Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

Position Tracking of Magnetic Dipoles Using a Modular AMR Sensor Grid

Master's Thesis
submitted to the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Dustin Schneider

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Dr. Martin Schmitz

Registration date: 19.05.2023
Submission date: 20.11.2023

Zentrales Prüfungsamt/Central Examination Office

RWTH AACHEN UNIVERSITY

Eidesstattliche Versicherung
Declaration of Academic Integrity

Schneider, Dustin	334766
Name, Vorname/Last Name, First Name	Matrikelnummer (freiwillige Angabe) Student ID Number (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende ~~Arbeit~~/~~Bachelorarbeit~~/ Masterarbeit* mit dem Titel
I hereby declare under penalty of perjury that I have completed the present paper/bachelor's thesis/master's thesis* entitled

Position Tracking of Magnetic Dipoles Using a Modular AMR Sensor Grid

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
independently and without unauthorized assistance from third parties (in particular academic ghostwriting). I have not used any other sources or aids than those indicated. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. I have not previously submitted this work, either in the same or a similar form to an examination body.

Mönchengladbach, 20.11.2023	
Ort, Datum/City, Date	Unterschrift/Signature

*Nichtzutreffendes bitte streichen/Please delete as appropriate

Belehrung:
Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
§ 156 StGB (German Criminal Code): False Unsworn Declarations
Whosoever before a public authority competent to administer unsworn declarations (including Declarations of Academic Integrity) falsely submits such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment for a term not exceeding three years or to a fine.
§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.
§ 161 StGB (German Criminal Code): False Unsworn Declarations Due to Negligence
(1) If an individual commits one of the offenses listed in §§ 154 to 156 due to negligence, they are liable to imprisonment for a term not exceeding one year or to a fine.
(2) The offender shall be exempt from liability if they correct their false testimony in time. The provisions of § 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Mönchengladbach, 20.11.2023	
Ort, Datum/City, Date	Unterschrift/Signature

Contents

List of Figures

List of Tables

Abstract

Several object tracking systems have been proposed for a variety of purposes, including rotational speed sensors for fabrication machines, avalanche beacons that can be used to track down buried skiers, wearable motion capturing systems, wireless capsule endoscopy, and many more. In this work, we devise a permanent magnet tracking system that is optimized for human computer interaction (HCI) purposes. The system is based on anisotropic magnetoresistive (AMR) sensors on tiny modules that can be composed to form networks of 2 to over 100 sensors, depending on the requirements of the application. By employing the I3C bus, we achieve low latencies and high sampling rates of up to 1kHz. In addition to the sensors, only a controller is needed, making the hardware sufficiently compact to be installable in a variety of existing environments. For the localization, we formulate a least-squares problem that we base on the ideal dipole B-field equations as an approximation to the field of a real magnet. We test gradient descent, Gauss-Newton and Levenberg-Marquardt to solve the resulting over-determined non-linear equation system. To improve their performance, we combine these methods with a line search, for which we evaluate three different methods: Grid Search, Golden Section Search and Newton-Raphson. By running several simulations, we found that Gauss-Newton, solved with the Cholesky decomposition and combined with a line search consisting of a coarse Grid Search followed by a Golden Section Search, leads to the fastest expected computation time of just $212\mu s$ for a position, orientation and magnetization estimation. For initial positions that are within 3cm of the actual position, the algorithm is even faster, needing only less than $100\mu s$ for a localization. This is at least two orders of magnitude faster than the commonly used approach based on Levenberg-Marquardt as a trust-region method, which makes it better suited to run in embedded systems.

Überblick

Viele Positions- und Orientierungsbestimmungssysteme existieren für eine Fülle von Anwendungen, angefangen bei Winkel- und Drehzahlmesssystemen in Fertigungsmaschinen, über Lawinenverschüttetensuchgeräte mit denen über größere Reichweiten verschüttete Menschen geortet werden können, bis hin zu tragbaren Bewegungserfassungssystemen. In dieser Arbeit konzipieren wir ein System für die Lokalisierung von Permanentmagneten, das für Anwendungen im Bereich der Mensch-Computer-Interaktion optimiert ist. Das System basiert auf anisotropen magnetoresistiven Sensoren, die als kleine Module je nach Anwendungsfall zu Netzwerken von 2 bis über 100 Sensoren kombiniert werden können. Durch Verwendung des I3C Busses erreicht unser System niedrige Latenzen und hohe Messraten von bis zu 1 kHz. Neben den Sensoren ist sonst nur ein Controller vonnöten, wodurch die Hardware hinreichend platzsparend ist, um in einer Vielzahl von bestehenden Umgebungen eingesetzt werden zu können. Zur Lokalisierung stellen wir eine quadratische Verlustfunktion auf Basis der B-Feld Gleichung magnetischer Dipole auf, die uns als Näherung für das tatsächliche Feld eines Permanentmagneten dient. Zur Lösung des resultierenden überbestimmten, nicht linearen Gleichungssystems testen wir die Gradienten-, Gauß-Newton- und Levenberg-Marquardt-Verfahren. Zur Verbesserung kombinieren wir diese Methoden zusätzlich mit einer Liniensuche, für die wir drei weitere Methoden evaluieren: die Rastersuche, das Verfahren des Goldenen Schnittes und das Newton-Raphson-Verfahren. Unsere Simulationen ergeben, dass das Gauß-Newton-Verfahren, gelöst mithilfe der Cholesky-Zerlegung und in Kombination mit einer Liniensuche basierend auf einer Rastersuche gefolgt vom Goldenen-Schnitt-Verfahren, die Position, Orientierung und Magnetisierung eines Dipols in durchschnittlich nur $212\,\mu s$ bestimmen kann. Wenn die geschätzte Startposition nur bis zu 3 cm von der echten Position des Dipols entfernt ist, vollendet das Verfahren Lokalisierungen sogar in unter $100\,\mu s$. Das ist mindstens zwei Größenordnungen schneller als die verbreitete Lösung mittels Levenberg-Marquardt als Trust-Region-Verfahren und macht die Lokalisierungsprozedur geeigneter für die Ausführung in eingebetteten Systemen.

Acknowledgements

I would like to thank Prof. Dr. Jan Borchers and Dr. Martin Schmitz for examining my thesis. Also, I want to thank my supervisor René Schäfer for his time throughout my thesis.

I would especially like to thank my family and friends who have supported me during my Master's thesis, in particular Jani Taxidis and Laura Drescher-Manaa.

Conventions

Variables

Name	Structure	Description	Eq.
k	$\mathbb{N}$	Generic counter	
n	$\mathbb{N}_{\geq 2}$	Number of sensors	
r	$\mathbb{R}^3$	Position relative to magnet	
p	$\mathbb{R}^3$	Magnet position	
m	$\mathbb{R}^3$	Mag. orientation and strength/dipole moment	
$\hat{p}$	$\mathbb{R}^3$	Position update vector	
$\hat{m}$	$\mathbb{R}^3$	Dipole moment update vector	
P	$\mathbb{R}^3$	$P = p - h\hat{p}$	
M	$\mathbb{R}^3$	$M = m - h\hat{m}$	
h	$\mathbb{R}$	Update scale for iterative methods	
p_{s_k}	$\mathbb{R}^3$	Position of sensor k	
B_{s_k}	$\mathbb{R}^3$	B-field measurement of sensor k	
B_s	$\mathbb{R}^{3n}$	Column vector of all sensor measurements	4.4
p_t	$\mathbb{R}^3$	True magnet position in simulations	
m_t	$\mathbb{R}^3$	True dipole moment in simulations	
p_i	$\mathbb{R}^3$	Initial magnet position in simulations	
m_i	$\mathbb{R}^3$	Initial dipole moment in simulations	

Functions

Name	Structure	Description	Eq.
$B_i(p, m)$	$\mathbb{R}^6 \to \mathbb{R}^3$	B-field at the position of sensor k	4.2
$B(p, m)$	$\mathbb{R}^6 \to \mathbb{R}^{3n}$	B-fields at all sensors	4.3
$R(p, m)$	$\mathbb{R}^6 \to \mathbb{R}^{3n}$	Residual vector	4.5
$L(p, m)$	$\mathbb{R}^6 \to \mathbb{R}$	Loss function	4.6
$J(p, m)$	$\mathbb{R}^6 \to \mathbb{R}^{3n \times 6}$	Jacobian of B	4.20
$d(a, b)$	$\mathbb{R}^6 \to \mathbb{R}$	Euclidean distance between two points	

In this work, the phrase *localization* is used to refer to the estimation of both the position and the orientation of a marker. *Markers* are localizable entities that can be attached to or integrated into objects to make them trackable.

