttime_outer);
  printf(" ");
  putdec(G_inttime_inner);
  printf("\r");
  OUTPUT_LOW(DE);

}

void ssn_write(int val)
{
             G_trisb = G_trisb & 0xFD;   // Mask off SSN bit
             SET_TRIS_B(G_trisb);        // and un-tristate the SSN pin
             OUTPUT_LOW(SSN);
             DELAY_US(5);
             if (val)
                OUTPUT_HIGH(SSN);
             DELAY_US(60);
             G_trisb = G_trisb | 0x02;   // Set SSN bit
             SET_TRIS_B(G_trisb);        // and re-tristate the SSN pin
             DELAY_US(5);
}

void main () {
byte cmd;

  initialize();
  while(1) {
    cmd = getc();
    if (cmd == 'R') {
      sense();
    }

    if (cmd == 'P') {
       cmd = getc() - '0';
       DELAY_US(70);
       if (cmd == id) {
         OUTPUT_HIGH(DE);          // Send out data
       }
       else {
         OUTPUT_LOW(DE);           // Pretend to send out data
       }
```

```
   ef_out();
}
if (cmd == 'S') {               // Talk to Silicon Serial number
      cmd = getc() - '0';
      if (cmd == id) {

              G_trisb = G_trisb & 0xFD;   // Mask off SSN bit
              SET_TRIS_B(G_trisb);        // and un-tristate the SSN pin
              OUTPUT_LOW(SSN);            // set SSN output low:
              DELAY_US(500);              // generate "reset pulse"
              G_trisb = G_trisb | 0x02;   // Set SSN bit
              SET_TRIS_B(G_trisb);        // and re-tristate the SSN pin
              DELAY_US(5);               // wait for bus to float back up
              i = 0;
              while (input(SSN) && i < 20) {    // wait for low "presence pulse"
                     i++;
                     DELAY_US(5);
                      }
              while (!input(SSN) && i < 40) {   // wait for end of presence pulse
                    i++;
                    DELAY_US(5);
                    }
              if (i>39) {
                  G_trisb = G_trisb | 0x02;  //added this line and the line below to reset
                  SET_TRIS_B(G_trisb);       //if failure
                      }
              else {
                  DELAY_US(50); //I think you need this delay
                  ssn_write(1);
                  ssn_write(1);
                  ssn_write(0);
                  ssn_write(0);
                  ssn_write(1);
                  ssn_write(1);
                  ssn_write(0);
                  ssn_write(0);
                  for(i=0;i<8;i++)
                  {
                    for (j=0;j<8;j++)
                    {
                    serial[i]=serial[i]<<1;
                    G_trisb = G_trisb & 0xFD;   // Mask off SSN bit
                    SET_TRIS_B(G_trisb);        // and un-tristate the SSN pin
                    OUTPUT_LOW(SSN);
                    DELAY_US(5);
                    G_trisb = G_trisb | 0x02;   // Set SSN bit
                    SET_TRIS_B(G_trisb);        // and re-tristate the SSN pin
                    DELAY_US(15);
                    serial[i]=serial[i] & 0xFE;
                    serial[i]=serial[i] | input(SSN);
                    DELAY_US(45);
                   }
                 }


         #use RS232(Baud=38400,Parity=N,Xmit=TX,Rcv=RX)
         DELAY_US(70);
         OUTPUT_HIGH(DE);          // Send out data
         putdec(serial[0]); putc(' ');
         putdec(serial[1]); putc(' ');
         putdec(serial[2]); putc(' ');
         putdec(serial[3]); putc(' ');
         putdec(serial[4]); putc(' ');
         putdec(serial[5]); putc(' ');
         putdec(serial[6]); putc(' ');
         putdec(serial[7]); putc('\r');
         OUTPUT_LOW(DE);

        }
    }
}
if (cmd == 'T') {
  cmd = getc() - '0';
  DELAY_US(70);
  if (cmd == id) {        // is this T message for us?
    G_iamxmitflag = 1;    // become xmitter if so
    G_rcvdelay = 0;
    OUTPUT_HIGH(DE);         // Send out data
  }
  else {
    G_iamxmitflag = 0;    // otherwise become rcver
    G_rcvdelay = 1;
    OUTPUT_LOW(DE);          // Pretend to send out data
  }
  printf("Y");
  OUTPUT_LOW(DE);
}

if (cmd == 'I') {
  G_inttime_outer = getc();
  G_inttime_inner = getc();
}
```

```
    if (cmd == '0') {
      cmd = getc() - '0';
      if (cmd == id) {
        debug_out();
      }
    }

    cmd = 0;
  }
}
```

B.4.1 txrx35.c

The code below is included by `synch371.c`.

```
// Joshua R. Smith
// MIT Media Lab (c) 1997
// txrx35 Added code to light LED on TX;  ***DBG: commented out first LED lighting cmd
// txrx31 Got rid of unused 16-bit quad variable (qd_hi, qd_lo)
// txrx29
// txrx28   First version
// Code that gets included
byte i;
byte j;
byte inphase;
byte quad;

    G_mag_hi = 0;
    G_mag_lo = 0;
    OUTPUT_BIT(TXLED,!G_iamxmitflag);  // turn TX LED on or off
    for (i=0; i < G_inttime_outer; i++) {
      OUTPUT_LOW(SHUNT);      // Stop shunting, start integrating.
      SET_TRIS_A(TANKTRISVAL);   // This makes TANK an output (0x0F --> 0x07) if we are XMITer
      DELAY_US(G_rcvdelay);
      for (j=0; j < G_inttime_inner; j++) {
         // HIGH PART OF CYCLE
         OUTPUT_HIGH(IDMOD);
         OUTPUT_LOW(QDMOD);    // invert
#ifndef TXCODE
         DELAY_US(1);
#else
         OUTPUT_HIGH(TANK);
#endif

         OUTPUT_HIGH(IDMOD);
         OUTPUT_HIGH(QDMOD);   // don't invert cause we're +ve now
#ifndef TXCODE
         DELAY_US(1);
#else
         OUTPUT_HIGH(TANK);
#endif

         // LOW PART OF CYCLE
         OUTPUT_LOW(IDMOD);
         OUTPUT_HIGH(QDMOD);    // don't invert cause we're still +ve
#ifndef TXCODE
         DELAY_US(1);
#else
         OUTPUT_LOW(TANK);
#endif

