Leiden University

Computer Science

Mobile radio tomography:

Reconstructing and visualizing objects in wireless networks with dynamically positioned sensors

Name:	Tim van der Meij
Date:	August 26, 2016
1st supervisor:	Walter Kosters (LIACS)
2nd supervisor:	Joost Batenburg (CWI & MI)

MASTER'S THESIS

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

Abstract

This master's thesis provides an in-depth analysis of reconstruction and visualization in the context of mobile radio tomography. This new form of radio tomographic imaging uses specially designed unmanned vehicles with wireless sensors to detect objects inside a network. Information about objects inside the network is gathered by measuring the loss of strength of signals that travel through them. Converting the signal strength measurements to a two-dimensional image is a computationally expensive process because the reconstruction problem is an ill-posed inverse problem and because of the unstable nature of the measurements. Regularization is required to suppress noise in the measurements.

We describe the reconstruction problem and provide a survey of weighting models and regularization methods. Moreover, we introduce a new weighting model based on a Gaussian function and apply maximum entropy minimization to the problem of radio tomographic imaging. The open-source mobile radio tomography framework from previous work is extended with reconstruction and visualization components. The effectiveness of our approach is determined through a series of experiments with datasets from a real-world network.

Keywords: mobile radio tomography, framework, reconstruction, visualization

Contents

1 Introduction

Radio tomography is a technique for obtaining information about objects inside a network using radio frequency (RF) signals. It is based on the same principles as *tomography*, for which any kind of penetrating wave may be used to gather information about the internal structures of an object.

The network is a bounded physical region. Wireless sensors surround the network. Each sensor is connected to every other sensor, resulting in bidirectional *links* (lines) that cover the entire network. The sensors constantly send low-frequency signals (ZigBee, or WiFi-like, signals at around 2.4 GHz) to each other and the receiving end measures the signal strength. The setup is depicted in Figure 1.

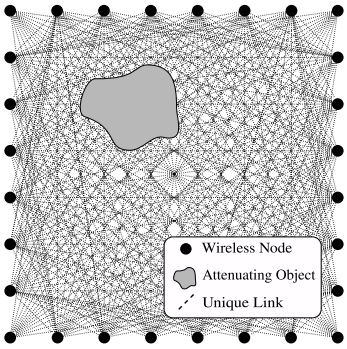


Figure 1: Overview of a radio tomography setup [1].

Object detection is based on the principle that a signal that travels through an object loses energy in the process. The signal strength for an unobstructed link is therefore higher than when the link is obstructed. Combining the signal strength measurements from the receiving ends of each link allows us to calculate where the object is located in the network, mainly because each link that crosses the object does this at a different angle. The more unique links we measure, the more information we obtain about the object, but also the longer the reconstruction process takes.

Tomography generally makes use of X-rays or other directional, high-frequency signals that (practically) travel in a straight line. However, radio tomography uses low-frequency signals that are not directional, i.e., signals may travel along any path from the sender to the receiver. This means that, aside from the signal that travels along the direct line-of-sight path, the receiver is also likely to measure signals from other paths. This phenomenon, which we discuss in Section 3.1, is especially problematic as signals that are reflected or refracted (for example by walls in indoor environments) lose energy and interfere with the line-of-sight signal that we are interested in. Having a means of suppressing noise, which we study in Section 5, is therefore of significant importance.

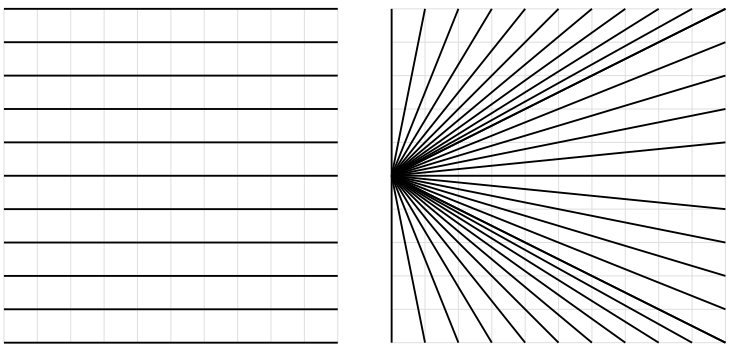
Creating a visualization, usually in the form of a two-dimensional image, from the information obtained using radio tomography is called *radio tomographic imaging*. This process is often intended to be real-time, allowing for immediate inspection of the reconstructed images.

Mobile radio tomography is a new form of radio tomography in which the statically positioned sensors around the network are replaced with sensors on moving unmanned vehicles, such as robotic rover cars or drones. Dynamically positioning the sensors makes it easier to deploy the hardware and paves the way for new applications and more advanced methods for scanning the network. However, at the same time new challenges arise, such as controlling the path of the vehicles and working with incomplete sequences of measurements. The objective of this work, combined with related work from Section 2, is to overcome these challenges and make mobile radio tomography a feasible new form of radio tomography.

1.1 Overview and objectives

The introduction described the concepts of tomography, radio tomography and mobile radio tomography. Mobile radio tomography is roughly divided into two stages, namely controlling the vehicles and their paths, as well as performing reconstruction and visualization of signal strength measurements. In this section we discuss the objectives of both stages and provide several applications of both radio tomography and mobile radio tomography.

The first stage is controlling the vehicles and their paths. In our setup, each vehicle is a rover car with dedicated hardware. The vehicles drive around on a grid to simplify tracking their exact location. More details regarding the hardware and the setup are provided in Section 6.1. The *mission* is the most important aspect of this stage as it defines the waypoints (locations on the grid) for each vehicle. It is the mission that is responsible for determining which links are measured and in which order. The decisions made at this stage directly affect the reconstruction and visualization stage as the former provides the data for the latter.



(a) Straight lines (b) Fan beam

Figure 2: Two common building blocks in missions.

The most common building blocks in missions are straight lines and fan beams. *Straight lines* are formed when two vehicles are exactly in opposite locations and move in the same direction. This building block is primarily used to quickly get an overview of the network. Figure 2a visualizes this building block. *Fan beams* are formed when one vehicle is stationary and another vehicle is driving on the

boundaries of the network. We speak of a *middle fan beam* if the origin is in the middle of a boundary and we speak of a *corner fan beam* when its origin is in a corner of the network. This pattern is suitable for measuring many links at once and discovering more details about objects as it takes care of intersecting many pixels from different angles. This building block is visualized in Figure 2b. Missions do not need to use these building blocks per se, for instance when an evolutionary *planning* algorithm is used to optimize the mission [2].

The wireless sensors on the vehicles perform signal strength meausurements for each link formed by the mission. The *received signal strength indicator* (*RSSI*) value is a measure of the power of the signal at the receiving end of the link. These values are transmitted to the *ground station* sensor, which is connected to a computer responsible for further processing.

The second stage consists of converting a sequence of signal strength meausurements to a two-dimensional image. This process is divided into *reconstruction* and *visualization* phases. Noise in the measurements is troublesome and therefore *regularization* or noise suppression is required. This work focuses entirely on this stage and is done in the context of creating an open-source mobile radio tomography framework.

1.2 Applications

Radio tomography has multiple advantages compared to other imaging methods. Deploying a wireless sensor network is both quick and inexpensive as the sensors are stand-alone and the costs are low when compared to the hardware required for other imaging methods (such as infrared cameras). Moreover, it is harder to block radio frequency signals than it is to block, for instance, a surveillance camera as the (harmless) low-frequency signals are capable of traveling through large obstructions. The accuracy of radio tomography is limited, which is an advantage when persons need to be tracked but not identified (for instance because of privacy regulations).

Mobile radio tomography specifically provides more flexibility for obtaining the measurements. Even though the hardware for a single vehicle is more expensive, the technique makes it possible to apply new scanning and imaging methods.

Applications for radio tomographic imaging are primarily security-based. Examples are monitoring restricted areas or buildings at night time. If movement is detected, an alarm is triggered and a security employee is able to inspect the images directly. Other applications are tracking-based, such as tracking how customers walk through a store. Tracking-based applications have a strong requirement of respecting the privacy of the tracked individuals. Radio tomographic imaging visualizes persons as blobs, which is ideal for gathering information about the behavior of the customers without having enough information to identify them.

Mobile radio tomographic imaging mainly provides a solution when radio tomographic imaging is impractical or when more fine-grained control is necessary. For instance, deployment of static sensors takes more time and the number of sensors needs to be increased drastically for large networks. Mobile radio tomography replaces these

static sensors with a smaller number of vehicles. Finally, we consider the case in which we wish to zoom in on objects. Mobile radio tomography makes this possible by moving the vehicles over the boundaries of the network towards the object and measuring local links to enhance the quality of the reconstruction. This *adaptive scanning* process updates the mission based on the measurements.

1.3 Research group

Our research in the field of mobile radio tomography is a collaboration between Leiden University and CWI Amsterdam. We formed a research group with members from both institutes for sharing knowledge about the subject, tracking progress and discussing issues and ideas.

In alphabetical order the members of the research group are Joost Batenburg (CWI), Folkert Bleichrodt (CWI), Leon Helwerda (Leiden University), Walter Kosters (Leiden University) and Tim van der Meij (Leiden University). Further assistance is provided by Willem Jan Palenstijn (CWI), Daan Pelt (CWI), Zhichao Zhong (CWI) and Xiaodong Zhuge (CWI).

This thesis is the result of over six months of research and forms the final work for the computer science master program at Leiden University. The work is supervised by Joost Batenburg and Walter Kosters.

2 Related work

Important research for applying regularization techniques to the problem of radio tomographic imaging was done by a group from the University of Utah in 2009 [1]. Later, in 2014, this work formed a basis for surveying, implementing and evaluating regularization algorithms for radio tomographic imaging using real-world networks [3].

The researchers of the University of Utah published a second paper in 2010, in which they studied radio tomographic imaging using a maximum a posteriori estimator [4], which is a statistical regularization method. This work showed that other regularization methods, in addition to those solved using a least squares or iterative approach, may also be applied to radio tomographic imaging.

The development of a communication protocol for radio frequency sensors by the University of Utah led to the construction of an open-source radio tomography toolchain and a study of sensor properties and network topologies in 2014 [5].

Mobile radio tomography is a new research area as of 2015, although many concepts of regular radio tomography remain applicable. Current work is done in the context of developing an open-source, component-based mobile radio tomography framework to facilitate research in this area. Components for wireless sensor communication [6], vehicle control [7] and mission planning [2] are present. This work completes the framework with reconstruction and visualization components.

3 Problem statement

Mobile radio tomography provides us with a sequence of signal strength measurements and the task of creating a reconstruction and visualization, in the form of a two-dimensional image, of the objects in the wireless network. Each measurement or link consists of three data points: the location of the source sensor, the location of the target sensor and the received signal strength indicator (RSSI) value. Formally, the sequence of signal strength measurements M consists of triples $\ell = (s, t, r)$ with s being the source sensor location, t being the target sensor location and r being the RSSI value. Therefore, $M = ((s_1, t_1, r_1), (s_2, t_2, r_2), \ldots, (s_k, t_k, r_k))$, in which k is the total number of measurements.

The reconstruction problem comprises converting M, the sequence of signal strength measurements, to a two-dimensional image. We approach the problem algebraically using matrix calculations. The input is a column vector $\vec{b}$ containing one entry per link, namely the RSSI value. The output is a 2D image of $m \times n$ pixels. The image consists of $m \cdot n$ square pixels that are evenly spaced in a two-dimensional grid that covers the space of the network. For the reconstruction problem, we model the image as a column vector $\vec{x}$ with $m \cdot n$ elements (stored in row-major order).

Moreover, we require a model that describes the influence of the pixel contents on each link. This model is represented by the weight matrix A. We discuss the weight matrix in detail in Section 4. The weight matrix describes the relation between signal strength values in $\vec{b}$ and pixel values in $\vec{x}$, allowing for the distribution of the signal strength value of a link over the pixels that have an influence on the link (according to their weights). Combining this information from all links makes it possible to reconstruct a two-dimensional image $\vec{x}$ of the objects in the network.

The reconstruction problem is expressed as the equation $A\vec{x} = \vec{b}$. Since A and $\vec{b}$ are known, we compute $\vec{x}$ by solving the inverse problem. We do not take a noise vector into account as we attempt to suppress any noise in the measurements using regularization techniques.

