Universiteit Leiden
Opleiding Informatica

Algorithms and models

for radio tomographic imaging

Name: Alyssa Milburn

Date: 27/08/2014

1st supervisor: Walter Kosters
2nd supervisor: Joost Batenburg

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

Abstract

This thesis discusses the use of sensor networks for radio tomographic imaging and the development of a software framework for experimenting with such networks, including implementation of algorithms and models for the image reconstruction, and practical results from experiments using real-world networks.

1 Introduction

Tomography, in general, is a technique for imaging physical objects (such as human bodies, and building structures) by means of observing the effect those physical objects have upon waves passing through them. The most well-known tomography method is *computed tomography* (CT) [14], for medical imaging using X-rays, and another is used to produce images from MRI scans.

One example of a relatively early application of radio waves (ignoring MRI, which uses them only to monitor changes in the magnetic field) for tomographic imaging can be found in *microwave tomography* for medical imaging, which is a specialised case of computed tomography; some early work on this is summarised by [18], while two modern algorithms are compared in [13]. Possible applications include detection of tissue malignancies (breast and lung cancer), as well as brain and cardiac imaging [24].

In recent years, inexpensive special-purpose radio hardware using "unregulated" radio spectrum (the "ISM bands", such as 2.4GHz) has become widely available, for using communication technologies such as WiFi (IEEE 802.11) and Bluetooth. One such technology which has become more popular in recent years is *ZigBee* (IEEE 802.15.4) [1], which is designed to be a simpler low-power communication system, and the availability of cheap ZigBee devices has led to it being affordable to perform experiments involving large numbers of radio devices.

Importantly, this type of radio hardware often provides easy access to the *Received Signal Strength Indicator* (RSSI), which is the most obvious (and widely-available) link statistic: an approximation of the signal "strength", the amount of power being received.

The idea of what we will call *radio tomography* is simple: we have a set of several radio sensors, generally combined transmitters/receivers — which we will call *nodes* — and we use them to generate an image of the area enclosed by the sensors.

Using an appropriate model which describes the connection between some data (such as the RSSI) from the links between nodes (using the "mesh" formed by their transmission links) and the objects in the space between the nodes (such as a room or an outside area) — which will have an attenuating effect on the radio waves — we can then solve the so-called "inverse problem" and reconstruct a real-time "image" of these objects.

Figure 1.1 illustrates such a *sensor network*. The object inside the area surrounded by the sensors will block the direct path between some nodes. In the illustrated idealised case, those links are marked with blue lines. It is clear that, even without knowing where the object is, an observer can easily derive an approximate location once they see which links are blocked.

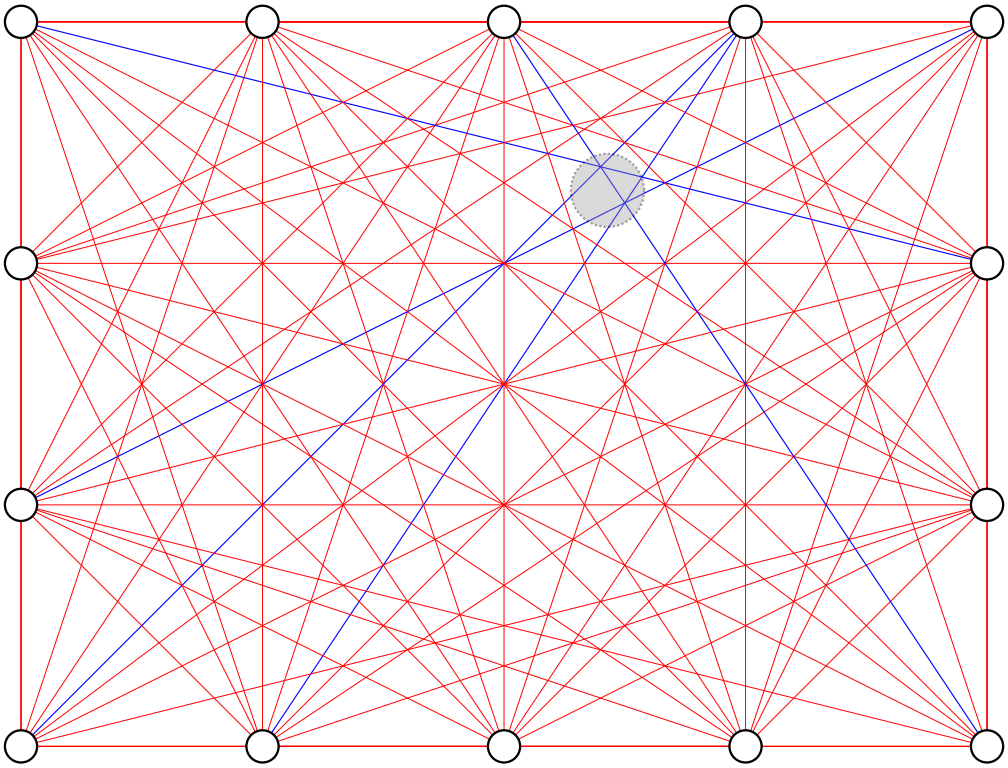


Figure 1.1: A sensor network and an attenuating object.

In 2007, researchers at the University of Iowa used TI's CC2420 ZigBee RF transceiver chip with low-end microprocessors to construct such a sensor network. With the sensors arranged in a circle, and using traditional CT reconstruction algorithms, they sent packets (bursts of data) between the sensors to approximate a standard "fan beam" projection, and then used the RSSI data associated with these packets to produce images which correctly located "phantom" objects (which are made of water, for use in tomographic experiments) [12].

A related field is that of *localization*, restricting the problem to that of determining the location of humans. Many different approaches to using radio signals for this purpose have been considered, and in 2008, Neal Patwari and Joey Wilson at the University of Utah used the newer CC2430 and an algebraic reconstruction algorithm to apply radio tomography to the problem of localization.

They used radio tomography to perform *device-free localization* ("device-free" referring to the fact that the humans are not required to carry a device with them) and produce a real-time image of the motion of humans [22, 30].

Their approach, which they called *radio tomographic imaging* (RTI), is the one which we will be focusing on in this thesis.

We begin, in Section 2, by considering the different possibilities for hardware, and discuss some relevant related work. In Section 3, we discuss the basic models and algorithms used in RTI, and the problems and challenges involved, and then discuss the implementation of a software framework for experimentation in Section 4. Sections 5 and 6 cover the results of simulations and real-world experiments respectively, and we finish by summarising the results and discussing possible directions for future work.

This bachelor thesis was written under the supervision of Walter Kosters (at Leiden University) and Joost Batenburg (at CWI, in Amsterdam), with thanks to the various people at both institutions who lent assistance, in particular Tim van der Meij, Folkert Bleichrodt and Willem Jan Palenstijn.

2 Radio devices

In this section, we investigate some of the various available and affordable radio devices for building a sensor network using the (unregulated) 2.4GHz spectrum, and the existing work which makes use of them.

2.1 ZigBee

The existing research from Iowa and Utah used sensor nodes based on TI's ZigBee SoC chipsets, which made them the obvious first choice for our experimentation. They have an integrated processor based on the 8051, so they can run all the necessary sensor code without needing a separate processor. Tim van der Meij modified the existing Utah sensor node software ("multi-Spin") so that it could be built with an open-source toolchain, allowing us to easily experiment with changes to this sensor code [28].

We constructed 18 sensor nodes comprised of battery packs connected to TI evaluation boards for the TI CC2530 ZigBee chipset, which included the CC2591 amplifier. We later purchased another 18 sensor nodes in the form of third-party CC2530 development boards (without the amplifier) from a Chinese manufacturer.

Utah's multi-Spin [8] makes relatively limited use of the available functionality, using only the standard ZigBee channels (16 channels spaced at 5MHz steps), and collecting one RSSI statistic for every received packet. The hardware exposes a lot more functionality [3], some of which appeared potentially useful for our work.

The frequency can be adjusted in a less coarse fashion, both in 1MHz steps, and (by adjusting the frequency tuning) in 16 fine-grained steps of about 10kHz. There is a "test mode" which allows transmission of a sine wave (a "baseband tone") at various frequencies, most importantly at 4kHz and 500kHz, which allowed us to easily use a cheap software radio to verify the actual frequencies being used.

The chipset can also be configured to provide incoming data (not raw waveforms, but from the analog-to-digital converter) on the I/O pins of the device, but this has been confirmed as unreliable and unsupported functionality by previous research [4].

Coarser information about signal quality is available from the chipset in the form of the "chip correlation rate", intended for use in calculating the ZigBee LQI (Link Quality Indicator). While some previous research [4] again claimed this was a useless statistic which was almost always constant, other research has found it to be more useful [27], and we found that it provided useful information as long as the chipset was correctly

configured to allow low-quality packets to be received (in particular, the demodulator correlator threshold value must be low). Tim van der Meij further modified our local variant of the multi-Spin software to support collection of this additional information.