         OUTPUT_LOW(IDMOD);
         OUTPUT_LOW(QDMOD);     // invert since we're -ve
#ifndef TXCODE
         DELAY_US(1);
#else
         OUTPUT_LOW(TANK);
#endif
      }

      inphase = READ_ADC();
      SET_ADC_CHANNEL(ADC_QUAD);
      inphase = (inphase > 128)? (inphase - 128) : (128 - inphase);
      add16(G_mag_hi, G_mag_lo, zero, inphase);
      //DELAY_US(8);             // can probably reduce this delay
      quad = READ_ADC();
      quad = (quad > 128)? (quad - 128) : (128-quad);
      add16(G_mag_hi, G_mag_lo, zero, quad);
      SET_TRIS_A(0x0F);       // Tristate TANK (this stops ringing) (0x07 --> 0x0F)
      SET_ADC_CHANNEL(ADC_IN);// Set ADC channel back to inphase
      OUTPUT_HIGH(SHUNT);     // Clear integ cap and leave it shunted
      DELAY_US(8);            //  till next read
    }
OUTPUT_HIGH(TXLED);     // turn TX LED off
```

B.5 School of Fish Interface Unit Firmware (version `hack485i`)

The main purpose of this interface code is to watch for RS232 data from the host PC and enable the RS485 transceiver whenever there is. From a hardware point of view, it would be feasible for the PC to manage the RS485 enable line itself, by controlling one of the unused serial pins, such as RTS. However, it turns out that because of long (mSec), unpredictable latencies introduced by the Windows operating system, this would be an inefficent and unreliable solution. One might also wonder about leaving the interface unit's RS485 enable line permanently enabled. However, for a School of Fish sensor unit to transmit its data back, it must have control of the bus, so the interface unit must release the bus when the PC is finished transmitting. The interface unit's code does not need to copy data from the PC to the School because this connection is hardwired. It just controls the enable line that allows the PC's serial output to flow onto the RS485 network.

In addition to this channel sharing function, the interface code also checks for one special command from the PC. When the PC issues the Q command, the interface PIC begins automatically polling the School, collecting a predefined sequence of 19 measurements. This is the sequence of measurements described in section 6.4 for the 10 unit geometry used in the orientation sensing mouse demo. The PC "sees" all the activity on the bus, and when its serial input buffer has filled with the correct number of characters, then (and only then) does it turn its attention to reading and processing the data. Because the interface unit performs the polling operation autonomously, the PC does not have to waste any time issuing the complex sequence of commands (and waiting for responses) needed to collect all the data. It can use this time for the inverse problem and for graphics.

```
// Joshua R. Smith
// MIT Media Lab (c) 1998
//
// hack485i  works!
// hack485h  break xmit/rcv into subs
// hack485g
// hack485f  look at bytes from host
// hack485e  Changed to C711
//           Code changes to correspond to 485 interface changes:
//           (enabled PIC controller to talk on bus)
//           ensure PIC's 485 TX line is normally tristated
//           send test poll message out on RS485 at start up
// hack485c  Changed to 38400 baud
// hack485b  Changed to new compiler format; swapped TX and RX pins
// hack485   Code for 232 <--> 485 transceiver.  Asserts 485 enable whenever necessary
// based on
// synch12
#include <16c71.h>
//#include <stdio.h>
#include <math.c>

#fuses HS,NOWDT,NOPROTECT
#use Delay(Clock=16000000)

/*Pin Declarations*/

// pin 6 not connected
#define TX  PIN_B0
#define T232  PIN_B0

// pin 7
#define R485  PIN_B1

// pin 8
#define DE  PIN_B2

// pin 9
#define RX  PIN_B3
#define R232  PIN_B3
#define T485  PIN_B3


#use RS232(Baud=38400,Parity=N,Xmit=T232,Rcv=R232)

byte G_inttime;
short exitflag;

// Same config routine used by normal guppies
void initialize() {
  SET_TRIS_A(0x0F);     // make all of PORTA (RA0-RA3) inputs
  SET_TRIS_B(0xFB);     // Everything is input except DE
```

```
  OUTPUT_LOW(DE);       // Get off RS-485 bus

  //OUTPUT_LOW(T232);

  SETUP_PORT_A(RA0_RA1_ANALOG);


}

int waitc() {
byte cmd;
long time;

#use RS232(Baud=38400,Parity=N,Xmit=T485,Rcv=R485)
   time = 0;
   exitflag = 0;
   cmd = 0;
   while ((time < 10000) && (exitflag==0)) {
      if (INPUT(R485) == 0) {
         OUTPUT_HIGH(TX);
         cmd = getc();
         OUTPUT_LOW(TX);
         exitflag = 1;
      }
      time = time + 1;
   }

   return(cmd);
}

void xmit(byte id) {
byte cmd;

            OUTPUT_LOW(DE);   // free bus
#use RS232(Baud=38400,Parity=N,Xmit=T485,Rcv=R485)
            SET_TRIS_B(0xF3);     // Untristate T485

            DELAY_US(70);
            OUTPUT_HIGH(DE);
            putc('T');
            putc(id);
            OUTPUT_LOW(DE);
            cmd = getc();  // gets the Y response

            DELAY_US(70);
            OUTPUT_HIGH(DE);
            putc('R');
            OUTPUT_LOW(DE);
            DELAY_US(70);
            cmd = getc();  // gets the X response
            DELAY_US(125);
}

void rcv(byte id) {
byte i,cmd;
#use RS232(Baud=38400,Parity=N,Xmit=T485,Rcv=R485)
            DELAY_US(70);
            OUTPUT_HIGH(DE);
            putc('P');
            putc(id);
            OUTPUT_LOW(DE);
            DELAY_US(70);
            for (i=0; i < 8; i++) {
               cmd = waitc();
            }
            DELAY_US(125);
}

void main () {
byte cmd;
int i;

  initialize();
DELAY_MS(100);
#use RS232(Baud=38400,Parity=N,Xmit=T485,Rcv=R485)
  SET_TRIS_B(0xF3);     // Untristate T485
  DELAY_US(100);
  OUTPUT_HIGH(DE);
  putc('0');
  putc('2');
  OUTPUT_LOW(DE);
#use RS232(Baud=38400,Parity=N,Xmit=T232,Rcv=R232)
  DELAY_US(100);
  SET_TRIS_B(0xFB);     // Everything is input except DE

  DELAY_US(100);

  while(1) {
    if (INPUT(R232) == 0) {// if we see a bit from PC (input goes low)
        OUTPUT_HIGH(DE); // turn on write enable
        //printf(" ");     // wait for one character
        cmd = getc();     // wait for one character
```

```
        if (cmd == 'Q') {
#use RS232(Baud=38400,Parity=N,Xmit=T485,Rcv=R485)
            xmit('0');
            rcv('1');
            rcv('5');
            rcv('6');

            xmit('1');
            rcv('5');
            rcv('4');
            rcv('2');

            xmit('2');
            rcv('4');
            rcv('3');

            xmit('3');
            rcv('4');
            rcv('9');

            xmit('4');
            rcv('9');
            rcv('8');
            rcv('5');

            xmit('5');
            rcv('8');
            rcv('7');
            rcv('6');

            xmit('6');
            rcv('7');

            xmit('7');
            rcv('8');

            xmit('8');
            rcv('9');

#use RS232(Baud=38400,Parity=N,Xmit=T232,Rcv=R232)

        }
    }
    else {
        OUTPUT_LOW(DE);  // lower write enable.
    }
    cmd = 0;  // This is just to make sure while loop happens
  }
}
```

B.6 Musical Dress Code

This code runs on a special (non-sensor) "mothership" School of Fish unit that has been wired to a MiniMidi synth. Both of these are located in the backpack of the musical dress. This code interrogates two ordinary sensor school of fish units, worn in the dress and connected to conductive fabric electrodes. The sensor units run the ordinary synch37 code. Based on the sensor data, this code generates Midi messages that the MidiBoat plays. For debugging purposes, a School of Fish Interface unit may be attached. Then all the activity on the bus can be displayed in terminal window on a PC.

This code is a useful illustration of how to use one School of Fish unit to communicate with other units. In all the other applications of the School, a PC interrogates the units—they do not talk to one another. There are some non-obvious details involved in having one unit communicate with others, in particular, various delays must be inserted for communications across the bus to work properly.