The difficulty of the reconstruction problem lies in the fact that the problem is an ill-posed inverse problem [1], meaning that there is generally not one unique solution to the problem. This is mainly caused by the unstable nature of the problem due to noise in the signal strength measurements. This, and the fact that the weight matrix is hardly ever square, makes the weight matrix not invertible in general. We have to apply regularization techniques to find a stable solution. The regularization methods are discussed in Section 5.

If the reconstructed image has size $m \times n$ and the total number of measurements is k, then A is of size $k \times (m \cdot n)$, $\vec{b}$ is of size $k \times 1$ and, by consequence, $\vec{x}$ is of size $(m \cdot n) \times 1$. Note that A and $\vec{b}$ have the same height k because each row at index i in A, representing a link, has a corresponding entry at index i in $\vec{b}$, representing the signal strength of the link, with $0 \leq i < k$.

3.1 Multipath interference

Wireless signal strength measurements generally contain a large amount of noise. Not only is this caused by reflection and diffraction of the signal by objects inside the network, but another major cause of noise is multipath interference.

Multipath interference is the effect in which the total signal strength measured by a target sensor is higher or lower than the signal strength from the signal traveling in the line-of-sight path between the source and target sensors. This happens because the wireless sensors are not directional, i.e., they send signals in virtually every direction. These signals are reflected by objects such as walls (for indoor environments) and could eventually reach the target sensor [8]. Figure 3 depicts examples of such signals.

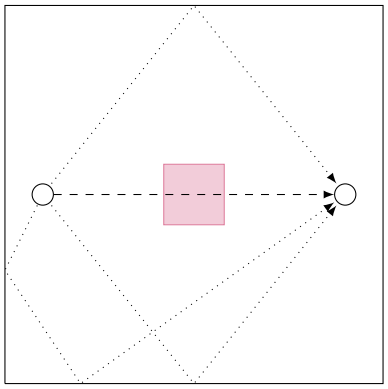


Figure 3: Illustration of multipath interference in a room surrounded by walls.

Even if the line-of-sight path between the source and the target sensor is blocked by an object, causing a low signal strength, the total signal strength measured by the target sensor may nevertheless be high because high-strength signals have reached the target sensor through other paths. This is the case for the situation in Figure 3, in which those paths are drawn with dotted lines. The line-of-sight path is indicated with a dashed line.

The surroundings of the network play a major role in the amount of multipath interference noise in the measurements. Even a small environmental variation may lead to a large difference in multipath effects. This noise is troublesome for the reconstruction process as it renders it nearly impossible to create a realistic reconstruction of the objects in the wireless network. Therefore it is of paramount importance to have a means of suppressing noise in the measurements.

Even though techniques exist to estimate the noise and to include that estimate in the model [4], we attempt to suppress noise using calibration measurements and regularization algorithms, i.e., not in our model.

We describe the regularization algorithms that we studied in Section 5 and the calibration procedure in Section 6.2.7.

4 Weighting

In Section 3 we briefly introduced the weight matrix. The *weight matrix* A describes the influence that the contents of the pixels have on the measured signal strength of a link. It is a matrix that contains a row for each link and a column for each pixel. If the contents of a pixel p have an influence on a link ℓ, according to a signal disruption model described later in this section, then the entry in the weight matrix in row ℓ and column p is nonzero. Otherwise the entry is zero [3, 4].

We model the wireless network as a pixel grid of size $m \times n$. Given a link $\ell = (s, t, r)$ the endpoints s and t define a line that intersects at least one pixel of the grid. It is important to know which pixels are intersected by the line as that information is used to determine the pixel values in the reconstructed image.

Multiple models exist to determine which pixels have an influence on a link and to weight them accordingly. These *signal disruption models* help in determining how the signal strength of a link is disrupted by objects in the network. Existing signal disruption models are the line and ellipse models [3, 4], which we study in Section 4.1 and Section 4.2, respectively. Furthermore, we explore a new signal disruption model based on a Gaussian function in Section 4.3.

4.1 Line model

The simplest signal disruption model is the *line model*. The model is based on the assumption that the signal strength of a link is primarily determined by objects lying on the line-of-sight path between the sensors. It assigns a weight of 1 to pixels that are intersected by this line and a weight of 0 to pixels that are not. Figure 4 visualizes how the line model assigns the weights for a link, in which the used line is depicted in red. The black pixels have weight 1 and the white pixels have weight 0.

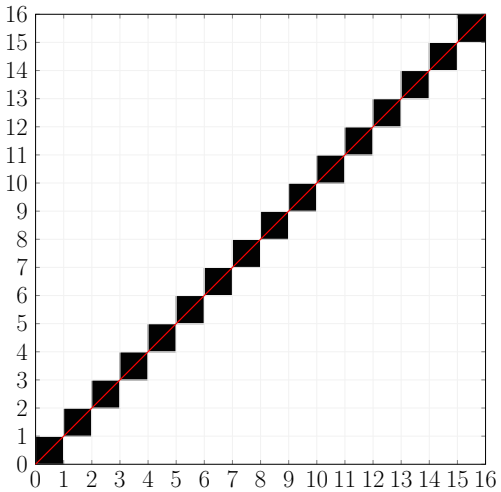


Figure 4: Weights for the link from $(0, 0)$ to $(16, 16)$ according to the line model.

In summary, the weight $w_{\ell,p}$ of a link ℓ and a pixel p is determined by Equation 1:

$$w_{\ell,p} = \begin{cases} 1 & \text{if } p \text{ is intersected by } \ell \\ 0 & \text{if } p \text{ is not intersected by } \ell \end{cases} \tag{1}$$

Even though this model is straightforward and not computationally expensive, it is not an accurate representation of how attenuating objects disrupt the signal strength in practice. Objects that are close to the line-of-sight path do influence the measured signal strength, albeit less than objects on the line-of-sight path.

4.2 Ellipse model

The *ellipse model* is a more realistic signal disruption model based on the definition of Fresnel zones. Fresnel zones are often used in radio communication theory to describe path loss between a sender and a receiver. A *Fresnel zone*, illustrated in Figure 5, is an ellipsoidal region with minor axis diameter λ and focal points at the endpoints of the link. Multiple Fresnel zones exist, with increasing minor axis diameters, but we only consider the first (smallest) Fresnel zone as the disruption caused by attenuating objects is most significant in this zone.

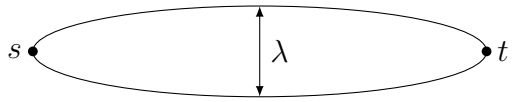


Figure 5: Illustration of a Fresnel zone with minor axis diameter λ. Given a link $\ell = (s, t, r)$, the focal points of the ellipse are s and t.

The model assigns a nonzero weight to pixels that are inside the ellipsoidal region and a weight of zero to pixels that are outside this region. Additionally, the length of the link, i.e., the distance between its two endpoints, is used to normalize the weights. This normalization enforces that shorter links get more weight than longer links. The rationale for the normalization step is that shorter links are preferred because the influence of other pixels than those on the line-of-sight path is minimized.

The weight $w_{\ell,p}$ of a link ℓ and a pixel p is determined by Equation 2, in which d_ℓ is the length of link ℓ and $d_{\ell,p}$ is the sum of the distances from the center of pixel p to the endpoints of link ℓ:

$$w_{\ell,p} = \begin{cases} \frac{1}{\sqrt{d_\ell}} & \text{if } d_{\ell,p} < d_\ell + \lambda \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Figure 6 visualizes how the ellipse model assigns the weights for a link, in which the used ellipse is depicted in red.

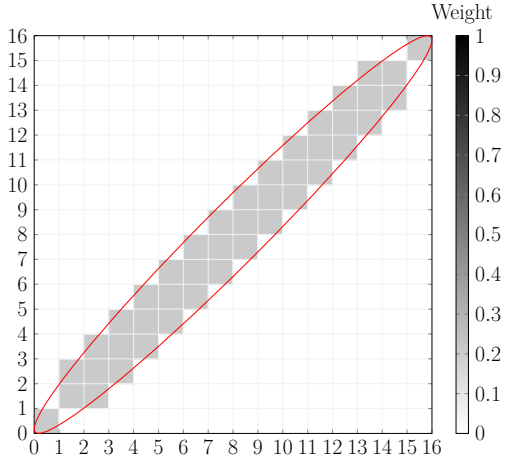


Figure 6: Weights for the link from $(0, 0)$ to $(16, 16)$ according to the ellipse model.

4.3 Gaussian model

The *Gaussian model* is a new signal disruption model that we propose as an alternative for the ellipse model. It is based on the assumption that the distribution of environmental noise, including multipath interference, conforms (reasonably well) to a normal or Gaussian distribution.

The model is based on the *log-distance path loss model*, one of many radio signal propagation models. The log-distance path loss model is defined in Equation 3 [9]:

$$PL_d = PL_{d_0} + 10\,\gamma \log_{10}\left(\frac{d}{d_0}\right) + \chi \tag{3}$$

In this equation PL_d is the path loss at distance d in decibels, d_0 is a reference distance (usually one meter) for which the signal strength is measured, γ is the path loss exponent (determined empirically for different environments) and χ is a Gaussian random variable. The log-distance path loss model therefore models environmental noise using a Gaussian random variable. This observation, together with the idea that a pixel should have less weight when it is farther away from the line-of-sight path, led to the idea of using a Gaussian distribution for the weight assignment.

The general Gaussian function is defined in Equation 4:

$$f(x) = \alpha\, e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4}$$

In this equation α is the height of the curve's peak, μ is the location of the peak's center and σ is the standard deviation that controls the width of the top of the curve. The meaning of these variables is depicted in Figure 7 along with a plot of the Gaussian function for $\alpha = 1$, $\mu = 0$ and $\sigma = 0.5$. The variable σ determines two inflection points of the curve at $x = \mu - \sigma$ and $x = \mu + \sigma$.

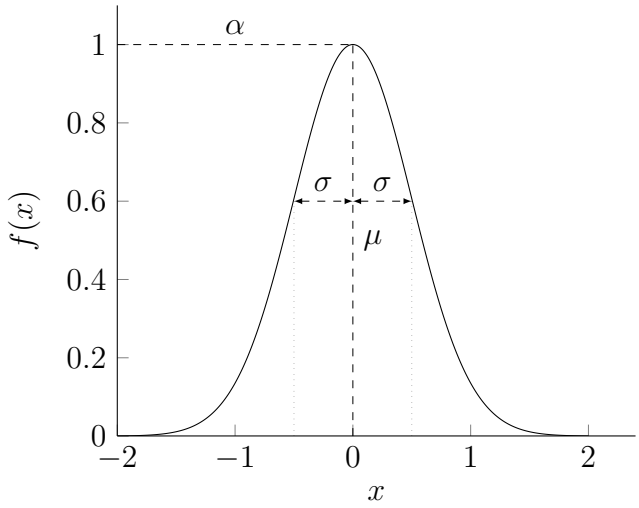


Figure 7: Gaussian function $f(x) = \alpha\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ with $\alpha = 1$, $\mu = 0$ and $\sigma = 0.5$.

The Gaussian model makes use of the Gaussian function to assign weights for the pixels. The idea is to give most weight to pixels on the line-of-sight path and less weight to pixels that are farther away, depending on their distance from the line-of-sight path. To achieve this objective, we specifically use a Gaussian function with $\alpha = 1$ and $\mu = 0$ as this ensures that pixels on the line-of-sight path get the highest weight (as their distance to the line-of-sight path is zero or very close to zero). The parameter σ may be tuned as in practice a wide range of values appears to be suitable.

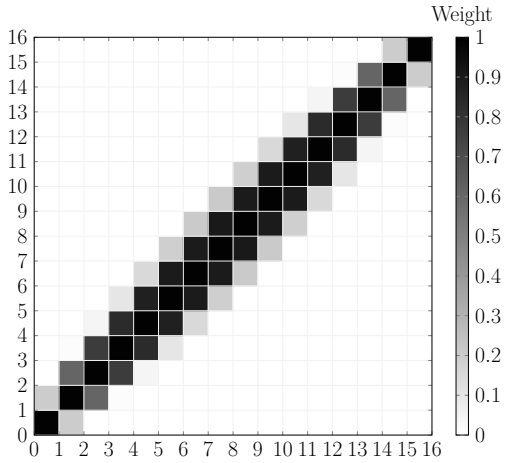


Figure 8: Weights for the link from $(0, 0)$ to $(16, 16)$ according to the Gaussian model.