Chapter 1

Introduction

The ability to estimate the position or orientation of magnetic dipoles is essential for many applications, including rotational speed sensors in fabrication machines, avalanche beacons that can be used to track down buried skiers [Pinies et al., 2006] and tracking medical apparatus inside of patients [Hu et al., 2010]. Accordingly, research into this problem has been conducted by scientists of several disciplines: Attempts by McFee and Das [1981] to localize dipoles by solving the B-field equations, for the purpose of detecting buried ordnances date back to 1981. In the field of human computer interaction (HCI), Liang et al. [2012, 2013, 2015] present several possible applications involving different objects with embedded permanent magnets and a dense sensor grid built out of 1D Hall sensors. Steurer and Srivastava [2003] use a large grid of Hall switches to build a smart table, which is capable of locating tiles that have been marked with magnetic tape. Their approach is intended to be used for locating tokens in board games. Hook et al. [2009] present a system based on a grid of pickup coils that can detect ferromagnetic objects in its vicinity. Combined with a ferrofluid bladder, this can be used to create a moldable touchpad, which might be useful for sculpting applications. Optical tracking is a common alternative to magnetic tracking, but it is only employable in applications that offer sufficient space for cameras, and in which the markers are guaranteed to stay in line of sight, making it unapplicable for most aforementioned use cases.

The localization approaches of the works that we cover can be divided into two categories: *graphical* and *equation-system-based*. The HCI systems tend to employ large grids of sensors, compile the measurements into images and perform the position and orientation estimation of the markers by graphical means [Steurer and Srivastava, 2003, Liang et al., 2012, 2013, 2015] or not at all [Hook et al., 2009]. All other approaches use the equations, that describe B-fields, to set up approximate linear equation systems or non-linear minimization problems, and solve for the dipole position and orientation [McFee and Das, 1981, Schlageter et al., 2001, Hu et al., 2005, 2007, 2008, Hashi et al., 2011, Lu et al., 2019].

Most approaches use passive markers [Hashi et al., 2011], which are usually permanent magnets [Steurer and Srivastava, 2003, Hu et al., 2006, 2010, Han et al., 2009, Liang et al., 2012, 2013, 2015], so that the markers can move freely. Using the sensors as active markers is also possible, especially for wearable motion capturing devices, where the cable connections do not get in the way [Fernandez et al., 2020]. For human motion tracking, the B-field based approach can also be combined with inertial measurement units (IMU) as additional sources of data [Pinies et al., 2006, Fernandez et al., 2020].

Some related approaches use coils to detect the B-fields [Hook et al., 2009], especially those that need to cover large distances [McFee and Das, 1981, Pinies et al., 2006] or have sufficient space available in the targeted application [Hashi et al., 2011]. Since coils can only detect changes in magnetic flux, these systems usually rely on alternating B-fields [Pinies et al., 2006, Hashi et al., 2011] or coils as fluxgate sensors [McFee and Das, 1981] to be able to localize dipoles that are not moving. Alternating fields also enable the identification of different markers via their frequency, but requires active markers or external excitation sources [Hashi et al., 2011, Fernandez et al., 2020], which require a significant amount of energy and space. This is why systems that aim for a high degree of integration favor miniaturizable sensors, such as Hall [Schlageter et al., 2001, Liang et al., 2012, 2013, 2015] or anisotropic magnetoresistance (AMR) sensors [Hu et al., 2006, 2010, Han et al., 2009].

Even though a lot of research has been conducted around magnetic tracking, all previously mentioned HCI systems use similar methods and come with the same drawbacks: the requirement to arrange sensors in a dense Cartesian grid, and the accordingly high cost, low sensing ranges and limitations in the orientation estimation. To alleviate this, we attempt to devise an affordable real-time permanent magnet tracking system that relies on fewer, fully modular and more sensitive sensors, is easy to set up and use, and has no limitations in the position or orientation estimations. This is something that the non-HCI systems already offer, but those usually come with high latencies and low localization rates, or large space and power requirements. We aim to create a tracking system that localizes the marker at high rates with low latencies by taking a closer look at some numerical methods and by making appropriate design choices for the hardware. We also try to minimize the space and power requirements, to make our system deployable in a variety of scenarios. The target application is, but should not be limited to, 3D gesture input.

In Chapter 2, we take a closer look at the related systems, with a general overview in Table 2.1. In Chapter 3, we discuss the development hardware, which we designed around the AMR sensor that was chosen in a previous work by Drescher-Manaa [2022]. Chapter 4 deals with the math behind our localization procedure and describes the methods that we evaluate. The results of several simulations for Grid Search, Golden Section Search, Newton-Raphson, Gradient Descent, Gauss-Newton and Levenberg-Marquardt are presented and analyzed in Chapter 5. In Chapter 6, we compare our progress to the related works. Finally, we give a summary and outline the next steps that need to be taken for further development in Chapter 7.

Chapter 2

Related work

In the following, we will present six related works in detail, all of which cover complete marker localization systems. The first three of those perform the localization by graphical means, while the last three construct and solve non-linear least-squares problems based on the equation that describes the B-field of an ideal dipole. For an overview of the related systems, refer to Table 2.1.

GaussSense, GaussBits and GaussStarter [Liang et al., 2012, 2013, 2015]

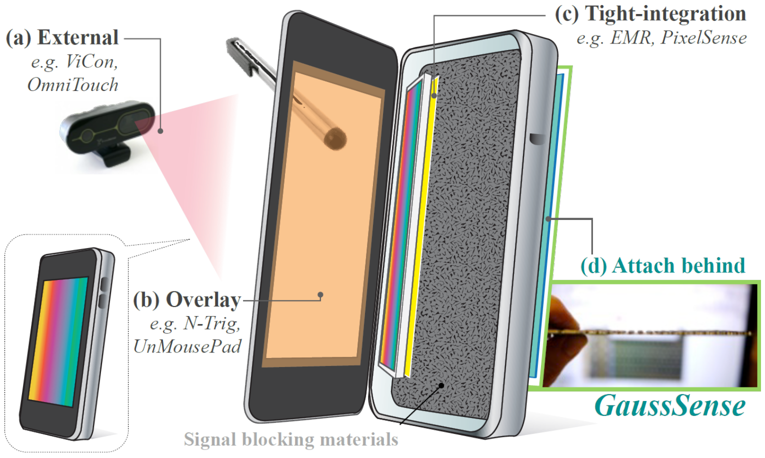


Figure 2.1: Intended application for GaussSense [Liang et al., 2012] and alternative approaches for stylus tracking: (a) visual tracking, (b) screen overlays, (c) integration into the device and (d) attachment of GaussSense to the back of the device.

In ”GaussSense”, Liang et al. [2012] retrofit devices that have capacitive touchscreens with stylus input capability by attaching a 2D grid of 1D Hall sensors to the back of the device, as indicated in Figure 2.1.

A dense grid of Hall sensors is used in conjunction with permanent magnets as markers.

The stylus is equipped with a permanent magnet whose field is measured to determine its tilt and pressure on the screen. The sensor array is made up of 12×16 Hall sensors with part number WSH138 in SOT-23 packages on a 60mm×80mm printed circuit board (PCB). The analog output voltages of the sensors are switched by 16:1 multiplexers, sampled and then transferred to a PC over USB by a microcontroller. The circuit board is shown in Figure 2.2.

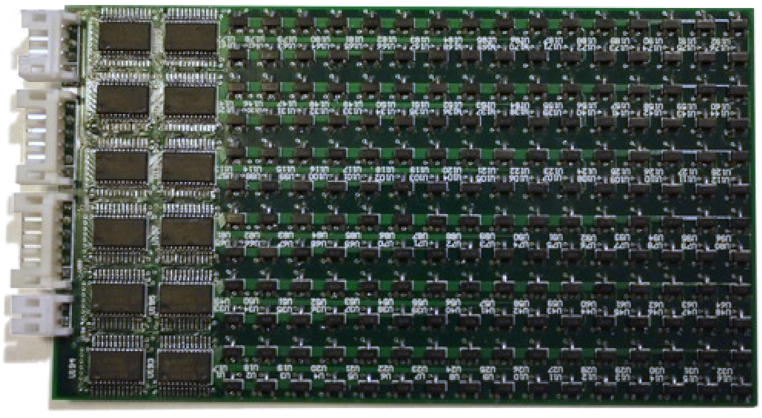


Figure 2.2: 12×16 1D Hall sensor grid by Liang et al. [2012]

The sampled values are corrected and transformed into an intensity map.

For the localization of the magnet, the first preprocessing step is offset correction. The offsets in the measurements are calculated once at startup with no magnet present as the deviation of each sensor value from the mean value of the array and are subsequently subtracted from all measurements. Secondly, a non-linear transformation is applied to each individual measurement to obtain intensity values that correspond more closely to the distance between the magnet and the sensor grid. The intensity values are then assembled into an image and upsampled for a finer resolution. For the localization, lower intensities are discarded until the resulting values no longer touch the boundary of the image. The distance between the centroid of the remaining values and the stylus tip position, as determined by the touch screen, is then used to determine the tilt of the stylus, while the maximum intensity value is interpreted as the stylus pressure (Figure 2.3).

The tip position is taken from the touchscreen, the pressure and tilt from the intensity map and the tip position.