```
#define ID 0
// Joshua R. Smith
// MIT Media Lab (c) 1997
//
// wear3 changed lower level of pitch
// wear1     Change music
// wear0     Code for school of fish mothership
// based on...
// synch34   changed to 38400
// ...
// based on
// synch1.c  First working Laz-i-fish code

#include <16C71.H>
// #include <stdio.h>
#include <math.c>


#fuses HS,WDT,NOPROTECT,PUT
#use Delay(Clock=16000000)

#define hi(x)  (*(&x+1))
#define lo(x)  (*(&x))


/*Pin Declarations*/

// pin 6
#define TX      PIN_B0
// pin 7
#define SSN     PIN_B1
// pin 8
#define DE      PIN_B2
// pin 9
#define RX      PIN_B3


// pin 10
#define IDMOD   PIN_B4
// pin 11
#define QDMOD   PIN_B5
// pin12
#define MIDIOUT  PIN_B6
// pin 13
#define SHUNT   PIN_B7


// pin 17
#define ADC_IN  PIN_A0
// pin 18
#define ADC_QUAD PIN_A1
// pin 1
#define NOISE_IN PIN_A2
// pin 2
#define TANK    PIN_A3

#use RS232(Baud=38400,Parity=N,Xmit=TX,Rcv=RX)

byte G_inttime_outer;
byte G_inttime_inner;
byte G_iamxmitflag;
byte G_rcvdelay;
static byte G_mag_hi;
static byte G_mag_lo;
byte zero;
long int max;
long int min;
```

```
long int val;
byte prev_on;

void initialize() {
  SET_TRIS_A(0x0F);     // make all of PORTA (RA0-RA3) inputs
  SET_TRIS_B(0x0A);     // RX, SSN are data input
  OUTPUT_LOW(DE);       // Get off RS-485 bus

  OUTPUT_HIGH(TX);
  OUTPUT_HIGH(SHUNT);   // Shunt the integ cap

  SETUP_PORT_A(RA0_RA1_ANALOG);
  SETUP_ADC(ADC_CLOCK_DIV_32);
  SET_ADC_CHANNEL(ADC_IN); // Set ADC channel to inphase

  setup_counters(RTCC_INTERNAL, WDT_144MS);

  G_iamxmitflag = 0;
  G_rcvdelay = 1;

  zero = 0;
  G_inttime_outer = 50;
  G_inttime_inner = 13;

}



void putdec(byte n) {
putchar( (n/100) +'0');
putchar( ((n/10) % 10) +'0');
putchar( (n % 10) + '0');
}



void read_ef() {
#define TANKTRISVAL 0x0F
// don't tristate if we are RCV


#include <txrx31.c>
}

void send_ef() {
#define TXCODE
#define TANKTRISVAL 0x07

#include <txrx31.c>
}



void sense() {
    if (G_iamxmitflag) {
      send_ef();
      OUTPUT_HIGH(DE);
    }
    else {
      read_ef();
      OUTPUT_LOW(DE);
    }
    printf("X");
    OUTPUT_LOW(DE);
}


// PRE: 485 has been enabled or not enabled as nec
// compiler bug?  when trying to pass globals directly, we
// get weird problems
// another weird problem: the space (ASC 32) between #s seems to be getting
// turned into 255 or something weird
void ef_out() {
byte loval;
byte hival;

   loval = G_mag_lo;
   hival = G_mag_hi;
   putdec(hival);
   printf(" ");  //***DBG  ??  does one space get dropped?
   putdec(loval);
   printf("\r");
   OUTPUT_LOW(DE);
}


void debug_out() {
  DELAY_US(70);
  OUTPUT_HIGH(DE);          // Send out data
  putdec(G_inttime_outer);
  printf(" ");
  putdec(G_inttime_inner);
```

```
  printf("\r");
  OUTPUT_LOW(DE);

}


void main () {
byte cmd;
byte hival,loval;

#use RS232(Baud=38400,Parity=N,Xmit=TX,Rcv=RX)

  initialize();
  while(1) {

    RESTART_WDT();

//  LOOK FOR stuff from our own body
//  send integ time params
    DELAY_US(70);
    OUTPUT_HIGH(DE);         // Send out data
    putc('I');
    putc(50);
    putc(20);
    putc('T');
    putc('0');
    OUTPUT_LOW(DE);

    cmd = getc();      // gets the Y

    DELAY_US(70);
    OUTPUT_HIGH(DE);         // Send out data
    putc('R');
    OUTPUT_LOW(DE);

    cmd = getc();      // gets the X
    RESTART_WDT();

    DELAY_US(70);
    OUTPUT_HIGH(DE);         // Send out data
    putc('P');
    putc('3');          // read from 3
    OUTPUT_LOW(DE);


    hival = 0;
    hival = 100*(getc()-'0');
    hival += 10*(getc()-'0');
    hival += getc()-'0';
    cmd = getc();       // get the space

    loval = 0;
    loval = 100*(getc()-'0');
    loval += 10*(getc()-'0');
    loval += getc()-'0';
    cmd = getc();       // get the cr

    hi(val) = hival;
    lo(val) = loval;
    val = val-600; // orig range: 600-2000; new range 0-1400
    val = val/(1400/127);
    loval = lo(val);
    loval = loval & 0x7F; // make sure we clip at 127
    loval = (loval < 10)? 10:loval;

     RESTART_WDT();

// Gen MIDI for our body
#use RS232(Baud=31250,Parity=N,Xmit=MIDIOUT,Rcv=RX)

    if (loval > 30) {
      putc(0x90);       // note off
      putc(prev_on);
      putc(0x00);

       putc(0xC0);
      putc(76);      // PAN FLUTE

      putc(0x90);
      putc(loval);
      putc(0x7F);
      prev_on = loval;
    }
  }
}
```

B.6.1 `txrx31.c`

This code below is included by `wear3.c`.