The weight $w_{\ell,p}$ of a link ℓ and a pixel p is determined by Equation 5, in which d_ℓ is the length of link ℓ and $d_{\ell,p}$ is the sum of the distances from the center of pixel p to the endpoints of link ℓ:

$$w_{\ell,p} = e^{-\frac{(d_{\ell,p}-d_\ell)^2}{2\sigma^2}} \tag{5}$$

Figure 8 visualizes how the Gaussian model assigns the weights for a link. Pixels on the line-of-sight path have most weight since $d_{\ell,p} - d_\ell \approx 0$ and the Gaussian function then assigns $w_{\ell,p} \approx 1$. The farther away pixels are from the line-of-sight path, the less weight they are assigned, as the value of the Gaussian function quickly decreases when $x = d_{\ell,p} - d_\ell$ increases, as seen in Figure 7.

4.4 Dynamic weight matrix

It is possible to precompute the weight matrix A if the sensor positions are static as the number of links is fixed. This is beneficial for real-time reconstruction as the weight matrix is constant. However, our approach in which the sensor positions are dynamic complicates this. We lose the a priori knowledge about the number of links, making it impossible to precompute the weight matrix and thus rendering real-time reconstruction more computationally expensive.

To address this issue, we propose an algorithm, described in Algorithm 1, for efficiently creating and updating a weight matrix when the sensor positions are dynamic.

Algorithm 1 Creating and updating a dynamic weight matrix

1. Create an empty weight matrix with $m \cdot n$ columns.
2. Create an empty distance matrix with $m \cdot n$ columns.
3. Create a sensor storage as an empty dictionary.
4. When a signal strength measurement packet arrives:
   (a) Extract the source and target sensor locations from the packet.
   (b) Check if the source sensor location exists in the sensor storage. If so, get its index in the distance matrix. If not, insert it into the sensor storage and get its index for the distance matrix. Repeat this for the target sensor location.
   (c) Update the distance matrix by adding a row for each *newly inserted* sensor location from step 4b. The values in the row are the distances from the sensor location to each center of a pixel on the grid. Calculate the distances using the Pythagorean theorem. Let $s = (s_x, s_y)$ be the sensor location and $p = (p_x, p_y)$ be the location of the center of a pixel on the grid, then the distance from s to p is $\sqrt{(p_x - s_x)^2 + (p_y - s_y)^2}$.
   (d) Update the weight matrix by adding a row for the new link. Use the line, ellipse or Gaussian model in combination with the rows from the distance matrix, pointed to by the row indices from step 4b, to set the weight of each pixel.

Using dynamically positioned sensors adds some overhead to update the weight matrix. To reduce this overhead as much as possible, Algorithm 1 makes use of caches to avoid unnecessary recalculations.

The sensor storage is a dictionary that maps sensor locations to row indices in the distance matrix. It is used as a quick look-up table to determine if the distances from a sensor location to all pixels have been calculated before. If so, we use the row index

to fetch the distances from the distance matrix to avoid recalculation. The distance matrix has rows for each sensor location and columns for each pixel on the grid. The value in row s and column p is the distance from sensor s to the center of pixel p.

5 Regularization

In Section 3 we introduced the reconstruction problem. The problem is expressed as the equation $A\,\vec{x} = \vec{b}$. To obtain $\vec{x}$, the pixel values of the reconstructed image, we need to solve the inverse problem. Contrary to a *forward problem*, in which the effect is calculated from the causes of the problem, mobile radio tomography provides us with an *inverse problem*, as we start with the effect (the vector of signal strengths $\vec{b}$) and determine the causes (pixel values in $\vec{x}$) that produced it under our model A.

We wish to solve the inverse problem using $\vec{x} = A^{-1}\vec{b}$. Unfortunately, this is not possible because the problem is ill-posed and because the weight matrix A is not invertible in general, as outlined in Section 3. A *well-posed problem* is a problem for which a unique solution exists and for which the solution depends continuously on the data, which means that small changes in the data result in small changes in the solution. Following that definition, an *ill-posed problem* is a problem that is not well-posed. To find a stable solution, we need to incorporate additional information or requirements to solve this ill-posed problem. This process is known as regularization.

Solving $A\,\vec{x} = \vec{b}$ exactly implies that the error $A\,\vec{x} - \vec{b}$ is equal to zero. Since we cannot solve $A\,\vec{x} = \vec{b}$ exactly, we attempt to find a solution $\vec{x}$ that minimizes the error. Multiple techniques exist for solving this minimization problem and applying regularization. First, we discuss least squares approximation in Section 5.1 as it forms the basis for all regularization methods. Next, common regularization methods for radio tomographic imaging are discussed, namely singular value decomposition in Section 5.2, truncated singular value decomposition in Section 5.3 and total variation minimization in Section 5.4. Finally, we study maximum entropy minimization in Section 5.5, which has not been applied to radio tomographic imaging before.

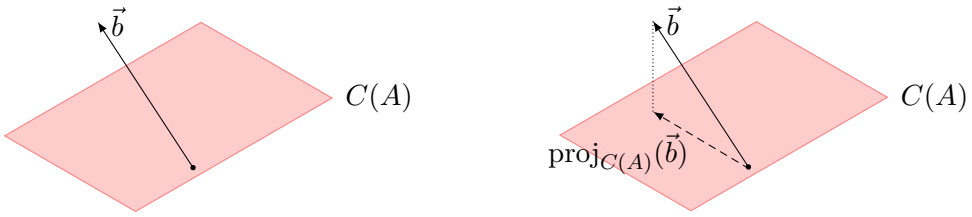
5.1 Least squares approximation

For problems working with models and measurements of physical phenomena it is relatively common that the equation $A\,\vec{x} = \vec{b}$ has no exact solution for $\vec{x}$. Instead, a minimization problem is solved to find a solution for $\vec{x}$ that minimizes the error. One of the most often used techniques for solving this minimization problem is the method of least squares, also known as *least squares approximation*.

The minimization problem for obtaining the least squares approximation $\vec{x}_{ls}$ of $\vec{x}$ is defined in Equation 6 [10], in which $\|\cdot\|_2$ is the L^2 norm outlined later in this section:

$$\vec{x}_{ls} = \arg\min_{\vec{x}} \left\| A\,\vec{x} - \vec{b} \right\|_2^2 \tag{6}$$

In general there is no exact solution to $A\,\vec{x} = \vec{b}$, thus no linear combination of the column vectors of A is equal to $\vec{b}$ or, equivalently, $\vec{b}$ is not in the column space $C(A)$ of A, i.e., $\vec{b} \notin C(A)$. This is depicted in Figure 9a in which $C(A)$ is visualized as a plane and $\vec{b}$ is not inside this plane.



(a) $\vec{b} \notin C(A)$ (b) $\text{proj}_{C(A)}(\vec{b}) \in C(A)$

Figure 9: Visualizations of least squares approximation of $\vec{b}$.

Least squares approximation finds a solution $\vec{x}_{ls}$ such that $A\,\vec{x}_{ls} \in C(A)$ is as close to $\vec{b}$ as possible. This implies that the length of the vector $A\,\vec{x}_{ls} - \vec{b}$ must be as small as possible.

We calculate the length of the vector $A\,\vec{x}_{ls} - \vec{b}$ using the Euclidian or L^2 norm. Given a column vector $\vec{v} = [v_0, v_1, \ldots, v_{n-1}]^T$, the length of $\vec{v}$ using the Euclidian norm is $\|\vec{v}\|_2 = \sqrt{\sum_{i=0}^{n-1} (v_i)^2}$. The norm is usally squared, i.e., one uses $\|\vec{v}\|_2^2 = \sum_{i=0}^{n-1} (v_i)^2$ to eliminate the square root. Applying this to the vector $A\,\vec{x}_{ls} - \vec{b}$ gives the length to minimize as $\left\|A\,\vec{x}_{ls} - \vec{b}\right\|_2^2$. This leads to the minimization problem definition in Equation 6. To minimize this value we minimize the values of the individual squares, hence the name of the approximation method.

By definition, the closest vector $\vec{v^*}$ in a subspace S to a vector $\vec{v}$ that is not inside S is the *projection* $\vec{v^*}$ of $\vec{v}$ onto S: $\vec{v^*} = \text{proj}_S(\vec{v})$. Finding the projection of $\vec{b}$ onto $C(A)$ thus leads to a least squares approximation, which is depicted in Figure 9b. Therefore, instead of solving $A\,\vec{x} = \vec{b}$, we solve $A\,\vec{x}_{ls} = \text{proj}_{C(A)}(\vec{b})$.

Subtracting $\vec{b}$ from both sides yields $A\,\vec{x}_{ls} - \vec{b} = \text{proj}_{C(A)}(\vec{b}) - \vec{b}$. The right-hand side of the equation is now orthogonal to $C(A)$, thus $A\,\vec{x}_{ls} - \vec{b}$ is in the orthogonal column space of A or, equivalently, in the left nullspace of A^T. The latter is defined as $\mathcal{N}(A^T) = \{x \in \mathbb{R}^m \;:\; A^T x = 0\}$ for any $m \times n$ matrix A, which implies that $A^T(A\,\vec{x}_{ls} - \vec{b}) = 0$. We further rewrite this to $A^T A\,\vec{x}_{ls} - A^T\vec{b} = 0$ to obtain the *normal equations* $A^T A\,\vec{x}_{ls} = A^T\vec{b}$. Solving the normal equations leads to $\vec{x}_{ls}$, the least squares solution.

5.2 Singular value decomposition

One way of solving the normal equations is using the pseudoinverse of A. Rearranging the normal equations gives $\vec{x}_{ls} = (A^T A)^{-1} A^T\vec{b} = A^\dagger\vec{b}$ in which $A^\dagger = (A^T A)^{-1} A^T$ is the *Moore-Penrose pseudoinverse*, which generalizes the notion of the inverse of

square matrices for non-square matrices [11]. Finding the pseudoinverse for a matrix A is possible by computing the singular value decomposition of A.

For an $m \times n$ matrix A with $m \geq n$ the *singular value decomposition* (*SVD*) of A is defined as $A = U\Sigma V^T$. We decompose A into three matrices:

1. U, an $m \times m$ orthogonal matrix in which the columns (*left singular vectors*) are the eigenvectors of the matrix AA^T;
2. V, an $n \times n$ orthogonal matrix in which the columns (*right singular vectors*) are the eigenvectors of the matrix A^TA;
3. Σ, an $m \times n$ diagonal matrix of the eigenvalues of the matrices AA^T and A^TA. The diagonal entries are the *singular values*, denoted by σ_i for $1 \leq i \leq r$ in which $r = \min(m, n)$ is the rank of A.

We refer the reader to [12] for a derivation of the singular value decomposition.

In particular we are interested in obtaining the Moore-Penrose pseudoinverse $A^\dagger$ from the singular value decomposition $A = U\Sigma V^T$ of matrix A. We use the derivation in Equation 7 to obtain $A^\dagger$:

$$\begin{aligned} A^\dagger &= (A^TA)^{-1}A^T \\ &= ((U\Sigma V^T)^T(U\Sigma V^T))^{-1}(U\Sigma V^T)^T \\ &= ((V\Sigma^TU^T)(U\Sigma V^T))^{-1}(V\Sigma^TU^T) \\ &= (U\Sigma V^T)^{-1}(V\Sigma^TU^T)^{-1}(V\Sigma^TU^T) \\ &= (U\Sigma V^T)^{-1} \\ &= (V^T)^{-1}\Sigma^{-1}U^{-1} \\ &= V\Sigma^{-1}U^{-1} \\ &= V\Sigma^{-1}U^T \end{aligned} \tag{7}$$

We substitute the singular value decomposition $U\Sigma V^T$ for A. Furthermore, we apply the property that the transpose of a product of matrices is equal to the product of the transposed matrices in reversed order. Note that the same property holds for the inverse operation.