GaussSense is improved in GaussBits to support multi-object tracking and larger detection distances.

In their work named ”GaussBits”, Liang et al. [2013] build

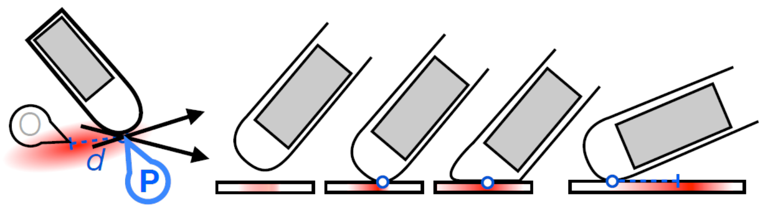


Figure 2.3: Schematic diagram of the stylus tracking with stylus tip P, B-Field centroid O and distance between tip position and centroid d [Liang et al., 2012]

upon their earlier work, GaussSense, and explore possibilities to enhance graphical user interfaces with trackable real-world objects. Improvements include allowing detection of multiple objects and increased distance from the sensor network at which hovering objects can be localized. The goal is to extend the limited 2D user interface design space of touch displays to the 3D near-surface space. One example named GaussClock is shown in Figure 2.4.

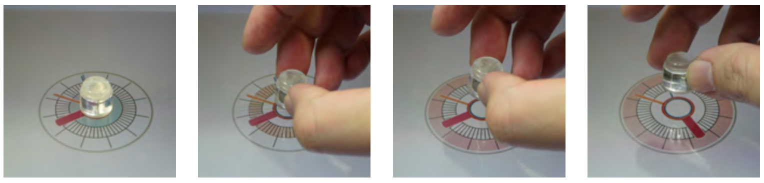


Figure 2.4: GaussClock: Multi stage rotary control, Liang et al. [2013]. Lifting the rotary knob shifts the control to the outer dials.

As in GaussSense, objects are being tracked using the static magnetic field that is emitted by a permanent magnet. The sensor technology remains the same, but four of the boards (Figure 2.2) have been combined to extend the measured area to cover larger screens. In contrast to the predecessor, which only used the N-component of the B-field in z-direction, the whole measurement range is used in GaussBits to allow for a more versatile localization procedure.

Compared to GaussSense, the sensor area and number has been quadrupled and the B-field range was extended to include the S-pole.

The position and orientation of the magnets are again estimated via cluster centroid positions and maximum intensities. The intensity is used to estimate the distance of the tracked object from the sensor grid, while a line drawn between the centroids of two opposite pole activation clusters is used to determine the roll angle. The center point of that line is assumed to be the position of the magnet. For tiltable objects, the centroids are calculated before and after non-maximum suppression and the line is drawn between these two centroids of the same cluster. The tilt angle is

The localization is based on clustering and graphical methods.

then deduced from the length of the line. Examples of the intensity maps are given in Figure 2.5.

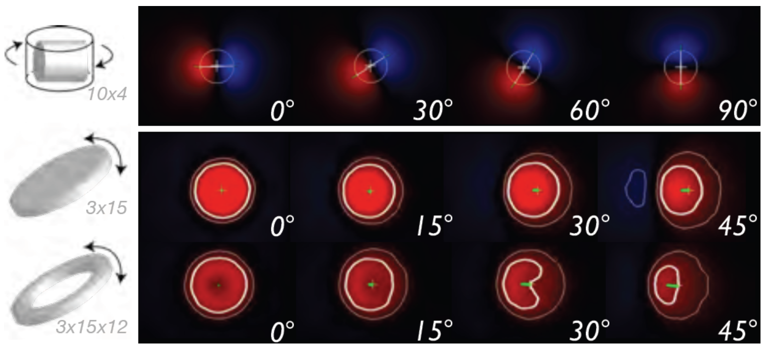


Figure 2.5: Intensity maps for magnets at different angles [Liang et al., 2013]. The first row depicts the rotation of a magnet around the roll axis. A line drawn between the centroids of the north and south clusters is used to derive the rotation angle. The second and third rows show intensity maps of two different magnets at different tilt angles. The length of the green line is used to determine the tilt angle. The line is drawn between two centroids of the same cluster, that had two different thresholds applied for discarding samples.

To make prototyping with this system easier, Liang et al. [2015] published "GaussStarter", in which the authors divide the sensor array into 4×4 grids (Figure 2.6). These can then be combined on a breadboard to construct sensor grids of varying sizes (Figure 2.7).

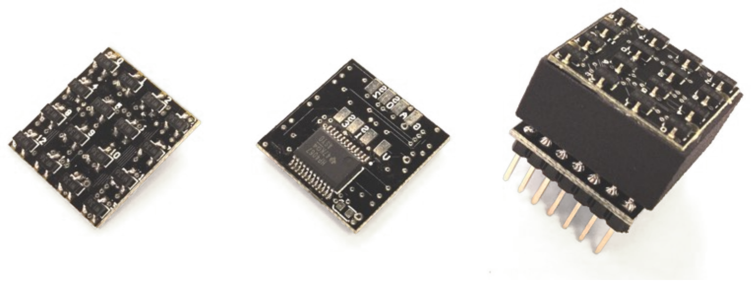
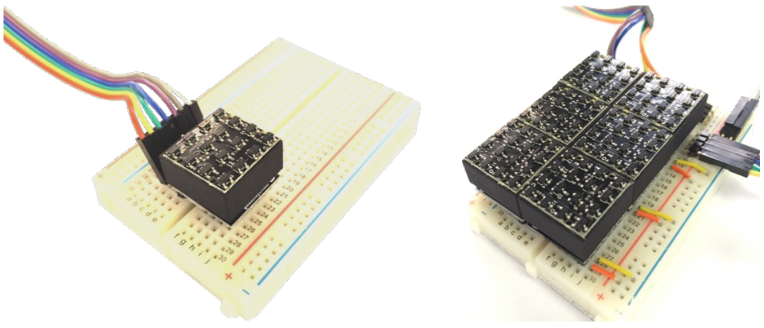


Figure 2.6: Hall sensor grid module by Liang et al. [2015], containing sixteen WSH136 sensors and one multiplexer

The power requirement of the prototype is very large, which is problematic in portable applications.

Assessment: Each of the individual Hall sensors typically draws over 3mA of supply current at 5V. For the 768 sensors, this adds up to at least 2.3A, or a power of 11.5W, not accounting for the multiplexers and microcontrollers. This exceeds the 2.5W power delivery capabilities of USB 2.0 by far [Com, 2000] and rivals the internal power consumption



(a) Single module **(b)** Hall sensor grid

Figure 2.7: Modular Hall sensor grid on a breadbord for prototyping permanent magnet tracking systems of variable sizes by Liang et al. [2015]

of connected mobile devices, draining their internal batteries and making the system less portable. This can be fixed easily, however, by multiplexing the power supply of the sensors, only enabling those that are currently being sampled. As such, this is a problem of the prototype, not a problem of the working principle itself.

The sampling resolution is low, resulting in a short maximum tracking distance.

The output voltage of the WSH138 sensor is roughly linear for fields from -200 to 200 Gauss and is sampled with 9-Bit precision. Whether this sampling depth covers just the linear range of the sensor or the full range is not mentioned by the authors. This leads to a resolution of 1356mG/lsb in the worst and 781mG/lsb in the best case. For most locations inside of the measurement volume, the measured B-field values will be rather low in intensity, as the B-field intensities decrease by the inverse of the distance cubed [Liang et al., 2012]. In conjunction with the low resolution, this leads to a very limited distance at which the localization can be performed successfully. In addition, the graphical algorithm does not make use of the extreme redundancy provided by the large amount of sensors and failed in the tests of the authors at distances between 17mm to 44mm, depending on the magnet. This could be improved on the electrical side by using higher resolution A/D converters, preamplifiers and filters. The output capacitors that are recommended in the WSH138 datasheet are missing completely, additionally worsening the signal-to-noise ratio.

The full potential of the sensors is not harnessed by the electrical system.

The orientation estimation has several limitations.

Additional downsides of the proposed algorithm are that firstly, a calibration is necessary for each magnet shape and strength for an accurate tilt detection, secondly, only 360° of the roll angle or up to 45° tilt can be detected due to ambiguities in the intensity map, thirdly, the intensity map upsampling is costly and does not increase the information content of the data and lastly, the tracked objects must not get too close to each other, or they will either stick to each other or be grouped into one object by the algorithm, depending on the polarity of their fields.

The graphical algorithm handles oddly shaped magnets very well.

An advantage of the algorithm compared to all other related works presented hereafter is that oddly shaped magnets are not only unproblematic, but even beneficial. Other localization techniques that are instead based on the dipole equations usually assume the magnetic dipoles to be point sources without any volume, and in these, deviations from that assumption are one of the main sources of error. In GaussBits on the other hand, unique field shapes are used for identification, as seen in Figure 2.8.