```
// Joshua R. Smith
// MIT Media Lab (c) 1997
//
// txrx31 Got rid of unused 16-bit quad variable (qd_hi, qd_lo)
// txrx29
// txrx28   First version
// Code that gets included
byte i;
byte j;
byte inphase;
byte quad;

    G_mag_hi = 0;
    G_mag_lo = 0;

    for (i=0; i < G_inttime_outer; i++) {
      OUTPUT_LOW(SHUNT);      // Stop shunting, start integrating.
      SET_TRIS_A(TANKTRISVAL);   // This makes TANK an output (0x0F --> 0x07) if we are XMITer
      DELAY_US(G_rcvdelay);
      for (j=0; j < G_inttime_inner; j++) {
         // HIGH PART OF CYCLE
         OUTPUT_HIGH(IDMOD);
         OUTPUT_LOW(QDMOD);    // invert
#ifndef TXCODE
         DELAY_US(1);
#else
         OUTPUT_HIGH(TANK);
#endif

         OUTPUT_HIGH(IDMOD);
         OUTPUT_HIGH(QDMOD);   // don't invert cause we're +ve now
#ifndef TXCODE
         DELAY_US(1);
#else
         OUTPUT_HIGH(TANK);
#endif

         // LOW PART OF CYCLE
         OUTPUT_LOW(IDMOD);
         OUTPUT_HIGH(QDMOD);    // don't invert cause we're still +ve
#ifndef TXCODE
         DELAY_US(1);
#else
         OUTPUT_LOW(TANK);
#endif

         OUTPUT_LOW(IDMOD);
         OUTPUT_LOW(QDMOD);     // invert since we're -ve
#ifndef TXCODE
         DELAY_US(1);
#else
         OUTPUT_LOW(TANK);
#endif
      }

      inphase = READ_ADC();
      SET_ADC_CHANNEL(ADC_QUAD);
      inphase = (inphase > 128)? (inphase - 128) : (128 - inphase);
      add16(G_mag_hi, G_mag_lo, zero, inphase);
      //DELAY_US(8);             // can probably reduce this delay
      quad = READ_ADC();
      quad = (quad > 128)? (quad - 128) : (128-quad);
      add16(G_mag_hi, G_mag_lo, zero, quad);
      SET_TRIS_A(0x0F);        // Tristate TANK (this stops ringing) (0x07 --> 0x0F)
      SET_ADC_CHANNEL(ADC_IN);// Set ADC channel back to inphase
      OUTPUT_HIGH(SHUNT);      // Clear integ cap and leave it shunted
      DELAY_US(8);             //  till next read
    }
```

Appendix C

The MiniMidi Embedded Music Platform (aka The MidiBoat)

C.1 Overview

The MiniMidi is a platform for standalone "no-PC" music applications. It has a Crystal Semiconductor CS9236 General Midi wavetable synth, digital to analog converter, preamp, PIC16F84 microcontroller that can run MIDI sequencer or interactive music software, 8K bytes of content EEPROM, and an LED. Its power requirements are on the hefty side: 9V DC at about 500mA. It has connectors for power, stereo audio out, and one for serial command data in. There are eight uncommited bi-directional digital I/O lines with holes for connections to off-board sensors, actuators, or other devices.

The serial-in connector drives pin B4 (pin 10) of the PIC. Pin B5 (pin 11) of the PIC drives the synth input. When the MiniMidi is running interactive music applications (such as the Musical Jacket), the software performs some complicated mapping to turn the sensor information coming through the serial connector to pin B4 into MIDI notes going out pin B5 to the synth. When you simply want to use the device as a synthsizer (generating the MIDI commands from somewhere other than the PIC), there are two methods. The first is to burn code ("midithru.hex") into the PIC that simply replicates the state of pin B4

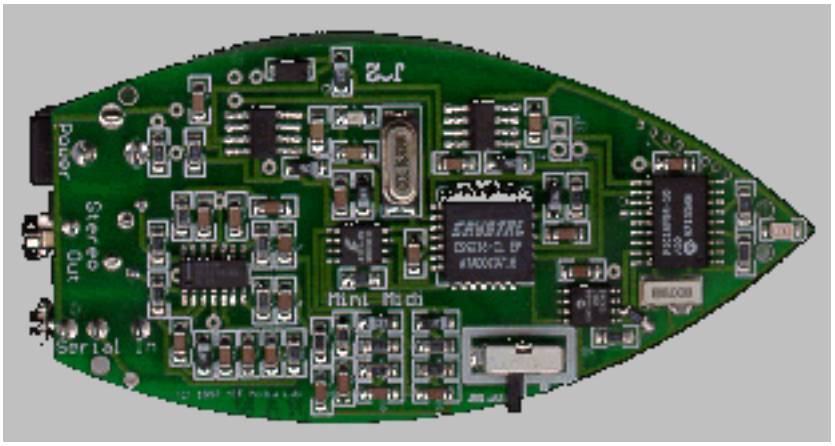

Figure C-1: The Minimidi.

on B5. The second is to connect directly to the MIDI input. In this case, you should burn code into the PIC to tristate pin B5. (Some users have cut the trace instead...please don't mangle our nice board in this way!)

In addition to the connectors and general purpose I/O holes, there are holes that can be stuffed with a two-pin .1" Molex connector if desired. The connector should be stuffed vertically (see figure) to duplicate the function of the serial command data in connector. In this configuration, the top pin is ground, and the bottom pin is the serial data in. The hole to the right of the serial data in pin is connected directly to the synthesize input. If you want to drive the synth directly from an off-board MIDI source, connect here, but remember to tristate pin B5 (pin 11) of the PIC to prevent contention on this line.

C.2 PIC pin assignments

Pin 1 PIN A2 A2—uncommitted digital I/O
Pin 2 PIN A3 A3—uncommitted digital I/O
Pin 3 PIN A4 A4—uncommitted digital I/O
Pin 4 MCLR
Pin 5 VSS (GND)
Pin 6 PIN B0 LED—LED!
Pin 7 PIN B1 B1—uncommitted digital I/O
Pin 8 PIN B2 B2—uncommitted digital I/O
Pin 9 PIN B3 B3—uncommitted digital I/O
Pin 10 PIN B4 KBDIN—"Keyboard in"
Pin 11 PIN B5 MIDIOUT—Midi out
Pin 12 PIN B6 TX—uncommitted digital I/O
Pin 13 PIN B7 RX—uncommitted digital I/O
Pin 14 VDD (+5V)
Pin 15 OSC2
Pin 16 OSC1
Pin 17 PIN A0 EECLK—EEPROM clock line
Pin 18 PIN A1 EEDATA—EEPROM data line

C.3 Warnings, Tips, etc

Don't run the MiniMidi at more than 9V. The synth we used is a 3.3V device, and at the currents drawn by the synth, the regulator that drops the voltage down to 3.3V overheats if its input is more then 9V.

If you want to drive the synth directly (rather than from the on-board PIC), remember to tristate PIC pin B5 (pin 11) so that it doesn't tie the synthesizer input line low or high. Please don't cut the trace!

C.4 MidiBoat Code

C.4.1 `midthru4.c`

This code is for using the MidiBoat as a synthesizer only, not a sequencer. When the boat is powered up, this code plays a start up sequence of notes and LED flashes. When the test

sequence is finished, the LED remains on, and any input from the serial in ("keyboard") line is copied by the PIC to the input of the synthesizer. The initial LED flashes allow the user to verify that the power supply and PIC are working and programmed properly. The test notes show that the speakers are working.