Finally, as well as providing a per-packet RSSI value — calculated as the averaged RSSI value over the first 8 symbols (one symbol corresponds to 4 bits of data) of the packet — the CC2530 also provides an updated averaged RSSI value (over the last 8 symbols seen) after every new symbol has been received, which means that RSSI changes over the time spent receiving a packet can also be observed.

The raw RSSI value provided by the TI radios is not linear in relation to actual received signal strength, but it has been previously shown that the CC2420 RSSI measurements follow a consistent response curve and it is thus possible to use calibration data to approximate a linear model [11], and the results for the CC2530 are similar.

An alternative ZigBee chipset which also supports relatively low-level access to these hardware details is the Atmel AT86RF230, which is also available in SoC form. However, it didn't provide any obvious advantage for our experimentation.

2.2 Software-defined radio

One obvious approach to localization is to use radar-like techniques; researchers at MIT used several commercial software-defined radios (radio systems which perform processing using software rather than hardware implementations) to detect humans (and identify some simple gestures) through walls [6], and then improved this to perform 3D tracking of the position of a human body [5].

The idea of making use of such software-defined radios for radio tomography in general is a promising one. The "standard" such device is currently the USRP, but while it is reasonably affordable compared to equipment such as vector network analyzers, it is still too expensive for our work. More recently, cheaper alternatives have become available, such as the BladeRF, but even the cheapest such devices remain around 10 times more expensive than the other options we consider here — although only a few (or even one single) such devices are needed for some alternative approaches.

However, one possibility for cheap software-defined radio receivers involves devices based on the RTL2382U chipset; this is commonly referred to as "RTL-SDR". They take advantage of the (undocumented) possibility of configuring this chipset (intended for use for receiving DVB-T television signals in low-cost consumer equipment) to output raw waveforms instead of decoding the signal internally, allowing them to receive a selection of different channels simultaneously.

Such devices are not capable of transmitting radio signals themselves, and so they are only useful in combination with other transmitters. One approach could be to use a sensor network comprised solely of such receivers, and to take advantage of existing radio

signals; for example, one experiment [23] studied the possibility of localization with the use of RTL-SDR devices which monitored the effect of movements on transmissions by commercial FM radio stations.

However, we wanted instead to evaluate the possibility of using such receivers in combination with our ZigBee sensor nodes. The tuner chips used in RTL-SDR devices are selected for the reception of DVB-T, which is transmitted using frequencies between 400 and 900 MHz. While most tuners have wider frequency ranges (up to 2.2GHz), these devices cannot be directly used with the 2.4GHz frequency band.

In early 2014, Omri Iluz described [15] a successful attempt to decode 2.4GHz Bluetooth LE signals by combining an RTL-SDR device with a MMDS-LNB — a low-noise block downconverter for a wireless technology known as MMDS — which receives the input signal within the desired range (around 2.4GHz) and then "downverts" it to a lower frequency.

We purchased both a RTL-SDR USB "dongle" (with a R820T tuner, with a range of 24–1766 MHz) and a MMDS-LNB which downverted the signal around 2.4GHz by 1998 MHz, bringing it fully inside the 400–900 MHz design range of the dongle. It proved difficult to obtain a LNB with a suitable (low) level of gain — important in order to avoid overload — since such LNBs are designed to receive signals over long distances. We compensated by reducing the power levels of the transmitting equipment, and adding attenuators between the antenna and the dongle.

The RTL2382U samples both in-phase (I) and quadrature (Q) components of the input signal, and reliably outputs pairs of 8-bit samples at a rate of up to 2.56MS/s (million samples per second). ZigBee at 2.4GHz uses a DSSS encoding (O-QPSK), and transmits 2 million chips (encoded bits) per second, i.e. 1Mbps on both the I and Q phases [3]. As such, we need at least 2MS/s of bandwidth to receive it, and so the sampling rate of the RTL-SDR is sufficient to allow us to decode ZigBee signals.

An experimental GNU Radio plugin had already been developed for decoding ZigBee packets [7], and we built this and successfully captured packets (both the test packets from our sensor nodes, and packets from devices in the surrounding environment, such as a light bulb controller hub which turned out to use ZigBee).

We modified the plugin to output some additional statistics; most importantly, the chip correlation rate (a good approximation of signal quality). We failed, however, to obtain or calculate any approximation of the signal strength (for example, using the amplitude of the signal); some software radio devices include dedicated hardware for obtaining the RSSI at an early stage in the signal processing.

2.3 Bluetooth

The Bluetooth wireless standard also uses frequencies in the 2.4GHz band, and devices using Bluetooth technology are widely available — support is near-universal in modern mobile phones for use with wireless headsets.

However, generally no useful signal quality or strength information is provided by Bluetooth chipsets and devices, and Bluetooth "hops" between 79 different channels (which are 1MHz wide each) around 1600 times per second, which makes it near-impossible to follow (as opposed to seeing only extremely short bursts of data on whichever frequency is being observed) with cheap hardware which is not specifically designed for use with Bluetooth.

We experimented with the "debug mode" of Bluetooth chipsets from one manufacturer, as discussed in [26], and confirmed that it allows the hopping functionality to be disabled. It also proved possible to generate sine waves for testing purposes using this functionality, but we concluded that — since we can't track Bluetooth devices without specifically placing them into a debugging mode — the hardware is less useful for our purposes than the ZigBee nodes.

However, since Bluetooth hardware is extremely cheap (USB Bluetooth dongles are often available for around €1 each, in bulk), it might be an interesting future source of cheap hardware for networks with extremely large numbers of nodes.

2.4 WiFi

WiFi (IEEE 802.11) [2] is a highly popular networking system, which also often makes use of the 2.4GHz band. It uses a much larger channel width (20MHz) than either ZigBee or Bluetooth, making it more resistant to interference and offering higher transmission speeds, but requires considerably more power and is far more complex, making it significantly more difficult to experiment with.

One recent experiment used a laptop with a WiFi chipset providing detailed information about the phase and magnitude of a received signal across multiple subcarrier frequencies. They managed to reliably identify the location of this laptop within an area by first collecting such "channel frequency response" data at a large number of locations, modelling it as a Gaussian mixture distribution, and then applying a clustering algorithm to find cluster "fingerprints" for the data at each location [25]. They claim that this method provides significantly better results than an RSSI-based approach.

Some recent Atheros 802.11n WiFi chipsets expose simpler "spectral analysis" information, providing information about received power, split into 56 "FFT bins" for each 20MHz channel, allowing for a relatively fine-grained view of the signal power present over the entire 2.4GHz spectrum (by sweeping through each 20MHz channel in turn).

We obtained a WiFi device which made use of such a chipset, verified the accuracy of the spectral analysis by using the sine wave functionality of both our ZigBee nodes and a Bluetooth device, and then used it to monitor radio activity during our other experiments and exclude occupied portions of the spectrum (largely due to WiFi usage, but we also encountered other interference) from the frequencies we were using.

At the time this thesis was being written, researchers at UCSB had just announced [20] their work in imaging an environment hidden behind thick walls with the use of only two WiFi devices (one transmitting and one receiving) mounted on moving robots, successfully reconstructing the objects inside with a high level of accuracy (2cm resolution).

3 Modelling

In this chapter, we will investigate the models and algorithms which we intend to use to perform radio tomographic imaging (RTI). Our goal here is to reconstruct an image of the objects inside the space we are "scanning", using signal strength information from the links of a sensor network placed in or around the space; specifically, we will attempt to image a 2D cross-section/slice of these objects.

After first reviewing the nature of the problem which we intend to solve, we will consider both how to model it, and how to use this model to produce the desired image.

3.1 Introduction

The assumption behind the models we will be using for RTI is that, as radio signals propagate, they reduce in power due to path loss; in particular, we are interested in absorption losses, which are caused by the signal passing through materials (such as air, or walls, or people). If we transmit a radio signal from one position and consider a receiver in another position, then the received signal gives us information about the materials in the path between the transmitter and the receiver; for example, the signal strength will be lower if a person is standing in the way of the signal.

The real-world behaviour of radio waves often doesn't match this idealistic model, even when we ignore diffraction and refraction effects and consider them primarily as direct-path waves. For example, in the situation pictured in Figure 3.1, there is an object blocking the direct line-of-sight path to one node — resulting in a reduced-strength signal — but multiple high-strength signals still reach it by reflection. This is called *multipath*

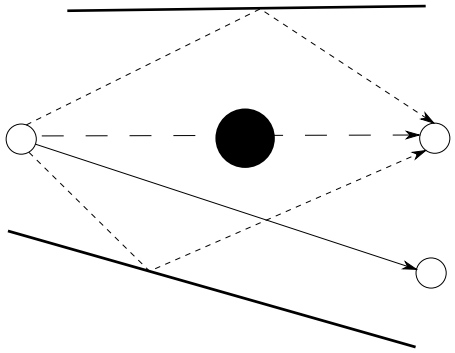


Figure 3.1: Direct and indirect signal paths between sensor nodes.

propagation, and means that the total signal strength received by this node is sometimes (especially in reflection-rich environments such as small indoor rooms) higher than that for the other node, despite the fact that the line-of-sight path to the other node is clear (resulting in a higher-strength signal on the direct path).