We also use the property that the product of a matrix and its inverse is equal to the identity matrix. Finally, the matrices U and V are orthogonal, which is a useful property of the SVD, and therefore $U^T = U^{-1}$ and $V^T = V^{-1}$. The matrix Σ^{-1} is obtained from Σ by taking the reciprocal of each diagonal entry [13].

In conclusion, the singular value decomposition of A makes it possible to find the pseudoinverse $A^\dagger$ of A, which in turn makes it possible to directly calculate the least squares solution $\vec{x}_{ls}$ for a given vector $\vec{b}$ using $\vec{x}_{ls} = A^\dagger\vec{b}$.

5.3 Truncated singular value decomposition

The singular value decomposition in Section 5.2 provides an exact decomposition of the original matrix. However, we wish to apply regularization to suppress noise in the measurements. This is necessary because using the singular value decomposition directly leads to major instability as the reconstruction problem is ill-posed. The difference between the smallest and largest singular values is significant and therefore noise has a major effect on the final images.

To reduce this effect, we need to limit the decomposition. The *truncated singular value decomposition* (*TSVD*) is a singular value decomposition for which only the τ largest singular values (and corresponding left and right singular vectors) are calculated [1]. The truncated singular value decomposition of A is defined as $A = U_\tau \Sigma_\tau V_\tau^T$.

This decomposition is often used for *dimensionality reduction*, for which the objective is to find a matrix of a lower rank that is the best possible approximation of the original matrix, i.e., in which the most important information is retained. It is often applied in the fields of data compression and data mining to store the most important information with the least amount of data.

Not only does the truncated singular value decomposition stabilize the solution by removing the smallest singular values that have a major effect on the final images, it is also faster to calculate.

5.4 Total variation minimization

Truncated singular value decomposition has the disadvantage that it does not provide a means of incorporating desired characteristics of the resulting images. Indeed, the singular vectors are only dependent on the measured links and the used signal disruption model. While dimensionality reduction does stabilize the solution, the resulting images may still contain unstable spots.

We wish to enforce that the resulting images are smooth, i.e., that the differences between neighboring pixels are as small as possible. *Total variation minimization* (*TV*) is a regularization method that incorporates this constraint by favoring solutions that minimize slow changes in the resulting image (to maintain sharp edges). Originally, the total variation minimization solution $\vec{x}_{tv}$ is defined in Equation 8 [1]:

$$\vec{x}_{tv} = \arg\min_{\vec{x}} \left\| A\,\vec{x} - \vec{b} \right\|_2^2 + \alpha \sum_{i=0}^{n-1} |\nabla \vec{x}|_i \tag{8}$$

In this definition n is the number of elements in the gradient of $\vec{x}$ and α describes the importance of the total variation factor. The parameter α indicates a trade-off for the regularization: if α is high, then noise suppression is increased, but at the same time the solution corresponds less to the actual measurements.

The gradient of $\vec{x}$, $\nabla\vec{x}$, is used in the minimization procedure as it provides a measure of the variability of the solution. Minimizing the total variation implies a smoother image as there are fewer differences between neighboring pixels.

The disadvantage of this definition is that the additional term is not squared, which makes it impossible to use a least squares approach for the solution. This requires us to use an optimization algorithm for the minimization, making the solution process more time-consuming [3].

Some optimization algorithms require (an approximation of) the derivative of the function that we wish to minimize. Let $f(\vec{x}) = |\nabla\vec{x}|$ or, equivalently, $f(\vec{x}) = \sqrt{(\nabla\vec{x})^2} = \left((\nabla\vec{x})^2\right)^{\frac{1}{2}}$, then the derivative $f'(\vec{x})$ is calculated as in Equation 9:

$$\begin{aligned} f'(\vec{x}) &= \frac{1}{2}\left((\nabla\vec{x})^2\right)^{-\frac{1}{2}} \cdot 2\nabla\vec{x} \\ &= \left((\nabla\vec{x})^2\right)^{-\frac{1}{2}} \cdot \nabla\vec{x} \\ &= \frac{\nabla\vec{x}}{\sqrt{(\nabla\vec{x})^2}} \end{aligned} \tag{9}$$

Clearly, this derivative is undefined for $\vec{x} = 0$. To solve this problem, we may use a function that is differentiable, namely $f(\vec{x}) = \sqrt{(\nabla\vec{x})^2 + \beta}$ for a small value of β, to prevent discontinuity for $\vec{x} = 0$. This function is an approximation, so we need to set β to a very small value to not deviate from the original function too much.

In conclusion, we define the total variation minimization solution $\vec{x}_{tv}$ in Equation 10:

$$\vec{x}_{tv} = \arg\min_{\vec{x}} \left\|A\,\vec{x} - \vec{b}\right\|_2^2 + \alpha \sum_{i=0}^{n-1} \sqrt{(\nabla\vec{x})_i^2 + \beta} \tag{10}$$

5.5 Maximum entropy minimization

Instead of considering the gradient as a measure of the variability of the solution, we may also use other measures, such as the entropy.

Entropy is a concept in thermodynamics that provides a measure of the amount of disorder in a structure. Highly ordered structures have a low entropy, but when the structure is heated, for example, its entropy increases as the structure ends up in a less ordered state. More precisely, entropy corresponds to the number of states that a structure may be in. Therefore, a low entropy structure may be in a small number of possible states, whereas a high entropy structure may be in a large number of different states.

This principle is also used for image processing and other fields in information theory. If the structure is a grayscale image, then the states are the possible gray levels of the pixels. This observation makes entropy potentially useful for application in a regularization context. Indeed, the purpose of regularization is to suppress noise

in the reconstructed images and noise typically manifests itself in pixels of highly varying gray levels, so if noise is present, the entropy of the solution is increased.

We use the Shannon entropy as a measure of the amount of disorder in a solution. The *Shannon entropy* H of an image is defined in Equation 11 [14]:

$$H = -\sum_{i=0}^{n-1} p_i \log_2(p_i) \tag{11}$$

In this definition n is the number of unique gray levels in the solution and p_i is the probability that the unique gray level i occurs in the solution. The probability p_i is thus equal to the number of pixels with gray level i divided by the total number of pixels in the solution. Note that this definition requires an optimization algorithm for the minimization. We calculate the derivative of the objective function numerically.

The purpose of *maximum entropy minimization* (*ME*) is to incorporate the constraint that the entropy of the solution must be low. This enforces that a solution may not have a high variability in terms of unique gray levels, which should result in smooth reconstructed images. This objective is thus the same as that of total variation minimization, but we make use of a different variability measure to achieve it.

While this regularization technique is not new [15], we have found no previous work about the application of this method to the problem of radio tomographic imaging.

In summary, we define the maximum entropy minimization solution $\vec{x}_{me}$ in Equation 12:

$$\vec{x}_{me} = \arg\min_{\vec{x}} \left\| A\,\vec{x} - \vec{b} \right\|_2^2 + \alpha \left(-\sum_{i=0}^{n-1} p_i \log_2(p_i) \right) \tag{12}$$

6 Implementation

The novelty of the research area provides us with the opportunity to create a solid basis for both current and future research. Our work is therefore done in the context of developing an open-source, component-based mobile radio tomography framework.

The objective of the framework is to provide the required tools for performing mobile radio tomography. It is open-source to ensure that everyone is free to use it and to inspect and extend its code. The functionality of the framework is split into multiple packages and modules to promote separation of concerns, reusability and extensibility. The framework is written in Python, making it easy to use it on different operating systems with minimal changes, and is highly object-oriented.

Testing plays a significant role for this framework. We need to ensure that the functionality is working as expected, especially since we are working with moving hardware of which we control the behavior. Unit tests that result in 100% line coverage and 100% (public) method coverage of all directly used packages and modules provide confidence in the implementation.

The contribution of this work is the development of reconstruction and visualization components for the existing framework [6, 7]. We implement the models from Section 4 and the algorithms from Section 5 and we extend the wireless communication code to support more types of hardware. Finally, we develop an advanced graphical user interface to unify interaction with the framework.

We describe the hardware for our work in Section 6.1 and the software for reconstruction and visualization in Section 6.2.

6.1 Hardware

In order to perform mobile radio tomography, we need at least two vehicles, which in our case are rover cars equipped with control units and sensors, but may also be drones or other moving hardware. Our vehicle is pictured in Figure 10.

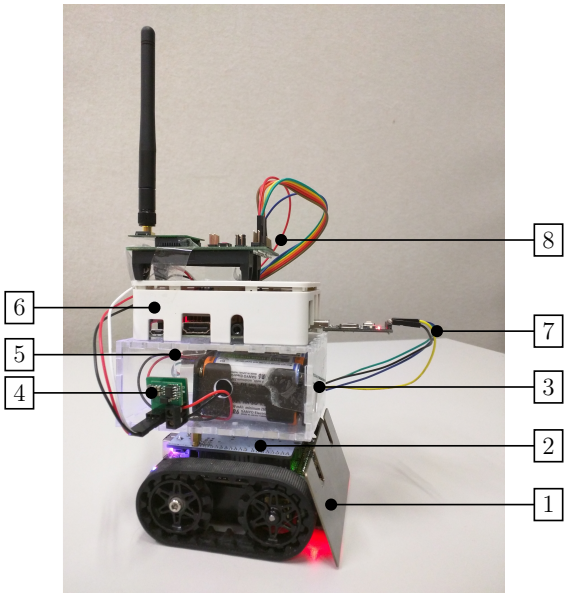


Figure 10: A vehicle for performing mobile radio tomography.

The vehicle is a stack of multiple components. Each component in Figure 10 is assigned a number that corresponds to the numbers below. We discuss the construction of the vehicle in terms of the individual components:

1. **Rover car**. The pre-assembled *Zumo Robot for Arduino* [16] forms the basis of the vehicle. The 10 × 10 cm Arduino-controlled robot has two motors, two rubber tracks and a 4 × AA battery holder. Important is that the robot comes with line following hardware that we use to make the vehicle drive to specific locations inside the network.

2. **Arduino**. The *Arduino Uno* [17] is an open-source hardware prototyping board. Its microcontroller is programmed with a modified version of C. The Arduino is mainly responsible for controlling the motors of the rover car and reading values from the line follower for determining its path.

3. **Casing**. The plastic casing is made from a custom design, using a laser cutting machine. It is a container for a battery pack and separates the Arduino from other hardware on the vehicle. The top of the case is removable for replacing the batteries.

4. **Power converter and battery pack**. The battery pack in the casing contains eight AA batteries. The power converter is required to safely power the Raspberry Pi off the batteries in the battery pack.

5. **Infrared sensor**. The infrared sensor makes it possible to start or stop the measurements with a remote control. Furthermore, it provides us with a manual kill switch that immediately stops the vehicle in case we, for example, foresee a collision. We use the open-source LIRC software to bind infrared signals to remote control buttons. Once an incoming infrared signal is recognized, our framework performs an action based on which button was pressed.

6. **Raspberry Pi**. The *Raspberry Pi 2 model B* [18] is a complete single-board computer and serves as the main control unit of the vehicle. Equipped with the minimal Arch Linux ARM operating system running Python code from our framework, its responsibilities include, but are not limited to, triggering and processing data from the wireless sensor, processing data from the infrared sensor and sending commands to the Arduino to move the vehicle to a particular location in the network.

7. **USB to TTL converter**. This converter is used to establish a serial connection from the Raspberry Pi to the Arduino and therefore makes it possible to send commands to the Arduino and the rover car over USB.

8. **Wireless sensor**. The wireless sensor takes care of sending and receiving packets on a fixed schedule for signal strength measurements. We use the wireless sensor from the *Texas Instruments CC2530 Development Kit* [19] as it gives us more stable signal strength measurements than the XBee hardware from previous work [6], although our framework has built-in support for both brands of wireless sensors.

Tracking the location of the vehicle in the network is of major importance. The locations of the source and target sensors are necessary in the reconstruction phase to define the link. Knowing the location of the vehicle is also essential for executing missions properly. The line following hardware, in combination with driving on a grid, enables us to accurately maintain the location of the vehicle.