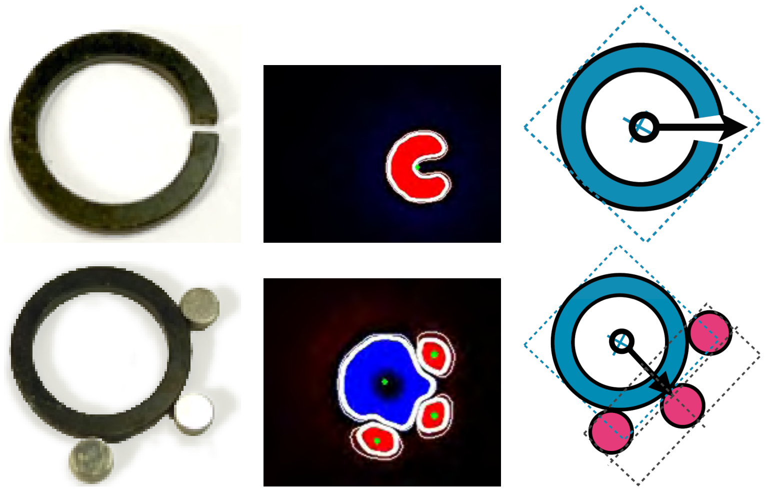


Figure 2.8: Magnet configurations that are identifiable by shape [Liang et al., 2013]. Each row shows a photo of the magnet, the corresponding intensity map and a schematic diagram of the magnet with an arrow marking the roll orientation.

The large Hall sensor network is very expensive.

Lastly, another issue of the system is cost. At the time of this writing, the most inexpensive Hall sensors cost about 10 cents a piece when bought on a whole reel at a distributor. Thus, each such Hall sensor array costs at least 80€ in parts, making this approach impractical compared to other related works.

System Design of Smart Table [Steurer and Srivastava, 2003]

In an earlier work, Steurer and Srivastava [2003] describe a similar system to GaussSense, in which object positions and identities are tracked on the surface of a table. Two approaches are tested separately, with the second one being of interest here. The first is based on small metal contact pads on the surface of the table, the second one is based on 9600 Hall switches distributed over 24 PCBs that can be mounted underneath the table. The authors are aware of the prohibitively high cost of such a system, which is likely the reason for them preferring Hall switches over linear Hall sensors. The tracked objects are equipped with multiple pieces of magnetic tape in different shapes on their bottom side. A triangle, formed by three bits of tape in the corners, is used for localization. A second pattern is used for identification, where the presence or absence of each bit of tape is recorded as one bit of information. The localization algorithm is somewhat similar to GaussBits: An activation map is sampled over the whole surface and, based on that, the positions of the magnetic pads are determined by finding clusters of up to four activations and averaging their positions. After that, the pads are assigned to objects and the object positions and orientations are deduced from the pad positions that form the triangle.

Hall switches are used as sensors.

Trackable objects are marked with several pieces of magnetic tape in different patterns.

Assessment: As mentioned before, this system is very expensive, as it consists of a high number of parts. Since Hall switches only produce one bit of data per piece, the position estimation is limited to two dimensions due to the low information content of the data. In addition, objects need to be rather large, with diameters in the order of 10cm, for their pads to be placed at a sufficiently large distance, so that a satisfactory angular precision can be achieved by the localization algorithm.

The smart table is very expensive and the measurement data is very limited.

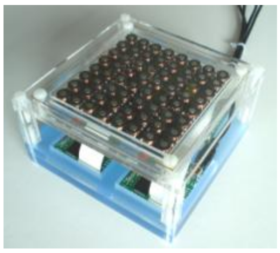
The tracked objects must be large.

A Reconfigurable Ferromagnetic Input Device [Hook et al., 2009]

Another intensity map based approach has been published by Hook et al. [2009], with the goal of creating a versatile input device for multiple ferromagnetic objects (Figure 2.9).

Coils with integrated magnets are used as sensors, so that trackable objects only need to be ferromagnetic.

Contrary to the markers that are used in other related works, the tracked objects do not emit measurable fields


(a) The detector

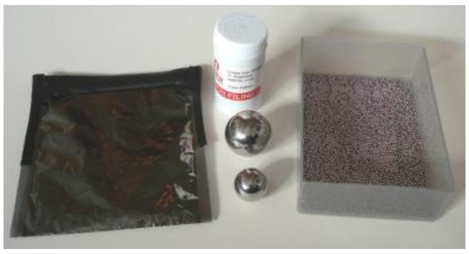
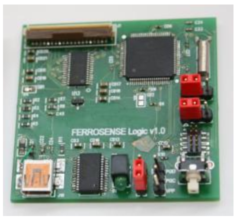

(b) Detectable ferromagnetic objects

Figure 2.9: *A reconfigurable ferromagnetic input device* by Hook et al. [2009]. The detector comprises 64 pickup coils in a 8×8 grid, covering an area of 10cm×10cm and can detect changes in arbitrarily-shaped, ferromagnetic objects.

on their own, so this is done externally with permanent magnets. Additionally, coils are used instead of traditional sensors to pick up changes in the B-field. With this setup, the magnets can be placed inside of the coils (Figure 2.11a). When ferromagnetic objects come close to the magnets, the field will be deflected. In turn, the changing B-field induces a voltage in the coil around the magnet, which is then amplified by an instrumentation amplifier, sampled by an ADC with 12 bit resolution at 55Hz and finally transmitted to a PC. The corresponding PCBs can be seen in Figure 2.10.

(a) Coil grid

(b) Amplifiers



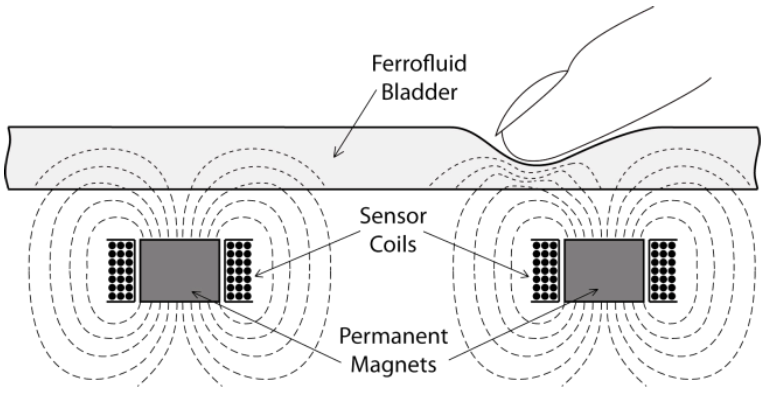
(c) Digital interface

Figure 2.10: The three modules that make up the reconfigurable ferromagnetic input device presented by Hook et al. [2009].

The intensity map is the output of the system.

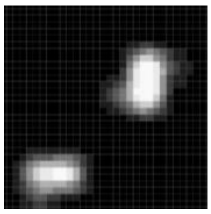
Like in earlier works, the intensity map is created by correcting offsets and ignoring measurements under a certain threshold followed by upsampling. After that, no additional steps are performed. The intensity map is the input for applications on the PC. An example for a sculpting application is depicted in Figure 2.11.

Assessment: Ferromagnetic objects have the advantage that they do not attract or repulse each other compared to ones containing permanent magnets. They are also cheaper

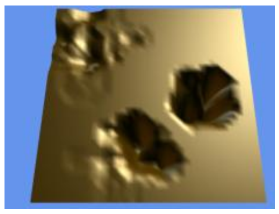


(a) Working principle when equipped with a ferrofluid bladder

(b) Interaction



(c) Intensity map



(d) Sculpting result

Figure 2.11: The detector is equipped with a ferrofluid bladder to create a deformable surface. The sampled intensity map is used to deform a surface in a sculpting application [Hook et al., 2009].

and ubiquitous compared to rare-earth magnets. The latter advantage is voided by the fact that rare-earth magnets are still needed in their work, just in the detector instead of the objects. The choice of coils as sensors has several drawbacks as well. Coils are large and the required spacing leads to a low spacial resolution. This is just obscured and not fixed by the upsampling step. Coils are heavy, so the weight reduces the portability of the system. The copper wire required to wind the coils is expensive, comparable to the cost of a cheap Hall sensor at the time of this writing. Lastly, coils only detect changes in the B-field. To obtain the state of the ferromagnetic object at a point in time, integration is necessary, which is prone to drift. When the ferrofluid bladder in Figure 2.11b is compressed a second time, close to the dent that was left by the first interaction, the original dent would be evened out, leading to a second registered activation that was not intended by the current interaction. These problems are neither mentioned nor addressed by the authors.

The pickup coils are large, heavy and expensive.

Only changes in the state of the system can be measured with coils and permanent B-fields.

Wireless Magnetic Position-Sensing System Using Optimized Pickup Coils for Higher Accuracy [Hashi et al., 2009, 2011]

Hashi et al. [2011] proposed a tracking system for medical purposes, which is also based on pickup coils. The system is shown in Figure 2.12.