The reflections of multipath propagation are heavily influenced by the location of the surroundings, and so small differences in the position of radio antennas can lead to an extremely large difference in the results. A recent paper discusses prototype hardware which can change its position by small amounts so as to dynamically select the best position for minimising these multipath effects [9].

We noticed during earlier experiments (see Section 5.4 of [28]) that, when trying to image humans in motion in an indoor environment, these problems meant that the variance of the signal strength was often a far better indicator of such motion than the signal strength itself. It has been shown [31] that it is possible to use the variance to easily track motion in situations where tomographic imaging using the signal strength appears extremely difficult to apply (such as when doing it through walls).

These factors which influence the received signal strength – such as the transmission power, static shadowing loss, multipath fading gain and noise – are discussed further in [30].

3.2 The problem

If we consider the problem as time-invariant, we can begin by summarising our input data as a single value for each link between nodes. For our work, this will either be the signal strength of the link, the signal quality, or the variance of one of those two. We will call this data y, a vector with one entry per link.

The output of our model should be an image. This applies even if we restrict the problem (as some of the existing work does) to only imaging the movement of objects.

A simple model for such an image involves square 2D pixels (or, potentially, 3D voxels), spaced identically in a 2D grid covering the entire space. This has a single parameter, the width of a pixel, which we will call ϕ. We can model these pixels as real numbers between 0 and 1, where 0 represents a pixel which is completely empty, and 1 represents a pixel which contains a heavily-attenuating object.

The output can then be described as an image of resolution $m \times n$. We can view it in our model simply as a vector containing mn elements (row-by-row), and refer to that vector as x.

The traditional model for X-ray tomographic image reconstruction involves the Radon transform, which is based on line integrals of a function; in this case, these represent the attenuation of beams through objects from various different positions and angles, and the function represents the density of the object. An algorithm such as filtered backprojection

[14] can then be used to obtain the desired image. However, even in the far simpler (due to the reduced amounts of absorption/scattering) situation of X-ray tomography, problems such as limited data can make reconstruction using such techniques impossible [10], and the non-line-of-sight situations present in radio tomography mean that this is simply not an effective approach.

The most obvious alternative technique, which has been shown to work well for RTI as well as more general tomographic problems, is to approach the problem algebraically (or statistically). The relatively small resolutions and links used in RTI mean that an approach using iterative algebraic reconstruction techniques is unnecessary.

We proceed by framing the problem as a linear system. We use a *weight matrix* (henceforth W), which contains "weights" describing how the contents of each pixel influence each link, and a vector b representing all other contributions to the link values (in particular, the noise). This b might contain both linear and non-linear components, but we ignore any dependency it might have on x. Our model can now be expressed as a simple equation:

$$y = Wx + b$$

We can then "solve" this linear problem by finding a solution to the so-called *inverse problem*. Specifically, we will attempt to find a vector x (the pixels of our image) which conforms to the known value of y (the vector containing our input data), under the assumption that the relationship between y and x is accurately described by W and b.

In this context, the inverse problem will be *ill-posed*; there will be no unique solution x, because the problem is underdetermined (there are far fewer links than there are pixels, and so the solution will be ambiguous), and any solution will be unstable, due to the underlying nature of the problem. In order to find a unique, stable solution, we will have to impose other requirements, a process known as *stabilization* or *regularization*.

We will proceed on the assumption that the noise vector b is unknown and we will not incorporate it into our model further; while it is possible to incorporate an estimate of the underlying noise (as was done in [30]), we will instead attempt to cope with the noise both by suppressing it outside the model using calibration (which we will discuss later), and by using regularization techniques which attempt to minimise the effect it has on the results.

3.3 Weighting

The weight matrix discussed above is best thought of as a collection of smaller matrices, one for each link, where each matrix consists of a set of weighting values, one for each pixel in the image.

If we assume that the strength of each link is primarily determined by objects lying on the direct path between the two nodes in question, then we can use a per-pixel weight of

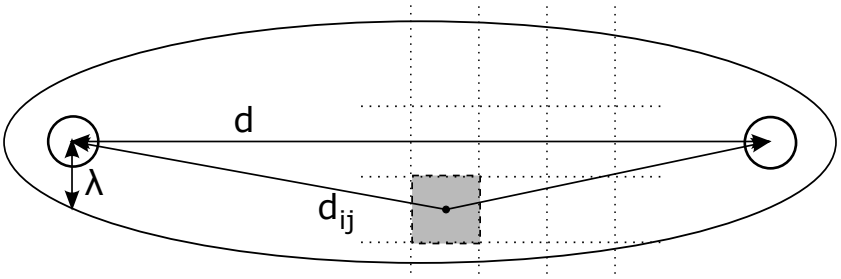


Figure 3.2: Ellipse model from [30] used for weighting links.

0 for pixels which are not on this direct line, or 1 otherwise.

An obvious approach would be to divide the space being imaged into a pixel-spaced grid and place an appropriate attenuating object at each pixel position, then measure the change in the link strength for each result and use the resulting per-pixel weights to construct the matrices. This would help account for factors such as environmental multipath effects (which are particularly important in indoor settings), but is time-consuming to construct and requires that the sensor network and the environment remain relatively static, since (as discussed) small movements can lead to huge changes in these effects.

In our experiments, we primarily used the ellipse model from [30]; instead of considering only the direct lines between nodes, this uses an ellipse with endpoints at the nodes, and a width specified by a parameter λ. The disruption of radio waves by obstacles is most significant inside the first Fresnel zone (an ellipsoid covering the space between the endpoints of a radio wave transmission), so this is a more reasonable way to model the situation. It also avoids the problem (inevitable unless we use very low resolutions or very dense networks) where many pixels may not be on the exact direct line between two nodes.

It also uses a per-pixel weight of $1/\sqrt{d}$ for a link of length d, rather than 1, thus reducing the significance of longer links in favour of shorter ones (which have been influenced by fewer other pixels in total). This is discussed further in the context of statistical modelling of the radio links by [22].

In this ellipse model, pictured in Figure 3.2, the weight w_{ij} for a link i and pixel j is given by

$$w_{ij} = \begin{cases} \frac{1}{\sqrt{d}} & \text{if } d_{ij} < d + \lambda \\ 0 & \text{otherwise} \end{cases}$$

where d is the length of the link, and d_{ij} is the sum of both distances from the center of the pixel to the nodes at the end of the link.

3.4 Regularization

After discussing the necessity of regularization by considering the issues with a simple least squares solution, we will discuss three standard techniques from the various possibilities for regularization of tomographic reconstructions: truncated SVD, Tikhonov regularization and Total Variation. Their application to RTI was first discussed in [33]. We will discuss them mostly on a practical level here; the theoretical background and derivation is explained in detail in standard texts such as [21].

Truncated SVD

The simplest approach to solving our inverse problem is to ignore the "noise" vector b and to use the method of least squares, without any regularization. Specifically, we could calculate the solution x which minimises the squared differences $||y - Wx||^2$ (by $||\cdot||$ we refer to the L^2 norm). If we write the singular value decomposition (SVD) of W as UDV^T, and construct D^+ by taking the reciprocal of each element (i.e., the singular values) of D, then this solution is $x = Ay$, where $A = VD^+U^T$ is called the *pseudoinverse* of W (often written as W^+).

Since A does not depend on the values of y at all, it can be pre-calculated once for a given weighting matrix, and then the per-sweep reconstruction can be done extremely quickly by just multiplying A and the relevant y together.

This does not work well in practice due to the ill-posed nature of our problem, leading to large amounts of instability because the smallest singular value differs dramatically from the largest one, so small changes in the input data (such as noise) can result in disproportionate changes to the result. Truncated SVD replaces the elements of D^+ which are smaller than a given threshold with zero, thus stabilising the solution, while remaining fast to calculate.

However, the resulting reconstructed images are far from ideal; the components of the data corresponding to these singular values are entirely left out of the calculation, and the resulting images remain noisy and unclear.

Tikhonov regularization

Generalized Tikhonov regularization stabilises the least squares method by incorporating a function $f(x)$ which constrains the solution of the problem in some way, and minimising the sum of the squared differences plus the square of this function: $||y - Wx||^2 + \alpha||f(x)||^2$, where α is the *regularization parameter* which specifies the relative influence of the constraint.

For example, one common application is to use $f(x) = x$, which minimises the norm of the solution, which in our situation would constrain the solutions to be relatively empty

(full of zeros).

Another standard approach is to assume that our desired image is "smooth", in the sense that it has as few differences between adjacent pixels as possible. We can do this using a *difference matrix* which calculates a numeric approximation to the derivative of the image in a certain direction by using central differences, and this is the approach taken by [33], which uses $f(x) = Dx = (D_x + D_y)x$ where $D = D_x + D_y$ is the sum of, respectively, the horizontal and vertical difference matrices.