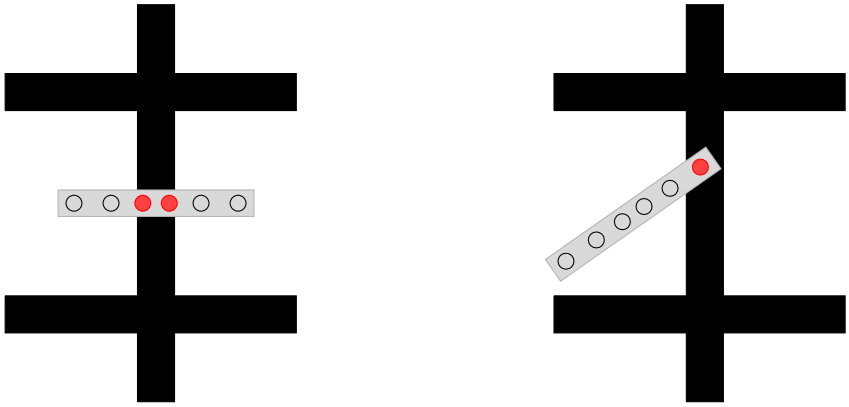
We provide the vehicle with two values when it starts: its *home location*, i.e., the coordinates of the vehicle on the grid, and its *home direction*, which is the direction the vehicle is facing. Intersections on the grid represent valid locations.

Updating the location is done by querying the line following hardware when driving. If we detect a straight line, then we have not yet reached a new location. However, if we detect an intersection of lines, then we did reach a new location. The new location is one grid cell farther than the old location, in the direction the vehicle is facing.

For example, if the previous location is $(4, 0)$ and the vehicle is going in eastern direction, then the current location becomes $(5, 0)$, i.e., we move on the x-axis.

Figure 11: Line following hardware with an array of six sensors.

The line following hardware consists of an array of six sensors, which are pairs of infrared LEDs and photodiodes. The sensors, highlighted in Figure 11, provide independent digital outputs. Since the grid consists of black lines on a white background, the functionality of the sensors is based on the principle that black absorbs and white reflects infrared signals. The sensor works by activating the infrared LED for several microseconds and measuring the amount of reflection. If the amount of reflection is below a threshold, we consider the space under the sensor to be part of a line (black) and otherwise it is empty space (white).



(a) Detecting a line.

(b) Detecting divergence from a line.

Figure 12: Detecting a line and divergence from a line with the line following hardware.

Figure 12a visualizes how the line following hardware detects a line. Only the sensors that are above the black line give a reading below the threshold. Combining this with the observation that those sensors are the two middle sensors tells us to follow the line until we reach the next intersection (when all sensors give a reading below the threshold). Figure 12b visualizes the situation in which the vehicle has diverged from the line, either when turning or by mistake. Detecting that only the rightmost sensor gives a reading below the threshold means that we should recover by turning the vehicle to the right until the two middle sensors are the ones that give a reading below the threshold. We use these recovery options in practice to ensure that the vehicles do not diverge from the grid.

The grid-based solution for location tracking is chosen for its high stability and accuracy. We did evaluate GPS-based solutions, but found that they do not provide the accuracy we need and that they generally have major stability issues when used indoors. Another disadvantage is that GPS-based solutions are generally expensive, whereas the grid-based solution is not. However, a disadvantage of the grid-based solution is that it is relatively cumbersome to set up.

6.2 Software

This work primarily extends the mobile radio tomography framework with components required for reconstruction, which are bundled in the `reconstruction` package. Moreover, we make the components for wireless sensor communication more generic to provide built-in support for multiple brands of wireless sensors that use ZigBee, in particular Texas Instruments CC2530 and XBee sensors. These components are bundled in the `zigbee` package.

The components in the `zigbee` and `reconstruction` packages work together to form a visualization of the network. Both packages are designed with low coupling and high cohesion in mind. The UML class diagram for both packages is displayed in Figure 13. We show the classes without their properties and methods for brevity and discuss these and other components in the following sections.

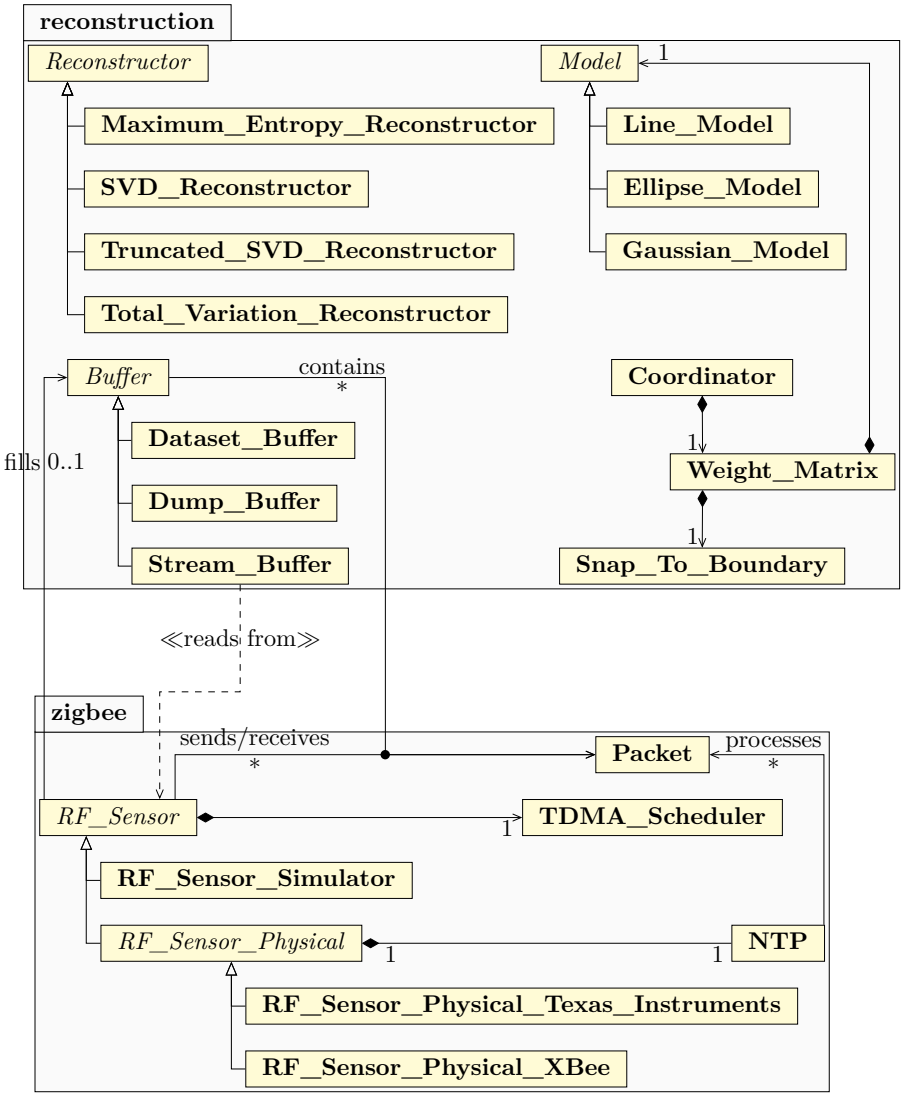


Figure 13: UML class diagram for the `reconstruction` and `zigbee` packages.

6.2.1 Reconstructors and models

The `reconstruction` package contains abstract classes for reconstructors and models, namely `Reconstructor` and `Model`, respectively. *Reconstructors* are implementations of regularization methods, such as those described in Section 5. `Models` are weighting models, such as those described in Section 4.

Implementation classes of reconstructors and models must extend these abstract classes to ensure that they share the same interface for usage by other components in the framework. The benefits of this approach are understandable code and improved extensibility of the framework. Creating a new reconstructor or model is reduced to making a class that extends the corresponding abstract class and implements the required methods, and adding it to the list of available reconstructors or models. Figure 13 shows that the framework contains implementation classes for all weighting models from Section 4 and all regularization methods from Section 5.

6.2.2 Buffers

The reconstruction components are capable of using data from multiple data sources. We are interested in streaming data for real-time reconstruction as well as data from dumps or datasets for analysis at a later time. To meet this requirement, we use *buffers* that extend the abstract `Buffer` class and provide the measurement data to other components in a structed manner. The buffers internally use a FIFO (first in, first out) queue.

The primary buffer is the *stream buffer*, which is filled by the ground station sensor when measurements arrive. If we are performing calibration measurements, the stream buffer simply returns the original packet. In other cases, the stream buffer takes care of reading a file with calibration measurements and returns the original packet as well as the calibrated RSSI value. The calibrated RSSI value is then used in the reconstruction phase.

The framework provides a means for recording measurements and storing them as a dump file. The dump files are JSON (JavaScript Object Notation) [20] files containing the following fields:

- number of sensors in the network (excluding the ground station sensor);
- origin of the network;
- width and height of the network;
- captured (raw) packets.

We create a *dump buffer* for loading these files to replay an entire mission. This makes it possible to perform analyses at a later time, as well as studying the effects of different regularization methods, weighting models and associated parameters. Reading a calibration dump and obtaining calibrated measurements is also possibile.

Moreover, we implement a *dataset buffer* for reading radio tomography datasets, in CSV (comma separated values) format, from the University of Utah [21]. While these datasets were made using a setup with statically positioned sensors, they are helpful for implementation verification and parameter tuning.

6.2.3 Coordinator

The task of the *coordinator* component is to streamline updating the reconstruction when a new measurement arrives. It maintains the weight matrix and RSSI vector and provides an interface for other components to use them.

We try to keep the number of rows in the weight matrix, and therefore the number of entries in the RSSI vector, small to reduce the time required for the regularization steps. However, at the same time we must ensure that enough data is available for those steps. To achieve this objective, the coordinator implements a replacement policy for incoming measurements.

When a new measurement arrives, the coordinator extracts the endpoints of the link and checks if the endpoints are present in a cache. This tells us if the link has been processed before. In case the endpoints are not in the cache, the coordinator instructs the weight matrix component to perform an update (which appends a new row to the weight matrix), appends the RSSI value in the packet to the RSSI vector and updates the cache to include the endpoints.

However, if the endpoints are present in the cache, we perform a replacement operation. There is no need to append a new row to the weight matrix as it is identical to the one that is already present for the link. Instead, we obtain the index of that row from the cache and only update the corresponding value in the RSSI vector.

The advantage of this approach is not only that the weight matrix and RSSI vector remain compact and fast to process, but the policy automatically takes care of replacing outdated measurements. The latter is especially useful for reconstruction of a dynamic environment with moving objects or persons.

6.2.4 Weight matrix and snap to boundary

We implement the dynamic weight matrix as described in Section 4.4 in the `Weight_Matrix` class. It is responsible for initializing and using one of the model classes for assigning weights to pixels.

Most of the time the vehicles drive on the boundaries of the network. However, we may add free space around the network in which the vehicles are allowed to drive, for instance to avoid collisions or for measuring more different links. Instead of discarding these measurements because the endpoints of the link are not on the boundary of the network, thereby losing information and time, we may attempt to alter them so that they become useful for the reconstruction phase.

The idea is to define a transformation that snaps endpoints of the link that are outside the network to the closest boundary of the network. This transformation provides us with an approximation of the original link as its length is decreased. Since distance and RSSI are correlated and the RSSI value does not change, this does influence the weighting model somewhat. There is a trade-off between reconstruction accuracy and information/time gain.

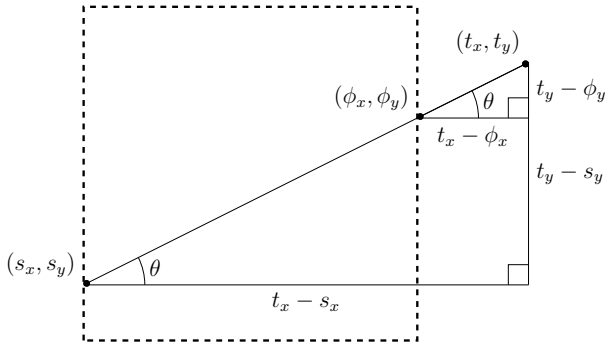


Figure 14: Illustration of snapping a point to the rightmost network boundary.

Snapping an endpoint to the rightmost boundary of the network is illustrated in Figure 14, but the transformations are equivalent for the other boundaries. The network is depicted with a dashed line. The endpoints of the link are (s_x, s_y) for the source sensor and (t_x, t_y) for the target sensor. Since (t_x, t_y) is outside the network, we snap it to the rightmost boundary to obtain (ϕ_x, ϕ_y). Notice that the slope of the line remains the same and only its length is decreased by the transformation. Moreover, if the link from (s_x, s_y) to (ϕ_x, ϕ_y) was already measured before, then no new rows or elements are added to the weight matrix and RSSI vector so that they remain compact.