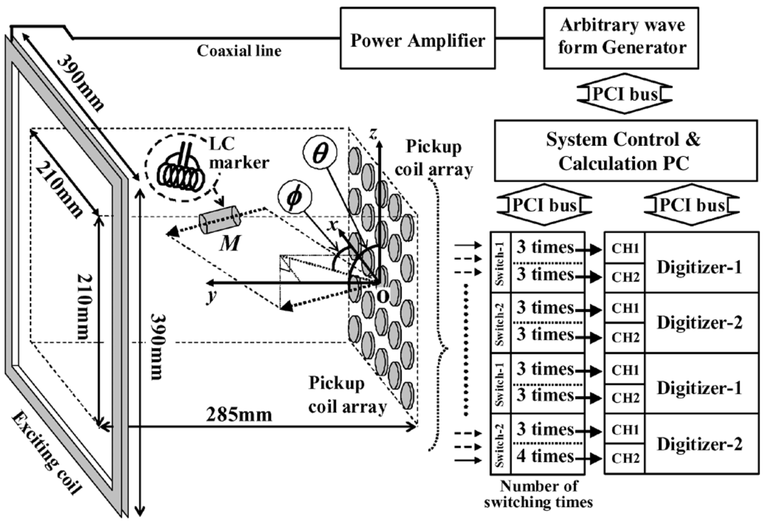


Figure 2.12: LC marker localization setup by Hashi et al. [2011]. LC (inductor and capacitor) resonant circuit based markers are excited by a large coil. The alternating B-field that the markers emit thereafter is picked up by a 5×5 array of coils that are 10mm in diameter and 1mm in height.

The system uses alternating B-fields that are emitted by passive markers after excitation with a large coil.

At 1mm height, the coils used in this system are much shorter than the ones used by Hook et al. [2009]. The excitation field is generated by a large coil instead of permanent magnets, and the markers are small LC oscillators, built out of a small capacitor and a coil wound around a ferrite core, which is just 3mm in diameter and 10mm long.

To start a measurement, the marker is excited by the excitation coil and the voltages at the pickup coils are sampled. The contribution of the excitation coil field has been measured before without a marker and is removed from the measurements, similar to the offset correction in previously mentioned works. For the localization, the field emitted by the marker is assumed to be close to an ideal dipole. The dipole equation is then used to construct a non-linear least squares problem, with six unknowns for the dipole position and orientation, where there is an equation for each pickup

The localization is done by solving an optimization problem based on the dipole equations with Gauss-Newton.

coil, resulting in 25 equations in total. This is the general approach that all the following works presented in this chapter pursue as well. The authors then use the Gauss-Newton method to find the marker position and orientation that minimize the loss of the over-determined equation system.

Assessment: LC resonant markers have the advantage over permanent magnets that they do not require rare-earth metals. They also allow for identification and multi-object tracking, since they can be tuned to different frequencies, which are mathematically orthogonal to each other and can thus be separated from each other in the sampled data. Unfortunately though, this is not covered by the paper. The alternating fields also fix the shortcoming of the ferromagnetic input device by Hook et al., which can only detect changes in the static fields and can thus never measure the current state of the objects. With the emitted field being alternating, the current absolute position of the marker can always be derived from the measured data.

No permanent magnets and thus no rare materials are needed.

A disadvantage of the electrical design of the tracking system is the high exciting peak voltage of 84V, which is not considered safe to touch and would be somewhat difficult to produce from low battery voltages in portable systems. It needs to be mentioned though, that the authors never intended this system to be portable. The waveform, that is used to excite the markers, is not given in the paper nor in its predecessor (Hashi et al. [2009]). In case a step or rectangle impulse is used, the exciting coils will act as an antenna and transmit signals with a wide band of frequencies that might cause interference with other devices.

High and potentially dangerous voltages are needed for the excitation of the markers.

The excitation coil might cause electromagnetic interference with other devices.

Unfortunately, no benchmarks or convergence rates of the numerical algorithm are given, apart from the general method and the accuracy of the localization results. The latter is out of the scope of our work, and thus not discussed here in more detail.

Tracking System with Five Degrees of Freedom Using a 2D-Array of Hall Sensors and a Permanent Magnet [Schlageter et al., 2001]

An influential early work that studies the suitability of a 2D grid of Hall sensors for tracking small rare-earth permanent magnets was published by Schlageter et al. [2001]. The authors analyze the physical limitations and tradeoffs

A combination of Hall sensors and permanent magnet markers is used.

in terms of sampling rate, sensing distance and localization accuracy of such systems in great detail. They do this by building and measuring a highly optimized real system, as well as by simulating several parameters that they deem too difficult to measure.

The electrical system is highly optimized to achieve the best signal-to-noise ratio.

1D Hall sensors are arranged in a 4×4 grid, with half of them oriented in x- and the other half in y-direction. The Hall sensors have been designed and built by Blanchard et al. in an earlier work. Special features are a cylindrical structure of the sensing element coupled with two integrated flux-concentrators, which are thin sheets of a highly permeable ferromagnetic material that divert some of the external field lines into the sensing element, resulting in a better detectivity than all of the commercially available sensors at that time. The differential sensor outputs are then amplified with a low noise instrumentation amplifier that was built for this specific purpose as well. After that, the offset is subtracted from the signal and it is put through a low-pass filter for noise reduction. It is then sampled with 16 bit at up to 50Hz and transmitted to a PC for further processing.

The localization is done by solving an optimization problem based on the dipole equations with Levenberg-Marquardt.

The localization is set up as a least-squares problem over five variables (3D position and 2D spherical angles) and solved with Levenberg-Marquardt as a trust region method. No additional information is given on implementation details or the performance and convergence properties of the algorithm.

Assessment: The electrical system is very well built, in contrast to many of the previously described systems, and is likely close to what is physically possible with the technology that is being used. With a different sensor technology, a higher sensing distance might however be achievable than the reported 14cm. According to the authors, only measuring the x- and y-components of the B-field leads to a decreased localization accuracy and sensing distance when the dipole is oriented vertically. The system was likely built this way, owed to the fact that their sensor design is not well suited to be placed upright. Since this paper has been published, 3D sensing structures have been developed and 3D sensors have become commercially available, so utilizing them might be a way to achieve even better results.

More suitable sensors based on other effects have become available.

We found that the reliability of Levenberg-Marquardt as a trust region method for this problem depends on the starting parameters (Section 5.5). This is unfortunately not examined in the paper.

3-Axis Magnetic Sensor Array System for Tracking Magnet's Position and Orientation [Hu et al., 2005, 2006, 2007, 2008, 2010]

Five methods to solve the optimization problem based on the dipole equations are tested in simulations.

Inspired by Schlageter et al., Hu et al. [2005] try to create a wireless capsule endoscopy system that can measure the position of a capsule in the human gastrointestinal tract and actuate its movement to guide it through unimportant areas more quickly. Due to the lack of implementation details on the localization procedure in related works, Hu et al. decide to evaluate several different methods in terms of their localization accuracy and execution time.

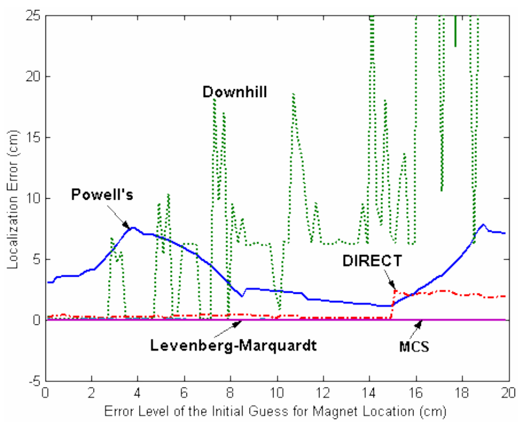
The equation system is based on the non-linear dipole equation with six variables, namely the 3D position and 3D dipole moment of the marker. The authors choose to normalize the dipole moment by adding the equation $\|m\| = 1$ to the system, discarding its length and with it the magnetization strength, effectively reducing the 3D dipole moment vector to the 2D orientation of the marker. The magnetization of the marker is instead included as a constant in their formulation of the dipole equation to compensate for that.

The influence of the distance between the initial guess and the localization result is examined in terms of position and orientation errors.

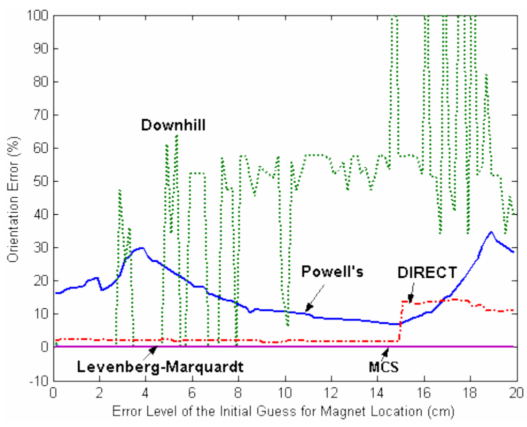
Five different optimization algorithms are tested: Powell's Algorithm, Downhill Simplex Algorithm, DIRECT, Multilevel Coordinate Search and Levenberg-Marquardt. The true position of the magnet for the first set of simulations is chosen as (10cm, 10cm, 12cm) and five 1D (x direction) B-field sensors are positioned in a cross pattern at 10cm distance between each other. This setup is used to explore the effects of the distance between the initial guess of the magnet position and its true position on the localization error and execution time for each method. The initial and true values for the orientations being used in the simulations are not given and neither is the amount of samples nor the sample distribution. The errors are measured as the Euclidean distances between the true and calculated values. The results are plotted in Figures 2.13 to 2.14.