Tikhonov regularization complicates the problem compared to truncated SVD by replacing the singular value threshold with the parameter α. An appropriate value can be established experimentally; [21] discusses some methods for evaluating candidate values.

As long as the function $f(x)$ is selected such that it can be expressed as a matrix multiplication (such as in the above two examples), the matrix A can still be pre-calculated and the reconstruction can then be performed with a multiplication. We do this simply by including the regularization component in our linear equation, "stacking" the regularization below the weights matrix; for example, using $f(x) = Dx$:

$$\begin{bmatrix} W \\ D \end{bmatrix} x = \begin{bmatrix} y \\ 0 \end{bmatrix}$$

Total Variation

A disadvantage of the smoothing approach with Tikhonov regularization is that the resulting images lose their sharp edges, due to the squaring of the derivative function. This can be mitigated by minimising the sum of the squared differences plus a function, i.e. $||y - Wx||^2 + \alpha f(x)$, where $f(x) = |Dx|$ is the absolute value (1-norm) of the derivative function rather then the square. This is known as Total Variation regularization.

However, since the regularization term is no longer squared, it can no longer be solved using a least squares approach, and so the reconstruction becomes far more computationally expensive. In order to minimise this sum, one possibility is to use an optimisation algorithm, and such algorithms must be provided with an approximation to the derivative of f, but the derivative of $|Dx| = \sqrt{(Dx)^2}$ is not defined at $x = 0$.

This problem can be resolved by replacing the $|Dx|$ term with one which is continuously differentiable around $x = 0$, and the obvious choice is $\sqrt{(Dx)^2 + \beta}$ for some value of β. The value of β must be chosen so that it is sufficiently small so as to avoid adding a smoothing effect, but not so small that it makes convergence problematic. In practice, a wide range of values appear suitable.

4 Implementation

In order to further investigate and make use of the models and reconstruction methods discussed in the last chapter, we decided to write software which would perform the image reconstruction and provide visualization of the source data as well as the resulting image.

We wrote this software in Python, using PyQt for the user interface and SciPy [16] for the numerical processing. This allowed us to easily modify and/or replace the models, algorithms and data sources involved. We have successfully used it on several different platforms (Windows, Linux and Mac OS X), and we intend to make it freely available for other work.

The visualization user interface, which can be seen in Figure 4.1, is split into two halves.

On the left, a visualization of the current state of the sensor network is shown, with nodes positioned in the correct locations relative to one another, and the strength of links illustrated using colour (red and blue indicate high and low strength respectively).

On the right, the result of the current reconstruction is shown. We chose to display both the sensor network state and the reconstruction at the same time because we found that it made it far easier to understand the cause for issues with the reconstructed image,

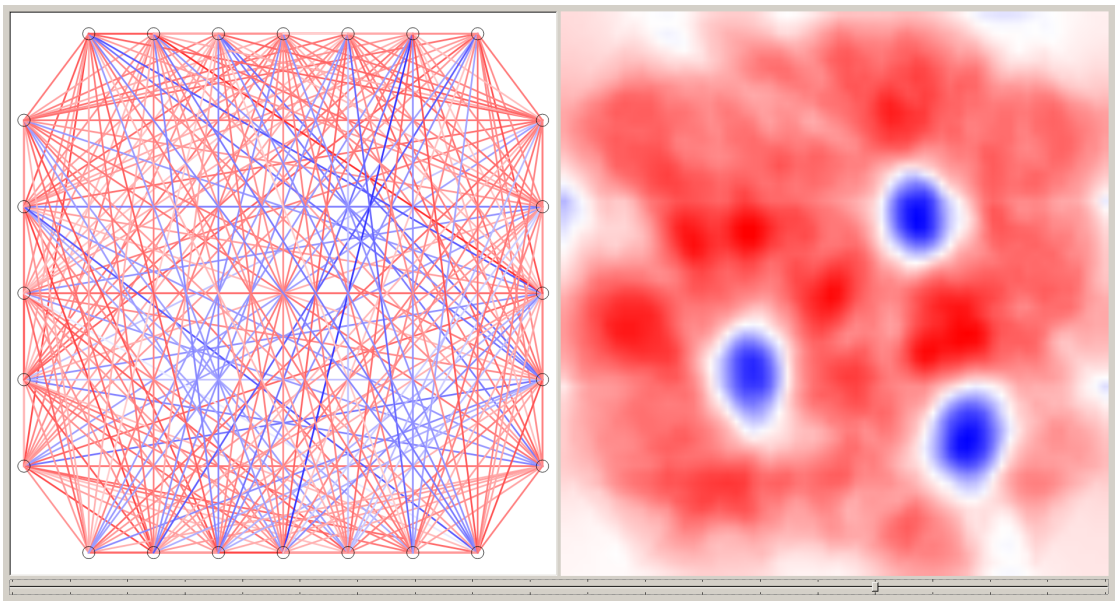


Figure 4.1: Visualization software viewing a frame from a simulation run.

especially when using the software on real-time data.

The software splits the incoming data, both static and real-time, into "sweeps". One sweep represents each timespan in which all sensor nodes have (attempted to) transmit a data packet. Each such packet contains, for each other node, the signal strengths and link quality information which the node collected during the last packet received from that other node. The transmission is done in-order, according to manually-specified node numbering.

When the software is being used to process previously-collected data (it can read data from CSV files, MATLAB .mat files and raw dumps from the ZigBee nodes), a slider at the bottom allows selection of the data sweep to display and use as a source for the reconstruction algorithm.

The reconstruction algorithm to be used and some of the parameters can also be selected from within the GUI, by using keyboard input.

The data from the ZigBee listener node (recieved via a USB stick running custom firmware) is collected using a different Python script, and then transmitted using a TCP socket to the reconstruction software. This allowed us to collect the data using a more portable device (in our experiments, a small netbook), which we could place closer to the sensor network, and then process and display it elsewhere in real-time.

This design also simplified collection of data for later experimentation, since the collection script has no non-standard dependencies and so can be run on almost any computer. The script also stores the timestamps for all received packets, which can be used to retransmit the contents of a previously collected data file at the same time intervals.

The user interface displays the value of a link between two nodes as the average of the strength of both directions of the link, if both these values are available for the sweep being displayed. Section 5.2.1 of [28] concludes that, in static situations, these values are generally identical.

For most of our experimentation, we assumed the links between sensors were indeed symmetric, which allowed us to also use one-directional or averaged signal strengths for the input to the reconstruction algorithm. Since this meant that our weighting model only needed to consider half of the links (i.e., the weight matrix was upper-triangular), this significantly reduced the computational complexity of the problem.

The function for each reconstruction algorithm was provided with the input signal strengths, the output image resolution, the most recently reconstructed image (or an empty image, if none), the reconstruction parameters, the weight matrix and both vertical and horizontal (central) difference matrices. For the Truncated SVD and Tikhonov algorithms, the relevant pre-computed matrix was also available. Sparse matrices were used where possible. For the Total Variation reconstruction, we made use of SciPy's L-BFGS-B optimisation algorithm [19].

Some of the existing literature suggests that performance problems might be a significant

problem for the use of the Total Variation reconstruction, and so we also investigated the possibility of making use of a GPU for the Total Variation optimisation work, using a library such as CUDA L-BFGS [29]. However, after appropriate vectorisation of the code, we didn't encounter any significant performance problems during our (relatively small-scale) experiments which couldn't be resolved without the added complication (and hardware demands) of GPU computation. We provide some figures for execution time in the next section.

The latest version of the software discussed here can be obtained from `https://github.com/fuzzie/radio-tomography`.

5 Simulations

We decided to first evaluate the model and reconstruction algorithms discussed in the previous chapter with data from simulated test sensor networks, both to simplify the development process (since our simulated data is likely to be less problematic than that from real-world networks, as discussed), and to allow us to easily evaluate the likely results of varying sensor network configurations without the time-consuming (and possibly expensive) requirement of actually setting up such networks.

We generated data for simulations by first creating high-resolution (512×512) grayscale images containing the obstacles/items we wanted to image, with the objects (primarily circles) in white, on a black background. We applied a blur filter and added some random Gaussian noise.