Given (s_x, s_y) and (t_x, t_y) we calculate the slope θ of the line. The length of the opposite side is $t_y - s_y$ and the length of the adjacent side is $t_x - s_x$, as depicted in Figure 14, so $\theta = \tan^{-1}\left(\frac{t_y - s_y}{t_x - s_x}\right)$. We know ϕ_x as it is equal to the width of the network, perhaps with a displacement if the origin of the network is not $(0, 0)$, and therefore we also know the length of $t_x - \phi_x$, which is how much we need to move in the horizontal direction for the snapped point.

Remaining is the calculation of ϕ_y. When two parallel lines are crossed by another line, then the corresponding angles are equal. This property is applied in Figure 14 in which the angle is also θ in the smaller triangle. Knowing this angle allows us to calculate $t_y - \phi_y$, the length of the opposite side, using $t_y - \phi_y = \tan(\theta) \cdot (t_x - \phi_x)$. Finally we obtain ϕ_y from this expression using $\phi_y = t_y - (t_y - \phi_y)$ and get the snapped point (ϕ_x, ϕ_y).

The `Snap_To_Boundary` class implements this transformation for all possible cases. The weight matrix may use this class when performing the transformation is desired.

6.2.5 RF sensor

The `zigbee` package contains components for working with the wireless sensors or *RF sensors*. We refer the reader to previous work [6] for details regarding the NTP, packet and TDMA scheduler classes and only describe the new structure for the RF sensor code in this work.

Extensibility is an important objective of the framework. Multiple brands and types of RF sensors exist and may be developed in the future. Since hardware generally evolves quickly, it is important that our framework not only allows adding support for other/newer brands or types of RF sensors, but also that updating existing implementations based on hardware changes is as easy as possible.

To achieve this goal, we split the RF sensor code into three layers of abstraction as outlined in Figure 13. The abstract `RF_Sensor` class defines the interface that all other RF sensor classes must implement and forms the first layer of abstraction. It contains properties and methods that all RF sensor classes share, such as getting the ID of the sensor, activating/deactivating the sensor and sending/receiving packets.

The second layer of abstraction is formed by the `RF_Sensor_Simulator` class and the abstract `RF_Sensor_Physical` class. The simulator class uses socket communication to simulate wireless communication and is primarily used during development, when working with physical RF sensors is impractical. In contrast, the class that contains code specific for physical RF sensors is abstract and forms the basis for the third layer of abstraction, which contains the implementation classes for each supported brand or type of physical RF sensor.

In addition to the existing support for the XBee Pro sensor, this work adds support for the Texas Instruments CC2530 sensor. For our purposes it provides more stable signal strength measurements than the XBee Pro sensor. The `RF_Sensor_Physical_XBee` and `RF_Sensor_Physical_Texas_Instruments` classes only contain code that is specific for these sensors as shared code is implemented in the abstract classes that they extend. By splitting shared code and hardware-specific code we increase both maintainability and extensibility.

6.2.6 Control panel

The *control panel* is the graphical user interface for the framework. Multiple views give the user control over practically all aspects of the steps required for mobile radio tomography. The control panel contains five views:

1. **Devices**. The devices view shows status information about the RF sensors on the vehicles and the RF sensor for the ground station.
2. **Planning**. The planning view makes it possible to create sequences of waypoints for the vehicles based on constraints and objectives, such as maximum network coverage, using an evolutionary algorithm [2].

3. **Reconstruction**. The reconstruction view exposes all tools for converting signal strength measurements from various data sources to two-dimensional images and applying regularization methods.

4. **Settings**. The settings view provides an organized interface to the settings file that contains all settings for each component of the framework.

5. **Waypoints**. The waypoints view allows the user to input custom waypoints and send them to the vehicles. It is also used to send the waypoints generated by the planning algorithm.

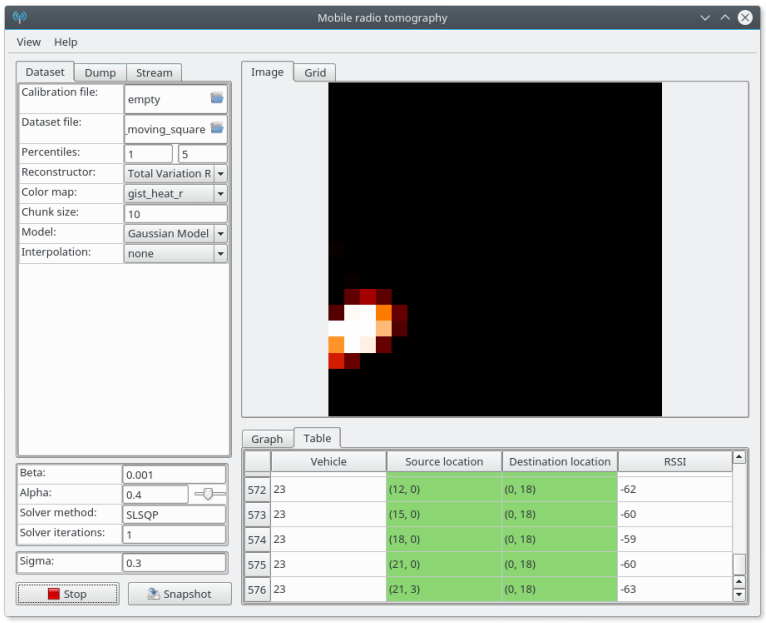


Figure 15: The reconstruction view in the control panel.

The reconstruction view is pictured in Figure 15. The tabs at the top left let the user pick the data source for the signal strength measurements. The parameters corresponding to the selected data source are presented in a table below the tab. For instance, the regularization method and weighting model may be chosen here. Specific parameters for the regularization method (such as the solver method for iterative optimization algorithms) or the weighting model (such as the number of singular values to keep for truncated singular value decomposition) may be changed in the other tables.

The real-time reconstruction and visualization process renders the image when enough data is available for the regularization algorithm. We limit the range of values for coloring the image to make the resulting images more aesthetically pleasing [3]. The grid view simplifies inspection of how well the processed measurements cover the network. The tabs below the image provide an overview of the processed measurements in the form of a table or a graph. The control panel initializes the reconstructor, the buffer, the coordinator and the RF sensor based on the input from the user.

6.2.7 Calibration

Calibration of signal strength measurements in a wireless network is essential for good quality reconstructions and visualizations. It is done by subtracting the RSSI value of the link in the empty network from the RSSI value of the link in the nonempty network. The rationale for this approach is that environmental noise, also present in the measurements of the empty network, is removed from the measurements of the nonempty network as good as possible so that only attenuation caused by objects inside the network remains. To keep our approach as general as possible, we assume no a priori knowledge about which links are used for the measurements. The calibration procedure must therefore perform a measurement for every possible link. As long as objects that we wish to detect are not in the network during calibration we may use any network, not just an empty one.

We construct a calibration mission for the vehicles that takes care of measuring every possible link in the empty network [2]. We use two vehicles that start at adjacent positions on a boundary of the network. At each moment in time, one vehicle is moving and one vehicle is stationary. The vehicles move in turns around the boundaries of the network until they reach each other, starting with the first vehicle. This scheme creates fan beams, ensuring that all links from the stationary vehicle to each other boundary position are measured.

The advantage of such a mission is that we obtain a calibration measurement for every possible link. The disadvantage is that measuring all possible links is a time-consuming process. However, in general, when the environment remains (largely) the same, calibration for a network needs to be performed only once. It is possible to speed up the calibration procedure by incorporating assumptions about the environment or symmetry in the network, but since we are developing a general approach, we do not consider this here.

7 Experiments

Mobile radio tomography is a new kind of radio tomography. Consequently, we wish to investigate the effectiveness of our approach. With the setup described in Section 7.1 we create multiple mobile radio tomography datasets for performing the experiments in Section 7.2.

7.1 Setup

Having a constant and clutter-free environment is desired to minimize environmental noise. Even though our framework is able to cope with this due to the regularization steps involved, for the sake of stability of the experiments we do not wish to introduce additional noise. Notice that we already have to work with environmental noise because of multipath interference caused by the room itself (walls, floor, et cetera). We create the setup in an empty experiment room at CWI Amsterdam.

Two vehicles, as described in Section 6.1, drive around on a grid with 20×20 intersections or, equivalently, 19×19 cells. The vehicles are only allowed to drive on the boundaries of the network, so we only create the outline of the grid. The outline is a combination of equal building blocks with one cell of 16.75 cm width and height printed onto solid square (cropped A3) paper. This allows us to easily increase or decrease the network size if necessary.

Figure 16 shows the complete setup that we use for our experiments, consisting of the room, the grid and the two vehicles. Outside the room is a human operator with a laptop and the ground station sensor for collecting the data from the measurements and running the reconstruction and visualization process.

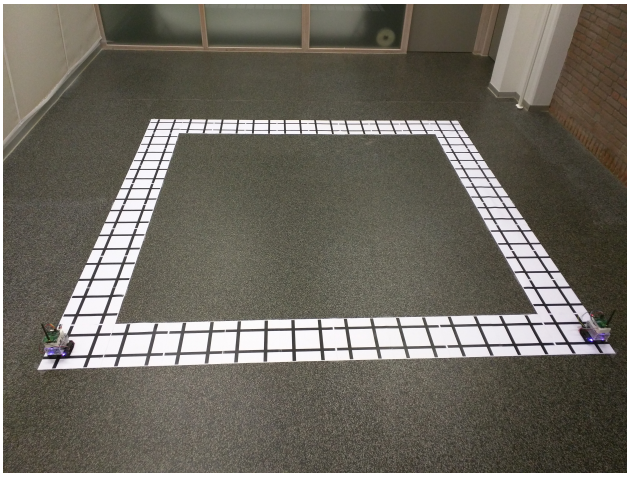


Figure 16: The room and setup for our experiments.

7.2 Datasets and results

The purpose of the experiments is to verify if our mobile radio tomography approach leads to good quality reconstructions of objects inside a sensor network. This implies that we study the effectiveness of all the steps required to go from raw signal strength measurements to two-dimensional images.

We create a calibration dataset for the network from Section 7.1 using the mission described in Section 6.2.7. This dataset is used to suppress environmental noise in further measurements. Moreover, we create six datasets with:

1. one person in the top left corner of the network;
2. one person in the top right corner of the network;
3. one person in the center of the network;
4. one person in the bottom left corner of the network;
5. one person in the bottom right corner of the network;
6. two persons, one in the middle of the left side and one in the bottom right corner of the network.

The mission used to obtain these six datasets is designed to cover the network completely using a series of straight lines, middle fan beams and corner fan beams [2]. The exact positions of the persons inside the network are depicted in Figure 17. The red squares indicate a person and the numbers inside the red squares correspond to the datasets above.

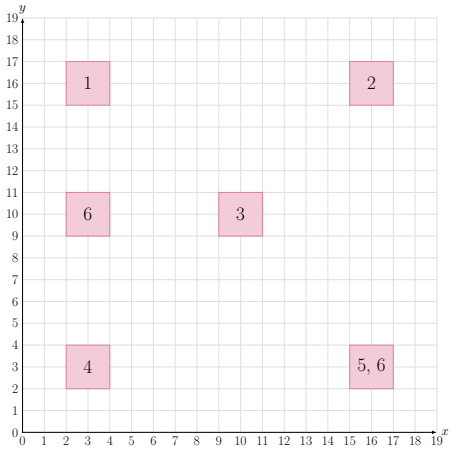


Figure 17: Exact positions of persons inside the network for the various datasets.

We purposely choose these specific datasets. We would like to determine if the reconstruction and visualization process yields accurate results for networks with both one person as well as with two persons inside. The latter is more difficult as more measurements are required to separate the two entities. If there is one person inside the network, we would like to determine if standing in the center or standing in corners has any influence on the reconstruction. This tells us if the coverage of the network is sufficient.