```
// Joshua R. Smith
// MIT Media Lab (c) 1997
//
// midthru4 FLASH LED on start up; leave it on at end
// midthru3
// midthrus follow serial input in softwar...copy to MIDIOUT
// midithru tristates everything so synth can be used directly

#include<16F84.H>
#fuses HS,NOWDT,NOPROTECT,
#USE Delay(Clock=8000000)

#ifndef EEPROM_SDA

//Pin 18
#define EEPROM_SDA  PIN_A1

// Pin 17
#define EEPROM_SCL  PIN_A0

#endif


// Pin 12
#define MODE_JUMPER  PIN_B6

// Pin 13
#define DATAOUT      PIN_B7

//Pin 10
#define CMDIN PIN_B4
//Pin 11
#define MIDIOUT PIN_B5

#USE RS232 (Baud=31250, Parity=N, Xmit=MIDIOUT,Rcv=CMDIN)


#define hi(x)  (*(&x+1))
#define lo(x)  (*(&x))

#use i2c(master,sda=EEPROM_SDA, scl=EEPROM_SCL)

#define EEPROM_ADDRESS long int
#define EEPROM_SIZE    8192


void initialize() {

   SET_TRIS_A(0x1F);
   SET_TRIS_B(0xDF); // Everything but MIDIOUT tristated
   OUTPUT_LOW(MIDIOUT);
}

main () {
byte data;
byte i,j;
byte modeflag;
byte cmd;

   initialize();

   OUTPUT_HIGH(PIN_B0);
   DELAY_MS(500);
   putchar(0x90);
   putchar(0x40);
   putchar(0x50);

   OUTPUT_LOW(PIN_B0);
   DELAY_MS(500);
   putchar(0x90);
   putchar(0x40);
   putchar(0x00);

   OUTPUT_HIGH(PIN_B0);
   DELAY_MS(500);
   putchar(0x90);
   putchar(0x50);
   putchar(0x50);

   OUTPUT_LOW(PIN_B0);
   DELAY_MS(500);
   putchar(0x90);
   putchar(0x50);
   putchar(0x50);
```

```
   OUTPUT_HIGH(PIN_B0);
   DELAY_MS(500);
   putchar(0x99);
   putchar(0x40);
   putchar(0x50);

   OUTPUT_LOW(PIN_B0);
   DELAY_MS(500);
   putchar(0x99);
   putchar(0x40);
   putchar(0x00);

   OUTPUT_HIGH(PIN_B0);
   DELAY_MS(500);
   putchar(0x99);
   putchar(0x50);
   putchar(0x50);

   OUTPUT_LOW(PIN_B0);
   DELAY_MS(500);
   putchar(0x99);
   putchar(0x50);
   putchar(0x50);

   OUTPUT_HIGH(PIN_B0);
   DELAY_MS(500);
   putchar(0x90);
   putchar(0x40);
   putchar(0x50);


   set_tris_b(0xDE);       // all inputs except MIDIOUT and LED (B0)
   set_tris_a(0x1F);

   while(1) {
      OUTPUT_BIT(MIDIOUT, INPUT(CMDIN));
   }

}
```

C.4.2 boatee14.c

This code demonstrates how to read data from the serial EEPROM and then play notes based on this data.

```
// Joshua R. Smith
// MIT Media Lab (c) 1997
//
// boatee14.c go to 38400
// boatee9 This works!!
// boatee8 turn on portb pullups...use grounding of a portb pin
//         to switch modes
// boatee7 stores a sequence and plays it back
// boatee6 test w/ pull up on CLK  This works!!
// boatee2
////////////////////////////////////////////////////////////////////////////
////   Library for a MicroChip 24C65                                    ////
////                                                                    ////
////   init_ext_eeprom();    Call before the other functions are used  ////
////                                                                    ////
////   write_ext_eeprom(a, d);  Write the byte d to the address a      ////
////                                                                    ////
////   d = read_ext_eeprom(a);   Read the byte d from the address a    ////
////                                                                    ////
////   The main program may define eeprom_sda                           ////
////   and eeprom_scl to override the defaults below.                   ////
////                                                                    ////
////////////////////////////////////////////////////////////////////////////

#include<16F84.H>
#fuses HS,NOWDT,NOPROTECT,
#USE Delay(Clock=8000000)

#ifndef EEPROM_SDA

//Pin 18
#define EEPROM_SDA  PIN_A1

// Pin 17
#define EEPROM_SCL  PIN_A0

#endif

// Pin 11
#define SERIAL_IN    PIN_B5

// Pin 12
#define MODE_JUMPER  PIN_B6
```

```
// Pin 13
#define DATAOUT      PIN_B7

//Pin 10
#define CMDIN PIN_B4
//Pin 11
#define MIDIOUT PIN_B5

#USE RS232 (Baud=31250, Parity=N, Xmit=MIDIOUT,Rcv=CMDIN)


#define hi(x)  (*(&x+1))
#define lo(x)  (*(&x))

#use i2c(master,sda=EEPROM_SDA, scl=EEPROM_SCL)

#define EEPROM_ADDRESS long int
#define EEPROM_SIZE    8192



void init_ext_eeprom() {
   output_low(eeprom_scl);
   output_high(eeprom_sda);
}


void write_ext_eeprom(long int address, byte data) {

   i2c_start();
   i2c_write(0xa0);
   i2c_write(hi(address));
   i2c_write(address);
   i2c_write(data);
   i2c_stop();
   delay_ms(11);
}


byte read_ext_eeprom(long int address) {
   byte data;

   i2c_start();
   i2c_write(0xa0);
   i2c_write(hi(address));
   i2c_write(address);
   i2c_start();
   i2c_write(0xa1);
   data=i2c_read(0);
   i2c_stop();
   return(data);
}


#USE RS232 (Baud=38400, Parity=N, Xmit=DATAOUT,Rcv=CMDIN)

void putdec(byte n) {
putchar( (n/100) +'0');
putchar( ((n/10) % 10) +'0');
putchar( (n % 10) + '0');
}


download() {
byte data;
long int addr;

   printf("EEPROM MODE\r");

   init_ext_eeprom();
   addr = 0;
   while (1) {
      data = getc();
      write_ext_eeprom(addr, data);

      data = 0;
      data = read_ext_eeprom(addr);

      putdec(data); putc(' ');

      addr = addr+1;
   }
}



#USE RS232 (Baud=31250, Parity=N, Xmit=MIDIOUT,Rcv=CMDIN)
playback() {
byte data;
long int addr;
byte j;
```

```
   init_ext_eeprom();
   addr = 0;

   while(1) {
      for (j=0; j < 3; j++) {
         data = read_ext_eeprom(addr);
         putchar(data);
         addr += 1;
      }
      delay_ms(500);
   }
}


main () {
byte data;
byte i,j;
byte modeflag;

   set_tris_b(0x50);       // B6, B4 are inputs
   port_b_pullups(TRUE);
   set_tris_a(0x1F);
//   OUTPUT_LOW(DBG);
   DELAY_MS(500);

   modeflag = INPUT(MODE_JUMPER);
   if (modeflag == 0) {
      download();
   }
   else {
      playback();
   }

}
```

C.4.3 midi19.c

This is the code that runs on the MidiBoat in the Musical Jacket. The Musical Jacket uses a fabric keyboard to control this sequencer program.