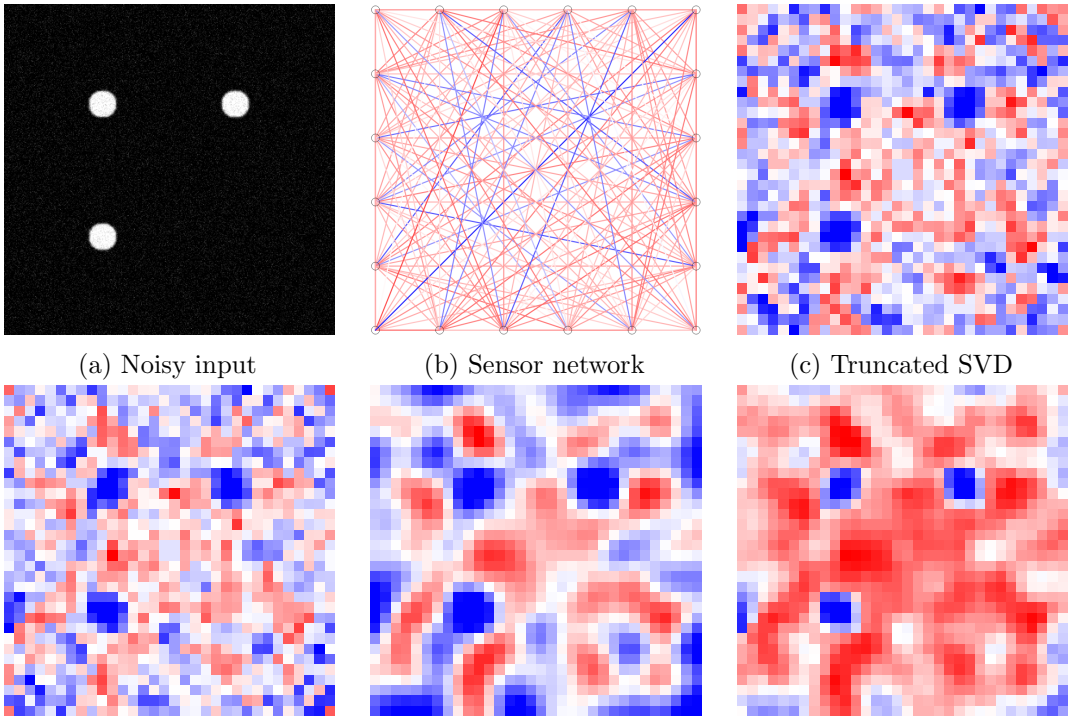
We then calculated a value for every link in the provided sensor network by applying a mask containing the direct line (calculated using Bresenham's line algorithm, which should suffice since the blur filter extends the edges of an object by several pixels in any case) between the nodes in question, and then using the highest pixel value in the resulting image (i.e., the most attenuating pixel).

We chose to take this approach, rather than simply generating data by directly applying the weight matrix from the model we will be using when doing the reconstruction, because it can more easily be extended to account for any aspects of the physical model which we might wish to consider in the future (such as multipath rays and increased attenuation of denser objects). It also inherently avoids a so-called "inverse crime" [19], where misleadingly positive results are obtained due to the use of an identical model for both the simulation and the reconstruction.

We simulated movement (between frames) simply by using different images (with the objects in different positions) for each sweep, but this system could easily be extended in the future to more accurately simulate movement by using a different image for every timeslice within a sweep.

As well as using them to evaluate our algorithms and their parameters, we also used simulations to explore the effect which differing sensor network layouts have on the precision of the reconstruction. While these results are obviously restricted in their applicability to real-world situations due to their dependance on the accuracy of the model being used, they provide some insight into which layouts would be productive to consider further.

To evaluate the accuracy of the reconstructed images, we have to compare them to the original image. When attempting to perform localization of objects (i.e., determine their



(a) Noisy input (b) Sensor network (c) Truncated SVD

(d) Tikhonov (norm) (e) Tikhonov (gradient) (f) Total Variation

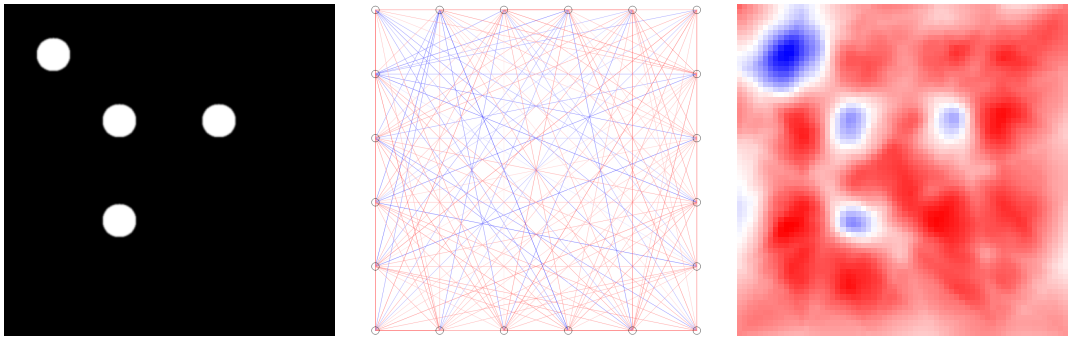
Figure 5.1: Images of reconstructions of a noisy simulation. Colouring runs from red (empty), through white, to blue (attenuating).

position), we can simply compare the calculated positions with the true values. For the evaluation of imaging in general, we can evaluate the error of every pixel in the image by subtracting it from the original data. We can then use the MSE (mean square error) as a statistic.

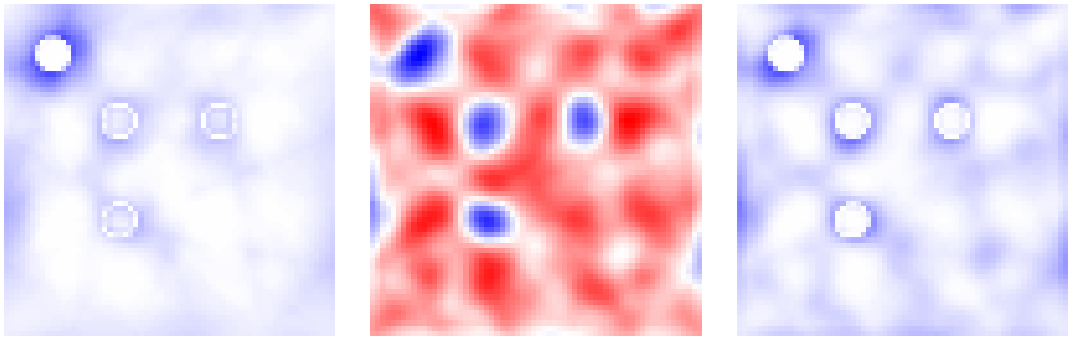
Assuming our goal is limited to producing a black-and-white image (i.e., that our RTI will not manage to differentiate different attenuation levels of objects), this requires we apply some threshold level.

When the data is intended for viewing by people rather than post-processing for a purpose such as localization, then we found it preferable to display the reconstructed image without applying a threshold. After being used to ensure the reconstructions were of acceptable quality, we found that lower error measures did not necessarily correspond to what were considered "better" results by observers, and so fine-grained adjustment of the parameters had to be done manually.

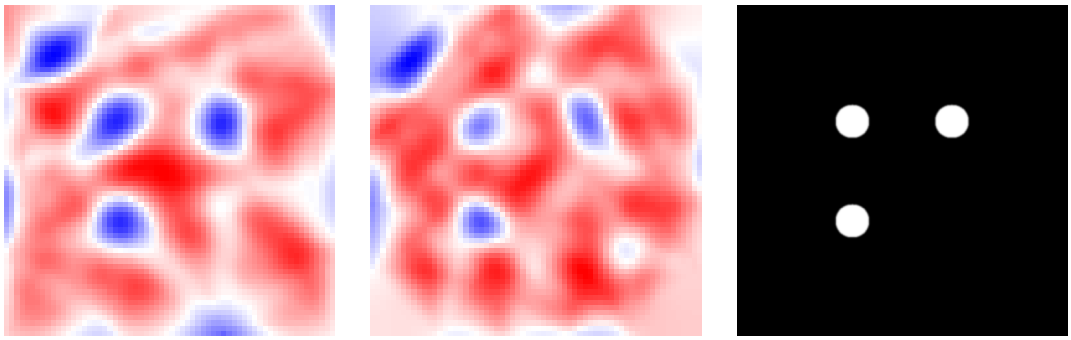
Some example reconstructions of an example static image using different algorithms can be seen in Figure 5.1, using a resolution of 32×32 and a sensor network of 20 nodes (4 on each side, plus 1 per corner). We adjusted the other parameters manually to provide visually-pleasing results for each algorithm.