7.2.1 Stability and accuracy of reconstructor/model combinations

This experiment makes use of the sixth dataset, i.e., the dataset with two persons inside the network, and the calibration dataset. The objective of the experiment is to compare all combinations of regularization method (from Section 5) and weighting model (from Section 4) to determine which combination is most stable and yields the most accurate solutions.

The stability of the reconstruction and visualization process corresponds to the amount of change between successively rendered images. Measurements arrive gradually, so we would like updates to the image to be applied gradually as well. High instability indicates that the regularization method is not effective, i.e., that new measurements cause erratic changes in the image.

The instability of the process is measured using the pixel value differences between each pair of successive images (of the same size). The difference measure that we use is the mean absolute error.

Let r be the total number of pixels in each image and let the two images be ω_1 and ω_2. The images are expressed as functions that, given an index q, return the value of the q'th pixel of the image. The *mean absolute error MAE* is then calculated as in Equation 13:

$$MAE = \frac{1}{r} \sum_{q=0}^{r-1} |\omega_2(q) - \omega_1(q)| \tag{13}$$

We calculate the mean absolute error for each pair of successively rendered images. The sums, averages and standard deviations of these values are used to reason about the stability of the process.

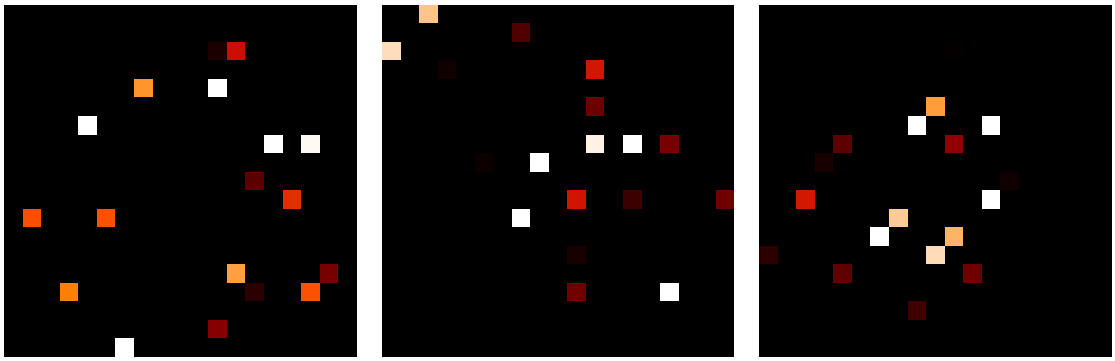
Furthermore, we provide observations regarding the accuracy of the solution, i.e., how well the positions from Figure 17 are visible in the reconstructed image. The color map that we use for the reconstructed images is depicted in Figure 18, in which black indicates low attenuation (for pixel value 0) and white indicates high attenuation (for pixel value 255).

Figure 18: Color map for the reconstructed images.

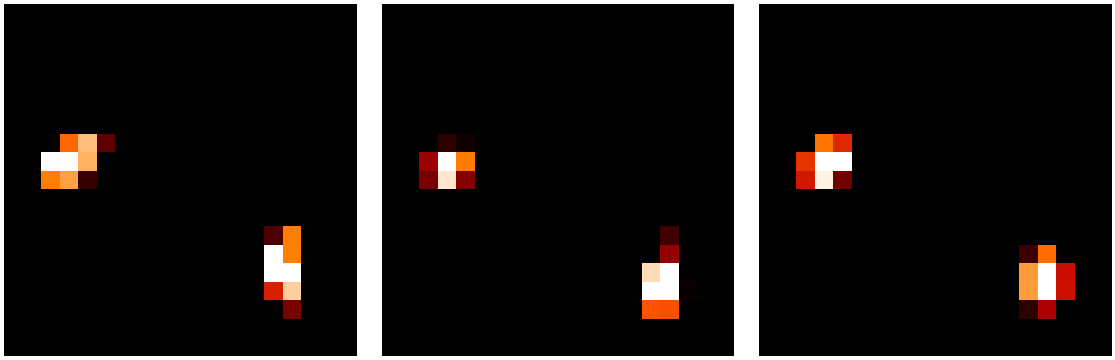
The instability values for all combinations of regularization method and weighting model are listed in Table 1, followed by the reconstructed images for all combinations in Figure 19. The parameters in the reconstruction view of the control panel are the default values as we determined those after careful tuning.

Combination	**Sum**	**Average**	**Standard deviation**
SVD, line model	792.75	5.18	1.24
SVD, ellipse model	798.32	5.22	1.17
SVD, Gaussian model	769.55	5.03	1.10
TSVD, line model	340.88	2.35	0.84
TSVD, ellipse model	297.81	2.05	0.82
TSVD, Gaussian model	338.37	2.33	0.86
TV, line model	166.76	1.09	1.52
TV, ellipse model	171.54	1.12	1.50
TV, Gaussian model	207.10	1.35	1.42
ME, line model	162.51	1.07	1.39
ME, ellipse model	166.23	1.09	1.41
ME, Gaussian model	207.10	1.35	1.42

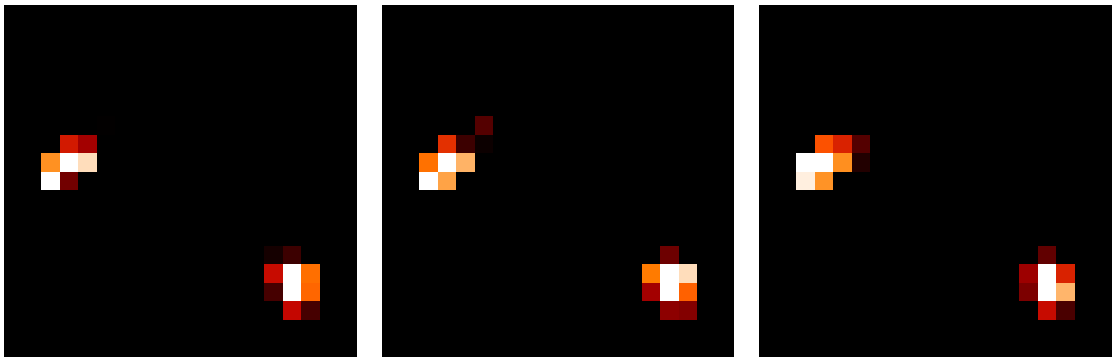
Table 1: Instability values (using the mean absolute error) for all combinations.



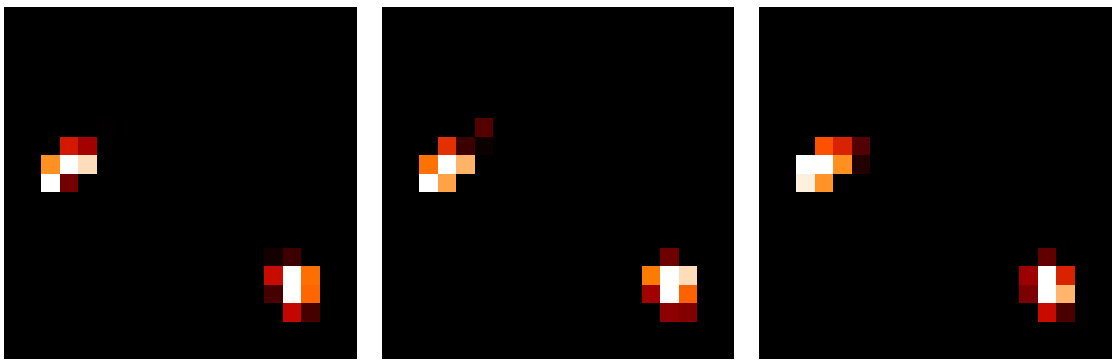
(a) SVD, line model (b) SVD, ellipse model (c) SVD, Gaussian model



(d) TSVD, line model (e) TSVD, ellipse model (f) TSVD, Gaussian model



(g) TV, line model (h) TV, ellipse model (i) TV, Gaussian model



(j) ME, line model (k) ME, ellipse model (l) ME, Gaussian model

Figure 19: Reconstructed images for all combinations.

The first observation is that singular value decomposition (without any regularization) indeed leads to major instability. Noise in the measurements is amplified to an extent in which the reconstructed images in Figures 19a, 19b and 19c do not provide any information about the positions of the persons inside the network, no matter which weighting model we use. The instability values in Table 1 and observations during the process confirm that the process is unstable. The sums and averages are more than twice as large as those for truncated singular value decomposition and over four times as large as those for the iterative regularization methods.

Moreover, we observe that truncated singular value decomposition, while being a relatively simple regularization method, provides a large improvement over regular singular value decomposition. The positions of the persons are clearly visible in Figures 19d, 19e and 19f. The ellipse model and the Gaussian model lead to similar results, whereas the line model leads to more noisy results when comparing the image to Figure 17. However, the intermediate images from the reconstruction and visualization process did show instabilities, which the instability values for truncated singular value decomposition indicate as well.

Interestingly, we observe that the reconstructed images for total variation minimization and maximum entropy minimization are practically the same. While the objectives for reducing variation are equal, the methods use two different variation measures. Close inspection shows that the images are not completely the same, but the differences between Figures 19g/19j, 19h/19k and 19i/19l are hardly visible to the naked eye. The instability values for both methods are practically equal as well, but maximum entropy minimization appears to stabilize slightly faster. The sums and averages of the iterative regularization methods are the lowest, but the standard deviations are higher. The minimization algorithm updates the solutions heavily at the beginning when only a small number of measurements is available, but stabilizes quickly as more measurements come in, which may explain the increase in standard deviation.

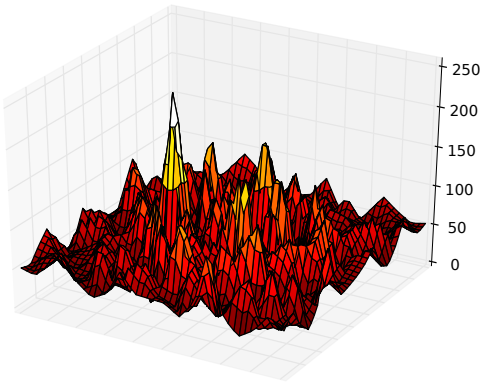
Finally, we notice that using the Gaussian model generally leads to higher instability values. A possible explanation for this is that the line and ellipse models assign either a zero value or a fixed non-zero value to each pixel, whereas the Gaussian model may assign any value. This could lead to more pixels getting a non-zero value and thus the probability of changes in successive images is increased.

In conclusion, we see that the iterative regularization methods (total variation minimization and maximum entropy minimization) provide the most stable and promising solutions. For the weighting models we see that the ellipse model and the Gaussian model yield the most accurate results. All combinations are suitable for real-time reconstruction.

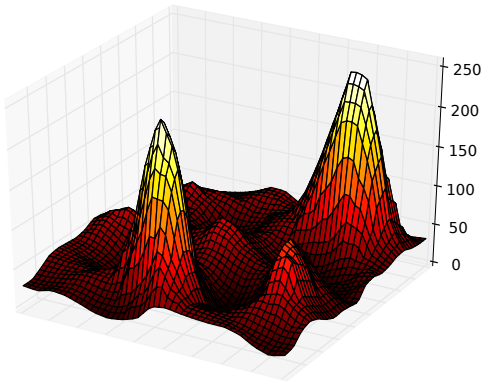
7.2.2 Smoothening effect of reconstructors

In addition to the requirement that the reconstructed images must provide a clear indication of where the persons are located inside the network, it is important that the images are smooth, in the sense that the differences between neighboring pixels are as small as possible. This implies that noise must be minimized as well.

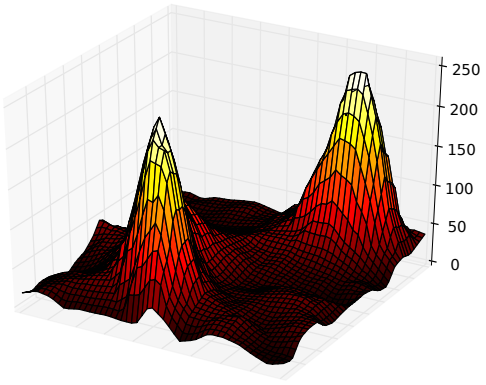
The objective of this second experiment is therefore to create 3D surface plots that give insight into the smoothening effect of the reconstructors. We use the same dataset, color map and reconstruction parameters as in the experiment of Section 7.2.1, but now we only use the Gaussian weighting model. The only difference between the surface plots is therefore the regularization method that we apply. The results of this experiment are visualized in Figure 20.