Multilevel Coordinate Search and Levenberg-Marquardt produce essentially perfect results.

The positioning error of Powell's Algorithm is unacceptably high over the whole range, the Downhill Simplex Algorithm yields good results up to a distance of about 3cm, DIRECT is acceptable up to 15cm, and both Multilevel Coordinate Search and Levenberg-Marquardt find the exact true position over the whole range of initial values (Fig-



(a) Position error



(b) Orientation error

Figure 2.13: Effect of the distance between the initial guess and the true magnet position on the localization error for five different optimization algorithms based on five 1D sensors and undisturbed measurements by Hu et al. [2005]. Overall, the Downhill Simplex Algorithm performs the worst and Levenberg-Marquardt the best.

ure 2.13a). The results in regard to the orientation error are virtually identical (Figure 2.13b).

Levenberg-Marquardt is the fastest algorithm by far.

The fastest algorithm is Levenberg-Marquardt with an average execution time of 0.11s, followed by the Downhill Simplex Algorithm at 0.29s, Powell's Algorithm at 0.37s, DIRECT 0.50s and the slowest is Multilevel Coordinate Search with an average execution time of 0.69s (Figure 2.14). Based on these results, Levenberg-Marquardt is the obvious choice, being the fastest, as well as the most precise algorithm of all the ones that are tested.

The simulation conditions are not described thoroughly, considerably limiting the significance of the results.

Assessment: We found that many different aspects determine the performance of the Levenberg-Marquardt algorithm, including the distances between the initial values and the true values, but also the absolute values themselves and their position relative to the sensor grid (Section 5.5). For example, initial values of zero lead to many zero entries in the gradient of the objective function, which impedes the convergence of gradient-based methods. Some true positions close to a sensor are also more difficult to calculate due to the equation system being ill-conditioned. Neither the starting nor the true orientations for the presented simulations are given and only one favorable true position is tested. In addition, while the initial position is varied, the way in which this is done is not specified either. This has a large negative impact on the conclusiveness of the data that is presented.

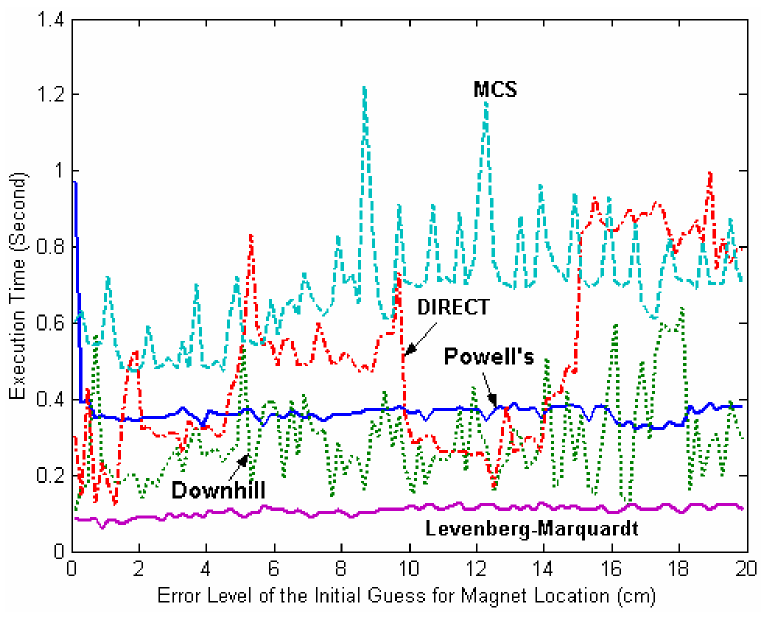


Figure 2.14: Effect of the distance between the initial guess and the true magnet position on the execution time for five different optimization algorithms based on five 1D sensors and undisturbed measurements by Hu et al. [2005]. Overall, Multilevel Coordinate Search is the slowest and Levenberg-Marquardt is the fastest approach.

The fact that the execution time is not close to zero for initial values that are close to the true values potentially indicates problems with the measurement of the execution time or a needlessly high constant overhead in the implementations of the algorithms, as they should only need a few, if not just one step to solve, unless an unfavorable initial value for the magnet orientation was chosen.

This linear algorithm is faster than Levenberg-Marquardt, but requires more sensors and is less precise.

To speed up the computation time and to increase the reliability of the localization compared to Levenberg-Marquardt, Hu et al. [2007] follow up by proposing a linear algorithm. A downside of the method is that it requires 15 measurements (five 3D sensors) to work, whereas only five measurements are needed in general to create a fully determined equation system. According to the simulations that the authors performed, more than five sensors are preferable when noise is present in the measurements. No exact numbers are given, but the authors claim that the proposed algorithm is about ten times faster than the solution using the Levenberg-Marquardt method, while having suitable localization accuracy under real-world conditions.

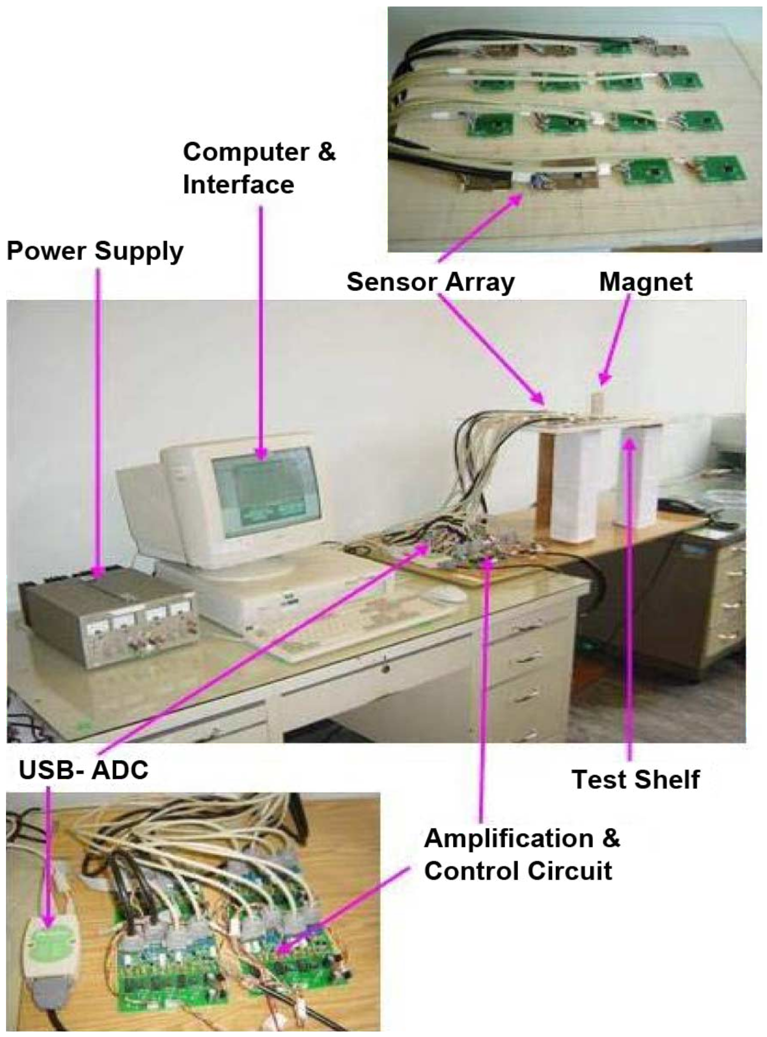


Figure 2.15: Permanent magnet localization system by Hu et al. [2006], consisting of sixteen HMC1053 B-field sensors, instrumentation amplifiers, analog-digital converters, a power supply and a PC

To profit from the strengths of both methods, namely the speed of the linear approach and the accuracy of the non-linear optimization, Hu et al. [2008] combine both methods by using the result of the former as initial values for the latter.

Based on their research on localization procedures, Hu et al. [2006] built a real system out of sixteen HMC1053 3D anisotropic magnetoresistive (AMR) sensors in a grid pattern (Figure 2.15). The sensors measure field strengths of up to ±6G and produce three analog output signals, which are amplified by AD623 instrumentation amplifiers and sampled at 12 bit, leading to a resolution of about 3mG/lsb.