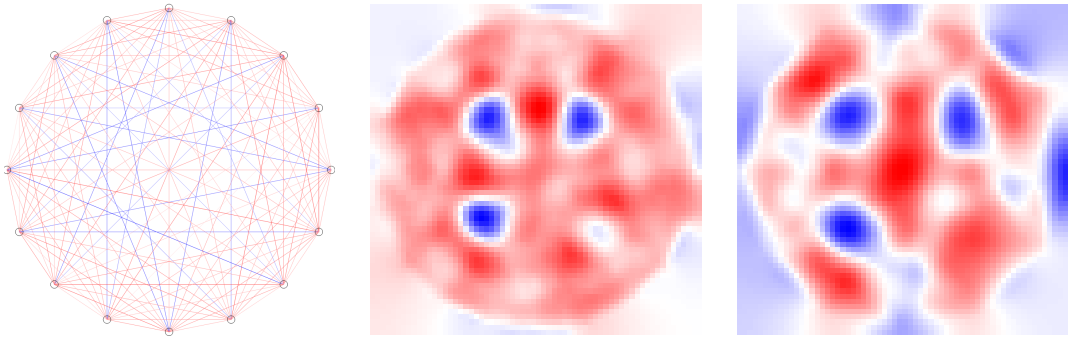
(a) Input image (b) 20-node square network (c) $\lambda = 0.01$: Reconstruction



(d) $\lambda = 0.01$: Error (e) $\lambda = 0.1$: Reconstruction (f) $\lambda = 0.1$: Error

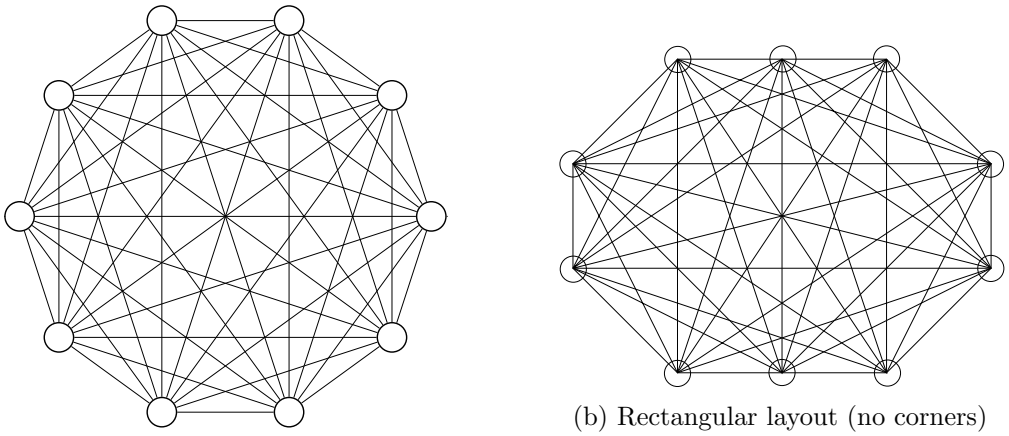


(g) $\lambda = 0.1$: 16 nodes (h) Without corners (i) Input image 2



(j) 16-node circle network (k) 16-node circle (l) 12-node circle

Figure 5.2: Images from a simulation run, using Total Variation reconstruction.



(a) Circular layout

(b) Rectangular layout (no corners)

Figure 5.3: Layouts for 10-node sensor networks.

Since we obtained better results from Total Variation both in simulations and (as we discuss later) when using real data, we focused on TV for our further work. Figure 5.2 shows some example reconstructed 64×64 images from a single simulated frame, using the Total Variation algorithm. The area being imaged contained four simulated circular objects (shown in (a)) with a diameter of $\frac{1}{10}$ of the area.

Red in our reconstructed images represent low values (empty space), while blue represents heavily-attenuating objects. This was chosen to correspond with the colour scheme used in our visualization of the sensor network links. The error images show the squared error between the reconstructions and the original images; darker shades of blue correspond to increased error values for a pixel.

Images (c) to (f) were generated using a simulated network of 20 nodes (depicted in image (b), 4 on each side plus 1 in each corner), using two different values of λ (ellipse size). For the rest of the images, we used $\lambda = 0.1$. In image (g), we removed 1 node from each side of the network (leaving 16 nodes), resulting in a more distorted image.

Image (h) shows the result of removing 1 node from each corner instead of each side; it is clear that, unsurprisingly, this results in far worse results for the simulated object near the corner of the network.

As can be seen, different values of λ significantly influence the reconstruction; the size of the circles in image (c), with a smaller value, are considerably closer to the input image than those of (f). However, excessively small values of λ cause reconstruction problems when some pixels are no longer covered by sufficient or any ellipses.

Images (k) and (l) perform the simulation without an object near the corner (see image (h)), using a circular network layout of 16 (image (j)) and 12 nodes respectively.

A visual comparison of circular and rectangular layouts is shown in Figure 5.3. Circular

layouts provide more evenly spaced coverage, and avoid the problem of extraneous overlapping links on the edges of a rectangular network. We'd expect them to provide better results, and indeed, our experiments above show that the circular network layouts seem more promising. Section 5.5 of [28] suggests that real-world sensor nodes also produce more useful results when not placed in corners. However, in real-world situations, we usually want to image a rectangular area (such as a room), and being forced to exclude the area outside the network is not a promising solution.

Another interesting result was that, for relatively slowly-moving objects (for example, representing humans walking through an area), a single iteration of L-BFGS-B (see Section 4) per sweep for Total Variation reconstruction produced acceptable images when the optimisation algorithm was provided with the image from a previous frame as the starting point, which could be done in far less time than would be needed for the algorithm to reach a reasonable level of convergence. Since the previous image in such a situation would be relatively similar to the new image, this is not surprising, and makes it possible to use this algorithm for reconstruction of images from larger sensor networks in real-time.

Figure 5.4 provides maximum execution times for reconstruction using an AMD Athlon II X2 250, using the simulation from Figure 5.2(a), with the objects moving over time. As would be expected, execution time increased when using larger values of of λ (i.e., where each link covered more pixels) and when using higher output image resolutions.

The average execution times did not differ greatly from the maximum times provided; for example, Total Variance in the 20-node environment took a maximum of 0.127s and a minimum of 0.094s to converge for this example. In any case, the maximum times are a better approximation of the worst-case situation which must be considered when deciding whether real-time reconstruction is possible.

Algorithm	Resolution	$n = 20$	$n = 20$ ($\lambda = 0.1$)	$n = 28$	$n = 36$
TV (one iteration)	32×32	0.016	0.021	0.021	0.029
	48×48	0.057	0.081	0.080	0.117
TV (convergence)	32×32	0.127	0.246	0.212	0.289
	48×48	0.535	0.941	0.961	1.142
Tikhonov (norm)	32×32	0.006	0.008	0.011	0.017
	48×48	0.011	0.018	0.026	0.054

Figure 5.4: Maximum execution time (in seconds) for reconstruction of frames in the four-object simulation, using square sensor networks. n is the total node count. $\lambda = 0.05$ where not otherwise specified.

6 Experiments

Since our simulated sensor data fails to accurately reflect many aspects of real-world sensor networks, it was also important for us to attempt to reconstruct images using data from such networks.

We first discuss the results of applying our proposed reconstruction to existing data, and then attempt to do the same with data obtained from our own experiments.

6.1 Object imaging

Since the wavelength of 2.4GHz signals is about 12.5cm, and (even in ideal circumstances) we might expect to have problems with resolving any details smaller than half that (around 6cm) due to diffraction, our simple version of radio tomography at these wavelengths is not a promising technique for imaging of relatively small objects. We decided to conduct some experiments on this kind of smaller scale, to investigate whether it is possible to produce some useful results in any case. We placed a 35cm-wide rotating turntable (marked at 8 equally-spaced points, allowing easy rotation by multiples of 45°) in the middle of a table in an open environment, between several ZigBee nodes (transmitters) on one side, and a receiver (either another ZigBee node, or an antenna for our RTL-SDR device) on the other side. The transmitter nodes were mounted side-by-side on a rail, and by moving the nodes along the rail, and rotating the turntable, we can simulate a fan beam being transmitted from many transmitters, at differing angles.

We then placed several different objects on the turntable, such as a metallic pan (with a diameter of 25cm) and a large oval-shaped plastic jug (20cm across at the widest point) containing water; this can be seen in Figure 6.1.

Our estimate of the attenuation of the signal (as measured by the RSSI of the links as measured by the ZigBee node) varied unpredictably, often increasing dramatically when a link passed through the centre of the objects, but also significantly differing after very small movements of the nodes along the rail. This made it impossible to perform any image reconstruction from the data gathered during these experiments.

Lower RSSI values did appear correlated with the presence of the edge of an object when placed in the direct path between a transmitter and the receiver, perhaps due to interference within the innermost Fresnel zone, but these results were unreliable. Figure 6.2 shows the data from one node in the experiment. We placed the edge of a jug such that it would always be in the path of the node in question when it was at position

Figure 6.1: The turntable and rail used for attempted object imaging, with a jug of water.

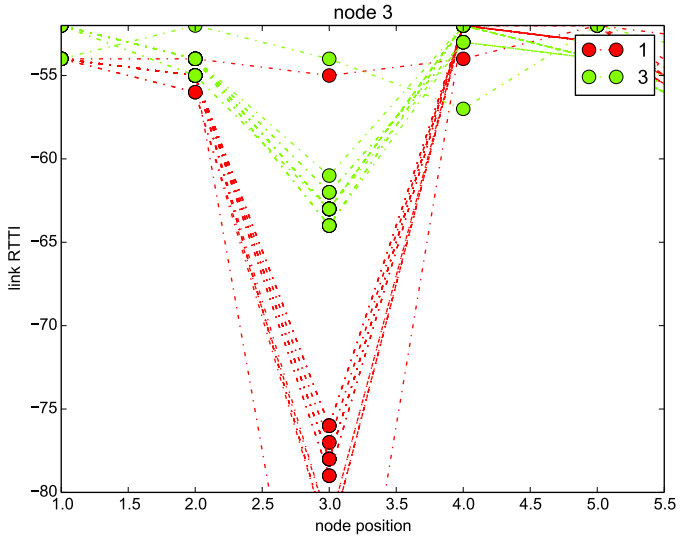


Figure 6.2: Some results from the jug experiment (one point per 500ms of data), plotting node positions (with about 2.5cm between adjacent positions) against average measured signal strength.