```
// Joshua R. Smith, Joshua A. Strickon
// MIT Media Lab (c) 1997
//
// Midi 19  implemented random drum changes
// Midi 18  added basis for random drum remapping
// Midi 17
//        *# chord to change to double time
// To do:
//        random shift in drum mode?
//        fix beeping bug.  Is it the 1 or the 9 (or both) we're hearing
//        generate note on when we enter note + drum mode?
//        move cuica off 8?
// Midi 15 added more modes
// Midi14 more modes??
// Midi13 replacing sequencing w/ mode
//    Remapped keys in single note mode
//    Remapped keys in sequence mode
//    Made * and # slow down and speed up sequence mode
//    Modes: 0 single note play w/ program changes on * and #
//           1 sequence
// Midi12
//       moved midi_out call into int routine from main
//       light LED
// Modified for new MIDIBOAT
// Midi11 Add tempo adjust in drum mode
// Midi10 Add sequenced layers
// Midi9  Add more sequence algorithms
// Midi8  Skip over ordinary note ons on chord
// Midi7  Add Strickon's Midi sequencer
// Midi6  Add the null chord (didn't work)
// Midi5  Recognize chords (applause on and off)
// Midi4  F84 instantiation of "Device independent" version of Midi1
// Midi3  14000 instantiation of "Device independent" version of
// Midi1  basic test version

#include <16F84.H>

#fuses HS,NOWDT,NOPROTECT,PUT
#use Delay(Clock=8000000)

/*Pin Declarations*/


// pin 18
#define EEDATA  PIN_A1
// pin 17
#define EECLK   PIN_A0
```

```
// pin 13
#define RX      PIN_B7
// pin 12
#define TX      PIN_B6

// pin 11
#define MIDIOUT PIN_B5
// pin 10
#define KBDIN   PIN_B4

// Force SW?
//#use I2C(MASTER,SDA=EEDATA,SCL=EECLK,SLOW)

// default tempo
#define tempo0_default 8
#define tempo1_default 15
#define prog_default 19


void pic_specific_tris() {
  SET_TRIS_B(0x90);     // RX & KBD are data input

}

byte kbdhimap[6];
byte kbdlomap[6];
byte kbdhi_old;
byte kbdlo_old;
byte kbdlo_tmp;
byte kbdhi;
byte kbdlo;
short fastquant=0;
unsigned int notepc = prog_default;
unsigned int mode = 0;

byte cycle;
byte drum_offset;
byte chord_old;
byte chord;
//byte sequencing;
byte beat;
byte bassdrum;
byte snare;
byte openhh;
byte closedhh;
byte handclap;
byte cowbell;
byte tambourine;
byte hibongo;
byte lobongo;
byte whistle;
byte opencuica;
byte mutecuica;
byte beat_timer;
byte tempo;
byte tempo_old;
byte appregio_high;
byte appregio_low;
byte appregio;

//#use RS232(Baud=38400,Parity=N,Xmit=TX,Rcv=RX)

void initialize() {
  pic_specific_tris();
  cycle = 0;
  drum_offset = 0;
  kbdhi_old = 0;
  kbdlo_old = 0;
  kbdhi = 0;
  kbdlo = 0;
  chord_old = 0;
  chord = 0;
  fastquant = 0;
  appregio_high=127;
  appregio_low=0;
  appregio=0;
//  sequencing = 0;
  mode = 0;
  tempo=tempo0_default;
  bassdrum=171;
  snare=136;
  openhh=4;
  closedhh=255;
  handclap=212;
  cowbell=59;
  tambourine=85;
  hibongo=75;
  lobongo=180;
  whistle=136;
  opencuica=0x49;
  mutecuica=0xB6;
```

```
/*
  // Chromatic, with mode change interrupts
  kbdlomap[0] = 71;
  kbdlomap[1] = 70;
  kbdlomap[2] = 69;
  kbdlomap[3] = 68;
  kbdlomap[4] = 67;
  kbdlomap[5] = 66;
  kbdhimap[0] = 65;
  kbdhimap[1] = 64;
  kbdhimap[2] = 63;
  kbdhimap[3] = 62;
  kbdhimap[4] = 61;
  kbdhimap[5] = 60;
*/
  // Major chords
  kbdhimap[5] = 55;     // 1 key
  kbdhimap[1] = 59;     // 2 key
  kbdlomap[3] = 62;     // 3 key
  kbdhimap[4] = 53;     // 4 key
  kbdhimap[0] = 57;     // 5 key
  kbdlomap[2] = 60;     // 6 key
  kbdhimap[3] = 48;     // 7 key
  kbdlomap[5] = 52;     // 8 key
  kbdlomap[1] = 55;     // 9 key
  kbdhimap[2] = 55;     // * key
  kbdlomap[4] = 59;     // 0 key
  kbdlomap[0] = 62;     // # key

}



//void putdec(byte n) {
// putchar( (n/100) +'0');
// putchar( ((n/10) % 10) +'0');
// putchar( (n % 10) + '0');
//}

//void pretty() {
//   printf(" pretty good Hello World!");
//}

void sequence_out(){
#use RS232(Baud=31250,Parity=N,Xmit=MIDIOUT,Rcv=KBDIN)   // talk to MIDI synth
byte temp;
temp=beat>>1;

putc(0x99);
if(bit_test(bassdrum, temp))
   {putc(0x24);
    putc(0x7f);
    }
if(bit_test(snare, temp))
   {putc(0x26);
    putc(0x7f);
    }
if(bit_test(openhh,temp))
   {putc(0x2a);
   putc(0x7f);
   }
if(mode==1){
if(kbdhi & 0x20) {      // 1 key
   if(bit_test(closedhh,temp)) {
       putc(((0x2a-35+drum_offset)%46)+35);
       putc(0x7f);
   }
   }
if(kbdhi & 0x02) {      // 2 key
   if(bit_test(handclap,temp)) {
       putc(((0x27-35+drum_offset)%46)+35);
       putc(0x7f);
   }
   }
if(kbdlo & 0x08) {      // 3 key
   if(bit_test(cowbell,temp)) {
       putc(((0x38-35+drum_offset)%46)+35);
       putc(0x7f);
   }
   }
 if(kbdhi & 0x10) {     // 4 key
   if(bit_test(tambourine,temp)) {
       putc(((0x36-35+drum_offset)%46)+35);
       putc(0x7f);
   }
   }
if(kbdhi & 0x01) {      // 5 key
   if(bit_test(hibongo,temp)) {
       putc(((0x3c-35+drum_offset)%46)+35);
       putc(0x7f);
   }
```