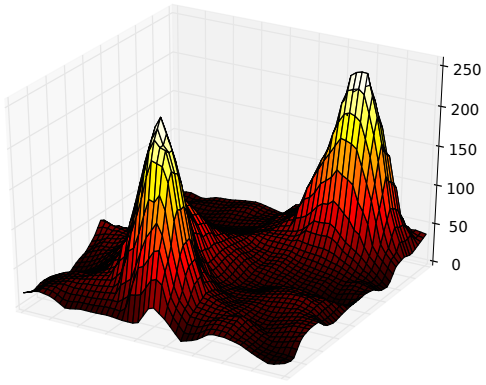
(a) SVD, Gaussian model



(b) TSVD, Gaussian model



(c) TV, Gaussian model



(d) ME, Gaussian model

Figure 20: 3D surface plots of the reconstructed images for all regularization methods.

The 3D surface plots in Figure 20 provide an alternative way to inspect the reconstructed images. The images are 8-bit grayscale and therefore 256 different shades of gray may be used for each pixel, which is what the values on the vertical axes represent. The ideal surface plot consists of a flat surface of pixel value 0 (black) with spikes of pixel value 255 (white) exactly at the locations of the persons.

Figure 20a clearly shows that singular value decomposition results in extremely noisy reconstructed images for the experiment of Section 7.2.1. The surface plot contains many small spikes because no smoothening is applied. This results in many differences between the values of neighboring pixels, which we observe as noise.

In contrast, the surface plot for truncated singular value decomposition in Figure 20b

is much smoother. It is also darker, which indicates that the values of pixels that correspond to empty locations in the network are low so that they do not stand out. The two large spikes do stand out, to clearly indicate the locations of the persons. However, the surface contains some small instabilities that lead to minor noise in the reconstructed images.

The surface plots for total variation minimization and maximum entropy minimization further improve the situation. Figures 20c and 20d show that the imperfections from truncated singular value decomposition are suppressed even more, leading to images with almost no observable noise.

7.2.3 Coverage of the network

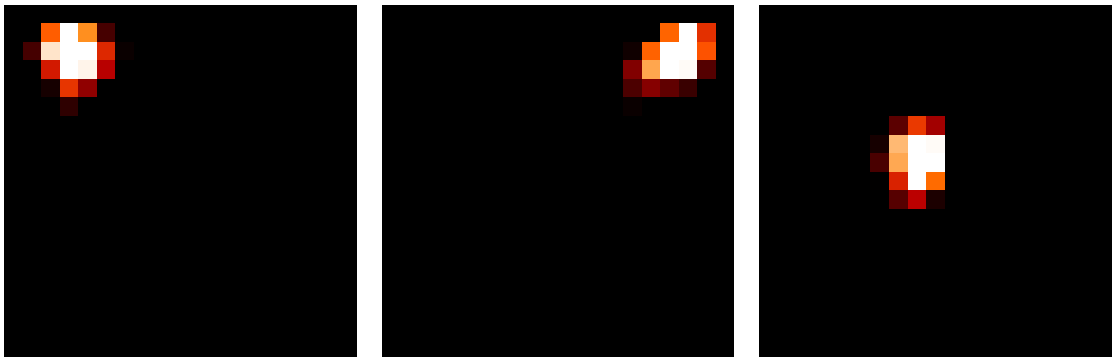
Coverage of the network plays a key role in improving the reconstruction quality for radio tomography in general. The missions for the vehicles in mobile radio tomography give us full control over the links that we measure and thus over the coverage that we obtain. It is therefore imperative that the missions are constructed in such a way that we get sufficient network coverage as quickly as possible. This trade-off between coverage and total execution time, combined with many other factors such as the number of vehicles to use, determines the mission that we should work with [2].

The objective is to determine if our current mission results in sufficient coverage. The mission, as introduced in Section 7.2, is carefully constructed to provide good coverage in a short amount of time. The straight lines are used to quickly get an initial guess of where the persons are located inside the network. Next, the middle fan beams intersect each pixel at least a second time to finetune the initial guess. We add the corner fan beams as the last step to make the reconstruction even more accurate, but only using straight lines and middle fan beams should already lead to reasonable images.

For this experiment we use datasets 1–5 from Section 7.2, which are the datasets with one person standing at different locations inside the network. We aim to find out if our mission creates enough coverage to detect the person at each location, i.e., that there are no dead spots where a person may be without being detected. For the images we use maximum entropy minimization combined with the Gaussian model, the default color map and the default parameters. The reconstructed images are shown in Figure 21.

First of all, we observe that the person is indeed clearly visible in each image. The accuracy is largely the same for all images. Certainly the accuracy could be improved with more measurements, but this also requires more time or more vehicles, which is the trade-off we mentioned earlier in this section.

In all cases, the persons are clearly visible after creating the straight lines and middle fan beams. The corner fan beams mostly increase the accuracy of the image by providing more measurements from different angles. This observation confirms that using only straight lines and middle fan beams may already produce satisfactory results if the corner fan beams are left out because of, for example, time constraints.



(a) Dataset 1 (top left) (b) Dataset 2 (top right) (c) Dataset 3 (center)

(d) Dataset 4 (bottom left) (e) Dataset 5 (bottom right)

Figure 21: Reconstructed images for datasets 1–5 from Section 7.2.

The construction of the mission may provide an explanation for any differences in detection speed. Reconstruction depends on intersections of pixels to determine their importance. If we create a fan beam from a certain location, then many links cross the pixels directly around that location, but pixels on the other side of the network are not intersected as much. This bias propagates to the reconstruction, so persons standing closer to the origin of the fan beam are likely to be detected sooner than persons standing farther away.

8 Conclusions

This master's thesis provides an in-depth analysis of reconstruction and visualization in the context of mobile radio tomography, a new form of radio tomography in which the statically positioned wireless sensors are replaced with dynamically positioned ones on moving unmanned vehicles. First of all, we discuss the process of converting a sequence of signal strength measurements to a two-dimensional image and the complications that arise due to noise, thereby pointing out the need for regularization.

Next, we discuss the available weighting models and regularization methods. Not only do we survey weighting models and regularization methods from regular radio tomography that are also useful for mobile radio tomography, but we present new ideas as well. We show that our new weighting model based on a Gaussian function is

a promising alternative for the existing ellipse model. Moreover, we apply maximum entropy minimization to the problem of radio tomography and show that its results are similar to total variation minimization.

The construction of dedicated hardware and the extension of the existing mobile radio tomography framework with reconstruction and visualization components form the next step. Implementations of all studied weighting models and regularization methods are provided. The advanced graphical user interface unifies interaction with the framework, simplifies performing mobile radio tomography and facilitates further research in this area.

Finally, we perform various experiments to determine the effectiveness of our approach. For this purpose we obtain multiple datasets by applying mobile radio tomography to a real-world network. The stability and accuracy of all possible combinations of weighting model and regularization method are studied, as well as the smoothening effects of the regularization methods and the coverage of the network. The framework creates a good quality reconstructed image for each dataset. The iterative regularization methods (total variation minimization and maximum entropy minimization) provide the most promising results when combined with the ellipse or Gaussian weighting model.

In conclusion, this work primarily shows that mobile radio tomography is feasible and paves the way for exciting new techniques and applications.

8.1 Future work

This work shows that mobile radio tomography has potential, but possibilities for improvement remain. Therefore, future work includes, but is certainly not limited to, the suggestions in this section.

Currently we use a grid for the vehicles, but future work could focus on evaluating other techniques for accurately tracking the location of a vehicle both indoors and outdoors to remove the dependency on the grid. With such techniques one could study the feasibility of using drones instead of rover cars and extending the reconstruction and visualization process to 3D.

Other interesting topics are determining the impact of different missions on the reconstruction quality, speeding up the calibration and efficiently scaling up our approach for large networks, large numbers of vehicles and higher resolution images. Making real-time reconstruction and visualization feasible in this case requires optimizing the existing regularization methods or implementing new techniques.

Finally, mobile radio tomography makes it possible to perform adaptive scanning. The idea is to scan the network quickly to determine the locations of objects. We then ignore empty parts of the network and focus on the regions in which the objects are located. The mission is adjusted based on these findings to make more links intersect pixels at locations where objects are likely to be. This scanning method makes more efficient use of time and allows us to visualize objects in more detail.

References

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

[2] L.S. Helwerda. "Mobile radio tomography: Autonomous vehicle planning for dynamic sensor positions". Master thesis, Leiden University. 2016.

[3] A. Milburn. "Algorithms and models for radio tomographic imaging". Bachelor thesis, Leiden University. 2014.
URL: http://liacs.leidenuniv.nl/assets/Bachelorscripties/Inf-Studiejaar-2013-2014/2013-2014AlyssaMilburn.pdf.

[4] J. Wilson and N. Patwari. "Radio tomographic imaging with wireless networks". *IEEE Transactions on Mobile Computing* 9 (2010), pp. 621–632.
DOI: 10.1109/TMC.2009.174.

[5] T. van der Meij. "Constructing an open-source toolchain and investigating sensor properties for radio tomography". Bachelor thesis, Leiden University. 2014.
URL: http://liacs.leidenuniv.nl/assets/Bachelorscripties/Inf-Studiejaar-2013-2014/2013-2014TimvanderMeij.pdf.

[6] T. van der Meij. "Mobile radio tomography: Constructing an open-source framework with wireless communication components". Research project report, Leiden University. 2016.

[7] L.S. Helwerda. "Mobile radio tomography: Object detection using autonomous unmanned vehicles". Research project report, Leiden University. 2016.

[8] T. Hult and A. Mohammed. "Assessment of multipath propagation for a 2.4 GHz short-range wireless communication system". *IEEE 65th Vehicular Technology Conference*. 2007, pp. 544–548.
DOI: 10.1109/VETECS.2007.123.

[9] J.B. Andersen, T.S. Rappaport, and S. Yoshida. "Propagation measurements and models for wireless communications channels". *IEEE Communications Magazine* 33 (1995), pp. 42–49.
DOI: 10.1109/35.339880.

[10] A. Björck. "Numerical methods for least squares problems". Society for Industrial and Applied Mathematics (SIAM) Philadelphia, 1996.

[11] I. Dokmanić, M. Kolundžija, and M. Vetterli. "Beyond Moore-Penrose: Sparse pseudoinverse". *IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 6526–6530.
DOI: 10.1109/ICASSP.2013.6638923.

[12] H. Bast. "An existence proof for the singular value decomposition". Lecture notes of Advanced Topics in Information Retrieval at Max-Planck-Institut für Informatik. 2004, pp. 1–3.
URL: https://people.mpi-inf.mpg.de/~bast/ir-seminar-ws04/lecture2.pdf.

[13] G. Golub and W. Kahan. “Calculating the singular values and pseudo-inverse of a matrix”. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis* 2 (1965), pp. 205–224.
DOI: 10.1137/0702016.

[14] C.E. Shannon. “A mathematical theory of communication”. *The Bell System Technical Journal* 27 (1948), pp. 379–423.

[15] A.C. Bovik. “Handbook of image and video processing”. Academic Press, 2005.

[16] Polulu Robotics & Electronics. “Zumo Robot for Arduino, v1.2 (assembled with 75:1 HP motors)”. July 2014.
URL: https://www.pololu.com/product/2510.

[17] Arduino. “Arduino Uno board”. Feb. 2012.
URL: https://www.arduino.cc/en/Main/ArduinoBoardUno.

[18] Raspberry Pi Foundation. “Raspberry Pi 2 model B”. Feb. 2015.
URL: https://www.raspberrypi.org/products/raspberry-pi-2-model-b.

[19] Texas Instruments. “CC2530 development kit”. Apr. 2010.
URL: http://www.ti.com/tool/cc2530dk.

[20] Ecma International. “The JSON data interchange format”. Standard ECMA-404. 2013.
URL: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf.

[21] J. Wilson and N. Patwari. “Radio tomographic imaging data set”. 2010.
URL: http://span.ece.utah.edu/rti-data-set.

The open-source mobile radio tomography framework is available at https://github.com/timvandermeij/mobile-radio-tomography.