3. Rotation 1 (red) shows the averaged results when the edge in question was the longer edge of the jug, while rotation 3 (green) shows values with the shorter edge.

We repreated the experiment with our RTL-SDR device, and discovered that the link quality metric (the correlation rate measured by our modified RTL-SDR software) dropped dramatically when the edge of the jug was in the direct path for a radio link. One potential alternative approach for radio tomographic imaging could be to attempt to use these variations to detect the edges of objects, rather than the objects themselves.

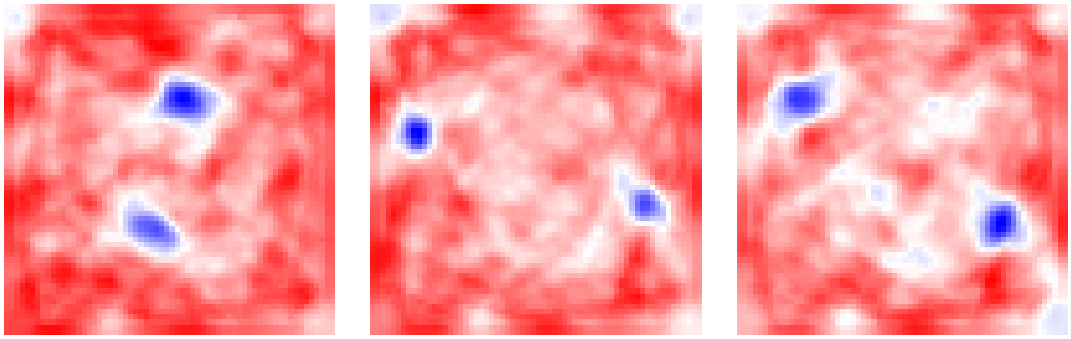


Figure 6.3: Some reconstructions using the data from the Utah outside network.

6.2 Utah outdoor network

Joey Wilson and Neal Patwari, from the University of Utah, published the data from one of their earlier radio tomography experiments [32]. It consists of sets of RSSI measurements taken from their 28-node sensor network, laid out as a square around an area approximately 21×21 feet.

Their sensor network was placed in an outside area, which minimised the problems discussed earlier caused by multipath propagation. The area to be imaged contained two trees, as well as one or two people either standing still or moving (depending on the dataset).

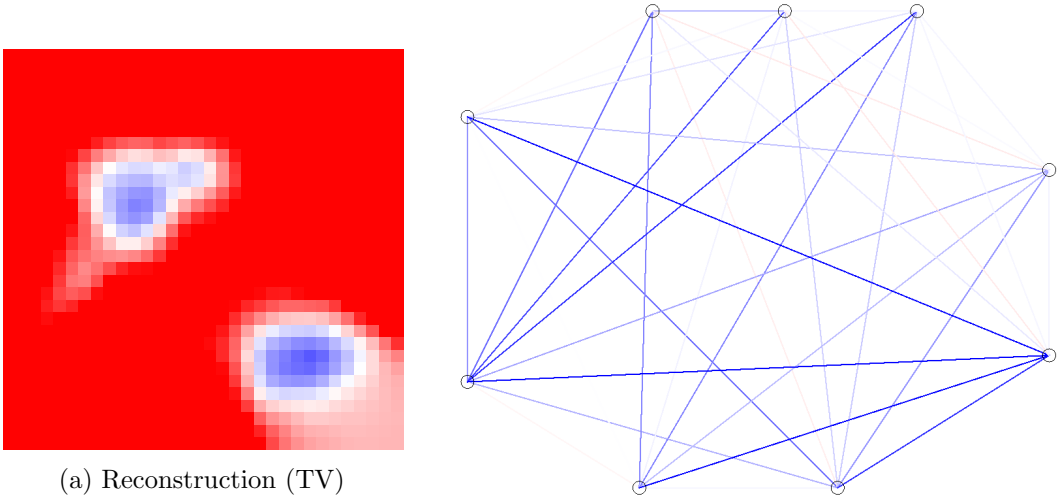
These measurements were provided in CSV files, so we modified our framework so it could read the data from these files directly, and also specified the (provided) positions of each sensor node.

We improved the results by applying a simple calibration method along the lines of that described in [30]; one of the sets of measurements provided was recorded when the area was empty, and so for every link, we simply averaged the link strengths provided in that empty dataset, and subtracted that per-link average from all later measurements. In terms of our model, this corresponds to attempting to account for some of the non-weighted contribution (b from our equation $y = Wx + b$).

The results of applying the reconstruction algorithms to this data were very promising; some sample reconstructions are shown in Figure 6.3.

The single dataset which we did not manage to reconstruct well involves a human running at sprinting speed through the area; since the provided data only includes a few sweeps of the network per second, our per-sweep approach to the reconstruction inevitably causes problems when there is a large amount of movement within a sweep.

Figure 6.4: The area of a living room used for experiments.



(a) Reconstruction (TV)

(b) Network state

Figure 6.5: A sweep of the 9-node living room network with two people standing in it.

6.3 Small indoor network

We built a sensor network around a rectangular area in the corner of a living room about 2m × 2m in size (see Figure 6.4), with 2 nodes on 3 sides of a rectangle, and 3 nodes on the remaining side. The nodes were placed on sofas and chairs, spaced equally on each side (but not in corners).

The reconstructions were successful, but the fluctuating noise in the background of the images was distracting. We modified the visualization software to colour the image using a more limited range of values, which gave considerably more visually-pleasing results.

Despite the small number of nodes, the reconstructed imagery from the network allowed

Figure 6.6: Part of our indoor sensor network at CWI.

us to reliably determine the location of two people simultaneously present in the network, with an accuracy of around 50cm. You can see an example such reconstruction in Figure 6.5, using an image resolution of 32×32. The accuracy was noticeably better for objects closer to the side with more nodes.

6.4 Larger indoor network

We conducted experiments using an indoor sensor network at CWI in Amsterdam, in a rectangular room approximately 5m $\times$ 4.5m, pictured in Figure 6.6. The network was comprised of ZigBee sensor nodes, placed on stands at a height of about 1m from the ground. There were full-height panes of glass along the length of two sides of the room. A table with a computer and monitor, for viewing the results, was placed in one corner.

Section 5 of [28] discusses the specifics of the hardware and firmware used, as well as some details (such as the optimal antenna direction).

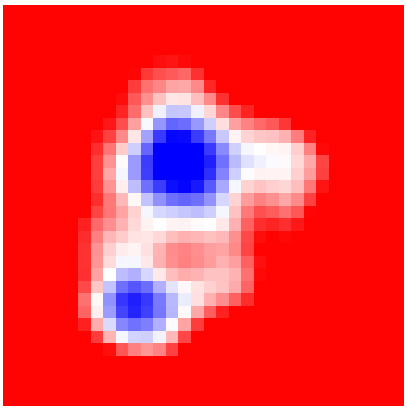
After our promising results with the Utah data and our small indoor network, we were surprised to find that we had great difficulty obtaining reasonable results of any kind in this indoor environment. The 2.4GHz band was in active use, interfering with our experiments, although we found that ZigBee channel 26 (which does not overlap WiFi channels) was generally far quieter.

The placement of the nodes also caused us some difficulty; links which were close to the table were clearly noisier than those elsewhere, and links passing through the glass panes (i.e., when nodes were placed outside the room on the other side of those panes) also

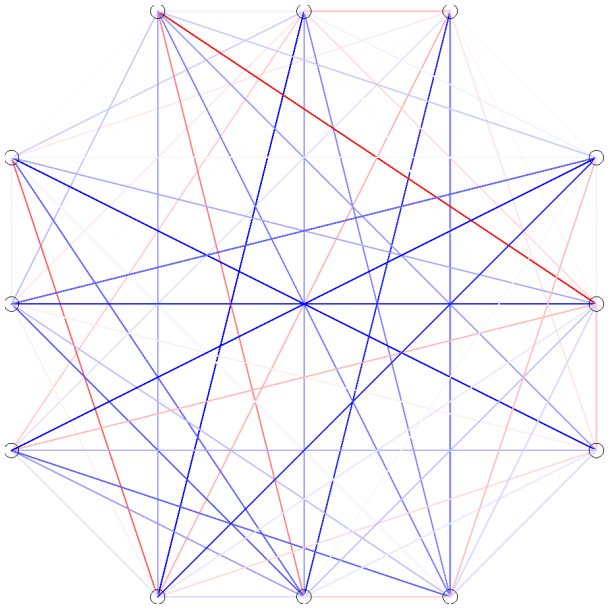
suffered considerably more disruption. There were also initially metal blinds covering one set of glass panes, which disrupted the signals.