```
   }

 if(kbdlo & 0x04) {     // 6 key
   if(bit_test(lobongo,temp)) {
       putc(((0x3d-35+drum_offset)%46)+35);
       putc(0x7f);
}
   }
if(kbdhi & 0x08) {      // 7 key
   if(bit_test(whistle,temp)) {
       putc(((0x47-35+drum_offset)%46)+35);
       putc(0x7f);
   }
}

if(kbdlo & 0x20) {      // 8 key
   if(bit_test(opencuica,temp)) {
       putc(((79-35+drum_offset)%46)+35);
       putc(0x7f);
   }
}

if(kbdlo & 0x02) {      // 9 key
   if(bit_test(mutecuica,temp)) {
       putc(((78-35+drum_offset)%46)+35);
       putc(0x7f);
   }
}


if(kbdhi  & 0x04){  // *
   tempo += 1;
   tempo = (tempo > 200)?200:tempo;
}


if(kbdlo & 0x01){   // #
   tempo -= 1;
   tempo = (tempo < 5)?5:tempo;
}


}
}

void midi_out() {
byte kbdhi_new;
byte kbdlo_new;
byte i;
byte mask;

#use RS232(Baud=31250,Parity=N,Xmit=MIDIOUT,Rcv=KBDIN)   // talk to MIDI synth


   kbdhi_new = kbdhi ^ kbdhi_old; // Take XOR to identify changes
   kbdlo_new = kbdlo ^ kbdlo_old;


   // Play individual notes
   mask = 1;
   for (i=0; i < 6; i++) {
     if (kbdhi_new & mask) {
      if (kbdhi & mask) {    // hi note on
         if(mode==0 || mode==2) {
           putchar(0xC0);          //   Program change on channel 0
           putchar(notepc);
           putchar(0x90);          //   Note on on channel 0
           putchar(kbdhimap[i]);   //   Pitch
           putchar(0x60);          //   default velocity
           if (i==2) { // *
             notepc -= 1;
             notepc &= 0x7F;
           }
         }
      }
      else {                  // hi note off
         putchar(0x90);          //   Note off on channel 0
         putchar(kbdhimap[i]);   //   Pitch
         putchar(0x00);
      }
     }
     if (kbdlo_new & mask) {
      if (kbdlo & mask) {    // lo note on
         if(mode==0 || mode==2) {
           putchar(0xC0);          //   Program change on channel 0
           putchar(notepc);
           putchar(0x90);          //   Note on on channel 0
           putchar(kbdlomap[i]);   //   Pitch
           putchar(0x60);          //   default velocity
           if (i==0) { // # in mode 0
             notepc += 1;
             notepc &= 0x7F;
```

```
          }
         }
         if (i==4) { // 0 key in *any* mode
            putc(0xC1);     // prog change on 1
            putc(123);      // bird tweet

            putc(0x91);     // play note
            putc(60);       // bird w
            putc(0x60);
            mode += 1;
            mode = (mode > 2)?0:mode;  // increment to add modes
         }
      }
      else {                     // lo note off
         putchar(0x90);          //   Note off on channel 0
         putchar(kbdlomap[i]);   //   Pitch
         putchar(0x00);
      }
     }
     mask = mask << 1;
   }
   if( (kbdhi_new & 0x04) && (kbdlo_new & 0x01) )  {   // * + #
      if ((kbdhi & 0x04) && (kbdlo & 0x01) ) {
        fastquant ^= 1;
      }
   }


   kbdhi_old = kbdhi;           // save new kbd state
   kbdlo_old = kbdlo;


}




#int_rtcc
rtcc_isr(){
if (--beat_timer==0){
  cycle = cycle+1;
  if ((cycle % 127)==0) {
    drum_offset = drum_offset+9;
  }
  beat=beat+1;
  if (fastquant) {
    midi_out();
    OUTPUT_BIT(PIN_B0, !(beat & 1));
  }
  else{
    if (!(beat & 0x01)) {
      midi_out();
      OUTPUT_BIT(PIN_B0, !(beat & 0x02));
    }
  }

  if (beat==16)
    {beat=0;}
  if((mode==2 || mode==1) && !(beat&0x01)) {
     sequence_out();
  }
  beat_timer=tempo;
  }

}



void main () {
byte cmd;

initialize();
set_rtcc(0);
setup_counters(RTCC_INTERNAL,RTCC_DIV_64);
enable_interrupts(RTCC_ZERO);
enable_interrupts(GLOBAL);
  while(1) {

    cmd = 0;
#use RS232(Baud=19200,Parity=N,Xmit=TX,Rcv=KBDIN,INVERT)   // talk to fabric kbd
//#use RS232(Baud=38400,Parity=N,Xmit=TX,Rcv=RX)            // talk to host for debugging only
    cmd = getc();

//    if (cmd & 0x80) {
//      if (cmd & 0x40) {
//         kbdhi = cmd;
//      }
//      else {
//         kbdlo = cmd;
//      }
    if (cmd & 0x80) {      // try this?
```