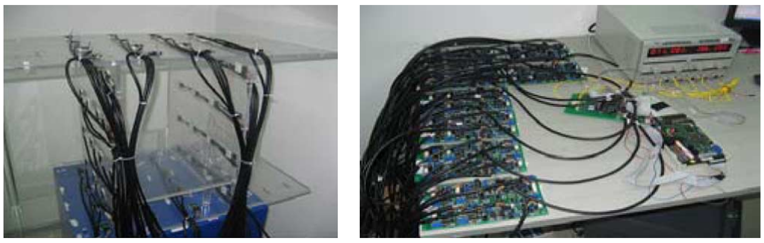
They use this system to determine the optimal number of sensors to minimize the positioning and orientation errors: A minimum of 8 sensors is needed for an average localization error of about 10%. After adding more than 15 sensors, the average error settles at around 5% and does not improve any further. They also test the influence of various materials when they come in between the sensor array and the magnet. Books, humans, copper plates and aluminum have almost no measurable effect, while a steel bar renders the localization results unusable.

In practice, adding more sensors to the network increases the accuracy of the results up to a point.

Non ferromagnetic materials do not interfere with magnetic tracking.

Hu et al. [2010] follow up their work by building a sensor network in the form of a four-sided cube to increase the measurement range over the previous limit of 15cm (Figure 2.16). This improved the localization accuracy significantly and allows the localization of magnets at most positions inside of the cube.

3D sensor arrangements can increase the accuracy and sensing range considerably.



(a) AMR Sensor cube **(b)** Signal processing hardware

Figure 2.16: Permanent magnet localization setup by Hu et al. [2010], with 64 AMR sensors on a four-sided cube and several modules for signal conditioning and analog-digital conversion

Assessment: The hardware is designed properly to get a good signal-to-noise ratio in the measurements, but better components have become available since this work has been published, which is an opportunity for improvement. The decision to arrange the sensors in 3D makes perfect sense for the application that the authors have in mind and the size and cost are not a problem either. For our application, we are interested in getting the best possible sensing range and performance out of planar sensor networks to support a wide variety of applications and environments, but it is evident that 3D sensor placements are very beneficial and should be supported as well.

publication	coverage [cm]	system size	tracked quantities	tag type	field type	sensor tech.	sensors	samples/s
Schlageter et al. [2001]	9×9×14	flat, portable	3D position, 2D orientation	passive	static	1D Hall	16	50
Hu et al. [2006]	20×20×15	flat, large	3D position, 2D orientation	passive	static	3D AMR	64	5
Han et al. [2009]	8×12	flat, portable	2D position	passive	static	2D AMR	2	$\gg$50
Hook et al. [2009]	10×10×n.a.	thick, heavy	2D intensity map	passive	static	coils	64	55
Hu et al. [2010]	50×50×50	cuboid, large	3D position, 2D orientation	passive	static	3D AMR	64	10
Hashi et al. [2011]	21×21×28	cuboid, large	3D position, 3D orientation	passive	alternating	coils	25	n.a.
Liang et al. [2012]	6×8×2	flat, portable	2.5D position, 60° tilt	passive	static	1D Hall	192	>60
Liang et al. [2013]	16×12×4.4	flat, portable	3D pos., 360° roll or 45° tilt	passive	static	1D Hall	768	>30
Fernandez et al. [2020]	≈20×20×10	moderate	3D position, 3D orientation	active	alternating	9D IMU	2	100
Our approach	arbitrary×t.b.d.	tiny, wearable	3D position, 3D dipole moment	passive	static	3D AMR	2-126	1000

Table 2.1: Comparison of other occlusion-free and predominantly B-field based localization systems.
The dimensions in the *coverage* column are given as x×y×z, with z being height. All of the systems with 2D sensor arrangements can be scaled in x and y directions, so the maximum sensing range z is the most interesting. It has to be noted that the sensing ranges have been determined under different thresholds and conditions, so the numbers are not directly comparable. They do, however, give a rough impression of what can be expected of the systems. Our sensor network is comparatively sparse and the sensors can be placed in arbitrary locations in 3D. 2D sensor configurations are especially useful in practice though, for installing the system under unused surfaces, so getting the most out of this kind of configuration should definitely be a focus for our work. We include our work in the table, but the final assembly and testing are out of scope, so the sensing range remains to be determined.

Our work

In the following, we summarize the key points of the discussed related work and explain their most limiting drawbacks for our intended application areas, e.g. gestural input in unused spaces [Drescher-Manaa, 2022]. Then, we give a short overview of the capabilities of our system.

GaussBits [Liang et al., 2013] offers high positional accuracy and the option to identify and track multiple markers, due to the design of the localization algorithm. However, the latter relies on a large number of sensors to work, which leads to a low refresh rate and an impractically expensive system. The Ferromagnetic Input Device [Hook et al., 2009] uses pickup coils as sensors, which makes it very large and heavy. In addition, it is only able to track changes in the system, and not the steady state. In contrast, the Wireless Magnetic Position-Sensing System [Hashi et al., 2011] uses much smaller and lighter coils and alternating fields, giving it the ability to measure the absolute marker position. On the other hand, the need for high exciting voltages and large exciting coils severely limits the flexibility to set up systems of different shapes and sizes. In addition, electromagnetic compatibility may be a concern. The Tracking System with Five Degrees of Freedom [Schlageter et al., 2001] misses the sixth degree of freedom and relies on Hall sensors, which limits the sensing range compared to AMR sensors. The 3-Axis Magnetic Sensor Array System [Hu et al., 2006] uses a slow approach for the localization, and thus suffers from a low refresh rate. In addition, the low degree of integration of the circuits makes the modules somewhat large and increases the power consumption, limiting the portability. It is based on old hardware as well, opening up some room for improvement.

We design a system that is compact and has low power requirements to maximize portability. It is possible to install the system in existing environments, without the constraints of having to place the sensors at a particular pattern, on a single plane or at certain orientations. In addition, the system is modular to allow for the customization of the sensing regions. The localization needs to be very fast to allow for high refresh rates of up to 1kHz and low latency, making it suitable for using it as a real-time input device for HCI applications like gesture detection. A high absolute localization accuracy would be beneficial, but is not necessary for applications, where visual feedback can be provided.

Chapter 3

Hardware

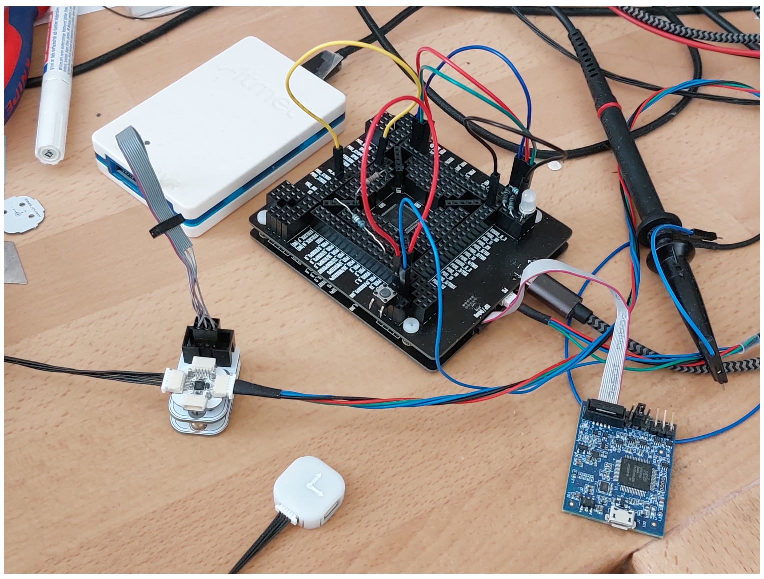


Figure 3.1: Overview of all the system components that are used in development: Two debug probes, namely an Atmel ICE (white box, top left) and a MCU-Link (blue PCB, bottom right), the evaluation board with the controller (black PCB, center), a B-field sensor PCB on the programming socket (white, bottom left) and a sensor with 3D printed case for protection (bottom center).

We prioritize compactness, flexibility and sensing range.

To achieve the goals that we set for our system at the end of Chapter 2, we have to reduce the amount of modules and components to the absolute minimum, while maximizing the sensitivity for a high sensing range. To that end, we build up on the work of Drescher-Manaa [2022] who investigated the suitability of the MMC5633 sensor for gesture input applications. The MMC5633 is a 3D AMR sensor with fully integrated sensing elements, signal conditioning, sampling, filtering and an I3C interface. Due to shortcomings in the design of the I3C interface and the implementation of the sensor, each sensor module PCB also hosts an ATtiny24A. As of this writing, very few controllers with I3C support are commercially available. We chose the LPC5536, an ARM Cortex-M33 based microcontroller with clock speeds up to 150MHz, a floating point unit, one I3C bus interface, eight serial interfaces and a USB 2.0 full speed host and device controller.