Most importantly, however, the multipath-rich nature of this indoor environment meant that we had to adjust our model to make use of the deviation of links from the calibrated baseline, since the signal strength of a link would often increase when an attenuating object or person was present in the direct line between nodes; we obtained good results by simply adjusting the input signal strengths to the reconstruction algorithm to be the (absolute) difference between the signal strength and the initial baseline (from the calibration), rather than the signal strength itself. We also modified the code to use a significantly longer calibration period (around 10 seconds).

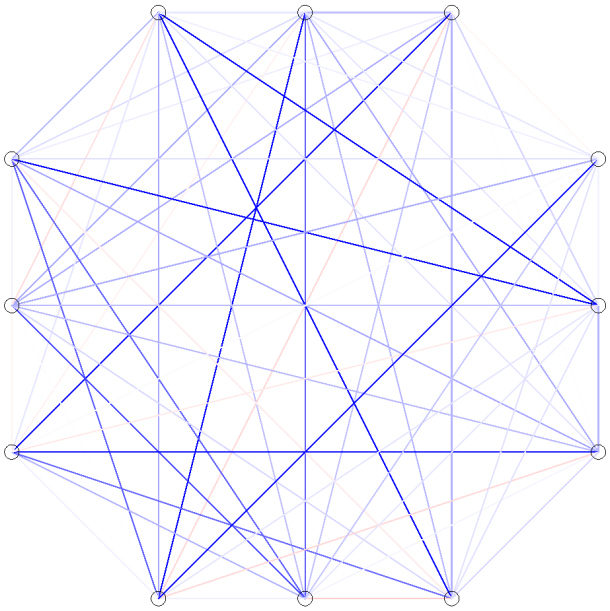
Figure 6.7 shows reconstruction results from a 12-node network (3 nodes on each side, none in the corners) in this environment, with two people present, again with a resolution of 32×32. Image (b) shows the link strengths before the variation is considered, while image (c) illustrates the values provided to the reconstruction.



(a) Reconstruction (TV)



(b) Network state



(c) Adjusted state

Figure 6.7: A sweep of the 12-node CWI network with two people standing in it.

7 Conclusion

We succeeded in developing our own software for radio tomographic imaging, which provides acceptable results with Total Variation reconstruction running in real-time, using data from a sensor network placed in indoor environments.

Our simulations were helpful but failed to reflect real-world situations in various ways; using a radio wave model or performing raytracing to simulate reflections are two example methods which might be used to improve them, and enable more accurate analysis of potential sensor network layouts.

As mentioned in Section 6.2, fast-moving objects in larger sensor networks can cause problems due to our implicit assumption that sweeps are instantaneous. Sweep times also become more significant when performing sweeps on multiple frequencies; it has been shown [17] that accuracy can be improved by doing this.

Another area to explore would be whether to allow the area around the edge of our network to be considered at all in the reconstruction; [22] suggests that the area around the sensors shouldn't be included, and so the short links are uninformative and should be excluded. This would also help resolve the problems which can be seen around the edges of reconstructions of circular networks, where large amounts of the image represent areas which lie entirely outside the sensor network.

The long-term behaviour of networks is also something we have not discussed; rather than requiring that calibration be performed beforehand on an empty area, as we have done here, it has also been shown to be possible to instead perform calibration while the network may be occupied [8].

We hope to improve the firmware of the sensor nodes to allow for dynamic reconfiguration of parameters such as the frequencies being used, the transmission signal strength and the correlation threshold, which could then be adjusted as needed during real-time reconstruction. Similarly, investigating the stability of the RSSI over the whole length of packets and making further use of software-defined radio hardware may be interesting directions to consider.

A significant limitation of our approach to reconstruction is the modelling of the reconstruction on the basis of a linear system. Some other work, such as the medical imaging applications discussed in the introduction, make use of some form of wave propogation modelling to obtain more accurate/higher-resolution results.

8 Bibliography

[1] IEEE standard for local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks. *IEEE 802.15.4-2011 (Revision of IEEE 802.15.4-2006)*, 2011.

[2] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *ISO/IEC/IEEE 8802-11:2012*, 2012.

[3] Texas Instruments CC253x/CC254x user guide (rev. D), Mar. 2013.

[4] T. Aaberge. *Low Complexity Antenna Diversity For IEEE 802.15.4 2.4 GHz PHY*. PhD thesis, Norwegian University of Science and Technology, 2009.

[5] F. Adib, Z. Kabelac, D. Katabi, and R. C. Miller. 3D tracking via body radio reflections. Apr. 2014.

[6] F. Adib and D. Katabi. See through walls with wi-fi! 2013.

[7] B. Bloessl, C. Leitner, F. Dressler, and C. Sommer. A GNU Radio-based IEEE 802.15.4 Testbed. In *12. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN 2013)*, pages 37–40, 2013.

[8] M. Bocca, O. Kaltiokallio, and N. Patwari. Radio tomographic imaging for ambient assisted living. In *Evaluating AAL Systems Through Competitive Benchmarking*, pages 108–130. Springer, 2013.

[9] M. Bocca, A. Luong, N. Patwari, and T. Schmid. Dial it in: Rotating RF sensors to enhance radio tomography. *arXiv preprint arXiv:1312.5480*, 2013.

[10] T. Buzug. *Computed Tomography: From Photon Statistics to Modern Cone-Beam CT*. Springer, 2008.

[11] Y. Chen and A. Terzis. Calibrating RSSI measurements for 802.15.4 radios. Technical report, 2009.

[12] M. B. Cover, K. S. Kanukurthy, and D. R. Andersen. Microwave tomography using dynamic 802.15.4 wireless networks. In *2007 IEEE International Conference on Electro/Information Technology*, pages 251–256, 2007.

[13] A. Fhager and M. Persson. Comparison of two image reconstruction algorithms for microwave tomography: two algorithms for microwave tomography. *Radio Science*, 40(3), 2005.

[14] G. Herman. *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*. Springer, second edition, 2010.

[15] O. Iluz. Sniffing and decoding NRF24L01+ and Bluetooth LE packets for under $30. `http://blog.cyberexplorer.me/2014/01/sniffing-and-decoding-nrf24l01-and.html`, 2014.

[16] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–2014.

[17] O. Kaltiokallio, M. Bocca, and N. Patwari. Enhancing the accuracy of radio tomographic imaging using channel diversity. In *2012 IEEE 9th International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 254–262, 2012.

[18] L. E. Larsen and J. H. Jacobi, editors. *Medical applications of microwave imaging*. IEEE Press, 1986.

[19] J. L. Morales and J. Nocedal. Remark on "algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization". *ACM Trans. Math. Softw.*, 38(1):7:1–7:4, Dec. 2011.

[20] Y. Mostofi, S. Depatla, and L. Buckland. X-Ray Vision for Robots: Seeing Through Walls with Only WiFi. `http://www.ece.ucsb.edu/~ymostofi/SeeThroughImaging.html`, 2014.

[21] J. L. Mueller and S. Siltanen. *Linear and Nonlinear Inverse Problems with Practical Applications*. Society for Industrial and Applied Mathematics, 2012.

[22] N. Patwari and P. Agrawal. Effects of correlated shadowing: Connectivity, localization, and RF tomography. In *Proceedings of International Conference on Information Processing in Sensor Networks, IPSN'08.*, pages 82–93, 2008.

[23] A. Popleteev. Device-free indoor localization using ambient radio signals. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp '13 Adjunct, pages 549–552. ACM, 2013.

[24] S. Semenov. Microwave tomography: review of the progress towards clinical applications. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1900):3021–3042, 2009.

[25] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka. You are facing the Mona Lisa: Spot localization using PHY layer information. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, pages 183–196. ACM, 2012.

[26] D. Spill. BlueSniff: Eve meets Alice and Bluetooth. In *Proceedings of USENIX Workshop on Offensive Technologies (WOOT)*, 2007.

[27] K. Srinivasan and P. Levis. RSSI is under appreciated. In *Proceedings of the Third Workshop on Embedded Networked Sensors (EmNets)*, 2006.

[28] T. van der Meij. Constructing an open-source toolchain and investigating sensor properties for radio tomography, 2014. Bachelor thesis, Leiden University.

[29] J. Wetzl, O. Taubmann, S. Haase, T. Köhler, M. Kraus, and J. Hornegger. GPU-accelerated time-of-flight super-resolution for image-guided surgery. In *Bildverarbeitung für die Medizin 2013*, Informatik aktuell, pages 21–26. Springer, 2013.

[30] J. Wilson and N. Patwari. Radio tomographic imaging with wireless networks. Technical report, University of Utah, 2008.

[31] J. Wilson and N. Patwari. Through-wall tracking using variance-based radio tomography networks. *CoRR*, abs/0909.5417, 2009.

[32] J. Wilson and N. Patwari. Radio tomographic imaging with wireless networks. *IEEE Transactions on Mobile Computing*, pages 621–632, 2010.

[33] J. Wilson, N. Patwari, and F. G. Vasquez. Regularization methods for radio tomographic imaging. In *2009 Virginia Tech Symposium on Wireless Personal Communications*, 2009.