```
      if (cmd & 0x40) {
         kbdhi = cmd;
         kbdlo = kbdlo_tmp;
      }
      else {
         kbdlo_tmp = cmd;
      }
    }
  }
}
```

Bibliography

[ACS94] B. Awerbuch, L. Cowen, and M. Smith. Efficient asynchronous distributed symmetry breaking. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, Montreal, Quebec, Canada, 1994.

[BB84] D.C. Barber and B.H. Brown. Applied potential tomography. *Journal of Physics E*, 17:723–733, 1984.

[Ber89] J.G. Berryman. Fermat's principle and nonlinear traveltime tomography. *Physical Review Letters*, 62:2953–2956, 1989.

[Ber90] J.G. Berryman. Stable iterative reconstruction algorithms for nonlinear traveltime tomography. *Inverse Problems*, 6:21–42, 1990.

[Ber91] J.G. Berryman. *Lecture notes on Nonlinear Inversion and Tomography: I. Borehole seismic tomography.* http://sepwww.stanford.edu/sep/berryman/NOTES/lecture_notes.html, 1991.

[BGM91] W. Bender, D. Gruhl, and N. Morimoto. Techniques for data hiding. In *Proceedings of the SPIE*, pages 2420–2440, San Jose, CA, February 1991.

[BH93] M.F. Barnsley and L.P. Hurd. *Fractal Image Compression.* AK Peters, Ltd., Wellesley, Massachusetts, 1993.

[BK90] J.G. Berryman and R. V. Kohn. Variational constraints for electrical-impedance tomography. *Physical Review Letters*, 65:325–328, 1990.

[BOD95] F.M. Boland, J.J.K. O'Ruanaidh, and C Dautzenberg. Watermarking digital images for copyright protection. In *Proceedings, IEE International Conference on Image Processing and its Application*, Edinburgh, 1995.

[Bre91] W. Breckon. Measurement and reconstruction in electrical impedance tomography. In *Inverse problems and imaging*, volume Research Notes in Mathematics 245, pages pp130–140. Pitman, 1991.

[CKLS96] I. Cox, J. Kilian, T. Leighton, and T. Shamoon. A secure, robust watermark for multimedia. In *Proceedings of the First Information Hiding Workshop*, Cambridge, UK, 1996.

[Cor95] Digimarc Corporation. Identification/authentication coding method and apparatus. *U.S. Patent Application*, June 1995.

[Dij65] E.W. Dijkstra. Solution of a problem in concurrent programming control. *Communications of the ACM*, 8(9):569, 1965.

[Dix94] R.C. Dixon. *Spread Spectrum Systems with Commercial Applications.* John Wiley and Sons, New York, 1994.

[FCA60] R.M. Fano, L.J. Chu, and R.B. Adler. *Electromagnetic Fields, Energy, and Forces.* John Wiley & Sons, New York, 1960.

[GS94] N. Gershenfeld and J.R. Smith. Displacement-current method and apparatus for resolving presence, orientation, and activity in a defined space. *U.S. Patent Application*, February 3, 1994.

[GS98] N. Gershenfeld and J.R. Smith. Displacement-current sensor and method for three-dimensional position, orientation, and mass distribution. *U.S. Patent Number 5844415*, December 1, 1998.

[Gul88] S. Gull. Developments in maximum entropy data analysis. In *Maximum Entropy and Bayesian Methods*, Cambridge, 1988. Kluwer.

[Hei77] W. Heiligenberg. *Studies of Brain Function, Vol. 1: Principles of Electrolocation and Jamming Avoidance.* Springer-Verlag, New York, 1977.

[HH90] P. Horowitz and W. Hill. *The Art of Electronics.* Cambridge University Press, Cambridge, England, 1990.

[Hum48] D. Hume. *An Enquiry Concerning Human Understanding.* 1748.

[Isa89] V. Isakov. *Inverse Source Problems.* Mathematical Surveys and Monographs of the AMS, Providence, RI, 1989.

[Jay83] E.T. Jaynes. Prior information and ambiguity in inverse problems. *SIAM AMS Proceedings*, 14:151–166, 1983.

[Kan83] I. Kant. *Prolegomena to any Future Metaphysics.* 1783.

[Kat78] M.B. Katz. *Questions of Uniqueness and Resolution in Reconstructions from Projections.* Springer-Verlag, Berlin, 1978.

[KS88] A.C. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging.* IEEE Press, New York, 1988.

[KV83] R.V. Kohn and M. Vogelius. Identification of an unknown conductivity by means of measurements at the boundary. *SIAM AMS Proceedings*, 14:113–123, 1983. A less formal and more accessible introduction to the approach developed in the other papers by the authors.

[KV84] R.V. Kohn and M. Vogelius. Determining the conductivity by boundary measurements. *Communications on Pure and Applied Mathematics*, 38:289–298, 1984.

[KV85] R.V. Kohn and M. Vogelius. Determining the conductivity by boundary measurements II, interior results. *Communications on Pure and Applied Mathematics*, 38:643–667, 1985.

[LP90] R.J. Lipton and A. Park. The processor identity problem. *Information Processing Letters*, 36:91–94, 1990.

[Mac91] D.J.C. MacKay. *Bayesian Methods for Adaptive Models.* PhD thesis, CalTech, 1991.

[Mar80] D. Marr. Visual information processing: the structure and creation of visual representations. *Philosophical Transactions of the Royal Society of London B*, 290:pp. 199–218, 1980.

[Max73] J.C. Maxwell. *A Treatise on Electricity and Magnetism.* 1873.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, New York, 1995.

[P+92] W. Press et al. *Numerical Recipes.* Cambridge University Press, Cambridge, England, 1992.

[Pfi96] B. Pfitzmann. Trials of traced traitors. In *Proceedings of the First Information Hiding Workshop*, Cambridge, UK, 1996.

[PG97] J.A. Paradiso and N. Gershenfeld. Musical Applications of Electric Field Sensing. *Computer Music Journal*, 21(2), 1997.

[Rab82] M.O. Rabin. The choice coordination problem. *Acta Informatica*, 17:121–134, 1982.

[SCI91] E. Somersalo, M. Cheney, D. Isaacson, and E. Isaacson. Layer stripping: a direct numerical method for impedance imaging. *Inverse Problems*, 7:899–926, 1991.

[Sem96] Dallas Semiconductor. *Automatic Identification Data Book.* Dallas Semiconductor, Dallas, Texas, 1996.

[Sez87] A. Sezginer. The inverse source problems of magnetostatics and electrostatics. *Inverse Problems*, 3:L87–L89, 1987.

[She95] T.J. Shepard. *Decentralized Channel Management in Scalable Multihop Spread-Spectrum Packet Radio Networks.* PhD thesis, Massachusetts Institute of Technology, July 1995.

[Sli33] L.B. Slichter. The interpretation of the resistivity prospecting method for horizontal structures. *Journal of Applied Physics*, 4:pp.307–322, 1933.

[Smi96] J.R. Smith. Field Mice: Extracting hand geometry from electric field measurements. *IBM Systems Journal*, 35(3 and 4), 1996.

[Smi98] J.R. Smith. *Electric Field Imaging.* PhD thesis, Massachusetts Institute of Technology, August 1998.

[SOSL94] M.K. Simon, J.K. Omura, R.A. Scholtz, and B.K. Levitt. *The Spread Spectrum Communications Handbook.* McGraw-Hill, New York, 1994.

[SU87] J. Sylvester and G. Uhlman. A global uniqueness theorem for an inverse boundary value problem. *Annals of Mathematics*, 125:153–169, 1987.

[SW49] C.E. Shannon and W.W. Weaver. *The Mathematical Theory of Communication.* The University of Illinois Press, Urbana, Illinois, 1949.

[Tak81] F. Takens. Detecting strange attractors in turbulence. In Rand and Young, editors, *Dynamical Systems and Turbulence*, pages pp.366–381. Springer-Verlag, 1981.

[Tan89] A.S. Tanenbaum. *Computer Networks*. Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[Tur47] A.M. Turing. Intelligent Machinery. In B. Meltzer and D. Michie, editors, *Reprinted in Machine Intelligence 5*. Halsted Press, New York, 1947.

[Web89] J.G. Webster, editor. *Electrical Impedance Tomography*. Adam Hilger, New York, 1989.

[ZSP+95] T.G. Zimmerman, J.R. Smith, J.A. Paradiso, D. Allport, and N. Gershenfeld. Applying electric field sensing to human-computer interfaces. In *CHI 95 Human Factors in Computing Systems*, pages 280–287, Denver, Co, 1995. ACM Press.