3.1 Issues and benefits of I3C

I3C is a bus specification that attempts to fix several shortcomings of I2C [MIPI, 2021]. I2C uses 7 bit addresses to enable the controller to communicate with specific target devices. The addresses are usually fixed by the manufacturer for each kind of device, making it likely for system designers to encounter address conflicts on buses with many targets. To make this less likely to happen and to make it possible to use more than one instance of the same device on the same bus, some vendors offer the same part under different part numbers with different preprogrammed I2C addresses or add extra pins that can be used to set some bits of the address externally. Both workarounds, however, increase the cost of the systems that rely on them.

I3C has fewer problems with address conflicts than I2C, except for in our case, where every sensor still has the same address.

To remedy this, I3C introduces a protocol that enables the controller to assign addresses to targets dynamically on startup. For this to work, an arbitration is necessary to prevent multiple devices from receiving the same address. Many details are omitted, but the core mechanism works as follows: First, the controller broadcasts a command that is an indicator for targets that the address assignment has started. The targets respond by sending their 48 bit provisioned ID, which consists of 15 bits for a manufacturer ID, 1 bit that dictates whether the next 32 bits are random or set by the manufacturer, 16 bits for the part ID, 4 bit for the instance ID and lastly, 12 bits for device characteristics. While all targets send their provisioned IDs in unison, they also listen to the data line of the bus to detect whether a target is responding with a lower provisioned ID. If there is one, the devices with the higher IDs will sit out the rest of the address assignment and wait for the next round. The problem here is that, for a bus where all targets are the same device, all provisioned IDs will only differ in the 4 bit instance ID section, exacerbating the problem that it was meant to fix. Even worse, the MMC5633 AMR sensors all have their instance IDs set to zero, meaning that all IDs are identical and only one can be used per I3C bus.

I3C makes it possible to start all measurements exactly at the same time.

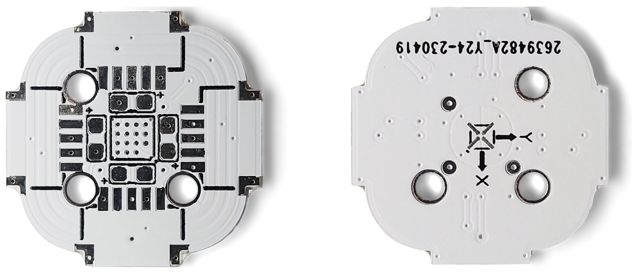
Another useful capability of I3C is the timing control. The most basic way to use the network to make continuous measurements is letting the controller instruct each sensor to start a measurement, one by one, and polling for the results in the same order. This will cause offsets in the timing of the individual measurement results, which translates to increased errors and possibly reduced performance in the localization step. With I3C, it is possible to broadcast an instruction that makes all sensors start taking measurements continuously, which aligns the timings very closely and reduces the workload of the controller. To prevent drift between the timings of the sensors, I3C specifies sync ticks that the controller has to send periodically to realign the individual sensor clocks. Polling is not necessary anymore, either. When a sensor completes a measurement, it sends an interrupt through the bus to convey this information to the controller, which can then collect the data. This again drastically reduces the workload of the controller, freeing it up to either perform more important tasks or enter a sleep mode to save energy, which might be beneficial for portable systems in battery powered applications.

3.2 Sensor

We made the sensor modules as small as possible.

It does not matter how the sensors are connected among each other, as long as they are connected.

We designed the sensor modules with the interests of future users in mind. 1mm pitch JST-SH connectors are placed on each of the four sides of the PCB (Figure 3.3) and wired in a way to allow the modules to be connected in a grid (Figure 3.4). If the cables all have the same length, this makes it very easy to keep the distances between sensors the same. Each of the connectors can be connected to any other, without any chance of causing damage, even when connecting ports of the same board. We chose this kind of connector for its size. Smaller connectors are available, but would be difficult to plug in. Larger connectors, on the other hand would increase the size of the module, which might limit the environments that it can be used in. Even the connectors that we chose are already responsible for about 85% of the PCB area (Figure 3.2). The dimensions of the PCB are 18mm$\times$18mm.

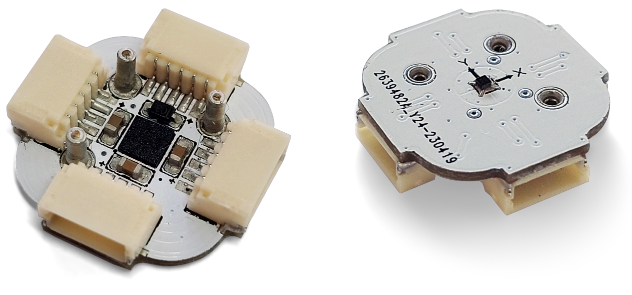


(a) Top **(b)** Bottom

Figure 3.2: Sensor module PCB, with a controller and connectors on the top and the B-field sensor on the bottom side

The sensors are tiny and consume little power, making them ideal for wearable applications as well.

The MMC5633 itself comes in a chip scale package of only 0.85mm×0.85mm×0.4mm, making it ideal for applications that require the B-field sensors to fit into tiny spaces. For non-modular applications, only a ceramic capacitor needs to be placed close to a sensor, making it ideal for integration in wearable items or fabric as well. Due to its small size, the orientation of the MMC5633 on the circuit board varies a lot from module to module though, so a calibration is required for good localization results. We placed the sensor at the bottom of the module to keep it as far away as possible from ferromagnetic metals such as nickel, which is often used as plating in component pins and will distort the magnetic fields. The lead connections of the four sensor pads break quite easily when external forces are applied, so it is advisable to place a drop of resin on the sensor for protection.



(a) Top **(b)** Bottom

Figure 3.3: Assembled sensor module PCB, with a controller and connectors on the top and the B-field sensor on the bottom side

Three ceramic capacitors are placed on the top side of the module, which are meant to stabilize and filter the power supply lines. Due to the low internal resistance of ceramic capacitors and the inductance of long wires, hot plugging modules to an active power supply will cause voltage transients that can readily damage logic-level integrated circuits. To prevent that from happening, we included a transient-voltage-suppression diode that will clamp the supply voltage on each module to acceptable levels. Hot plugging may still cause transients on the two I3C lines, which is not dangerous, but will be picked up by the controller. The controller then assumes that to be an indicator for an in-band interrupt and will get stuck in a subroutine waiting for a target to send data, which evidently never happens, so hot plugging should be done with care.

Hot plugging can cause issues, but we took measures to prevent damage.

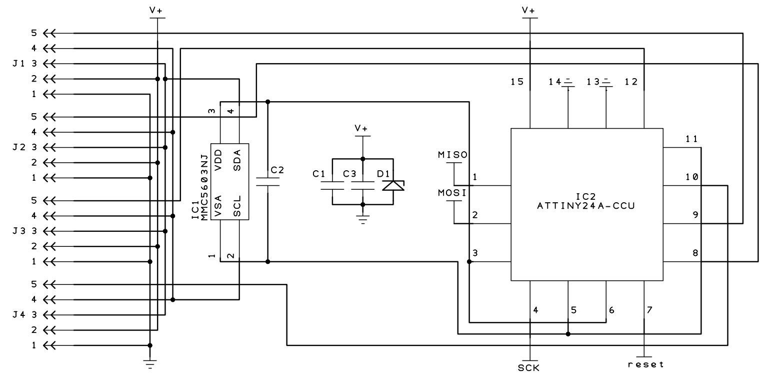


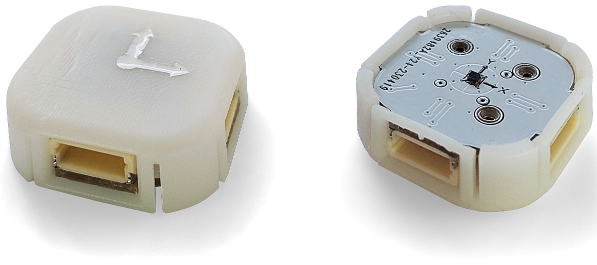
Figure 3.4: Sensor module schematic

To make dynamic address assignment work, we included the ATtiny with the smallest package size that still has a sufficient number of I/O ports on the sensor module, which turned out to be the ATtiny24A. With a microcontroller on each module, it would be possible to preprogram them with different static addresses and use I2C, but even under the circumstances that we described, I3C still offers much higher baud rates with a clock frequency of 12.5MHz compared to 100kHz for standard, or 400kHz for fast I2C. In addition, not having to resort to static addresses makes all sensor modules easier to program and users will not need to be concerned about connecting more than one sensor module with the same address to the same network.

A microcontroller on each sensor module fixes the dynamic address assignment of I3C.

I3C has much higher baud rates than I2C.

To fix the address assignment, we added a fifth wire to the connector. Each of the four connectors has the fifth pin connected to a different I/O port of the microcontroller. With this and a simple one wire protocol, we implemented a depth-first network discovery and toggle mechanism. Each time the controller sends a pulse through wire five, one of the sensors will have its power supply cut off and reconnected. This makes it forget its dynamic address and makes it willing to participate in the next dynamic address assignment. The controller can thus just alternate between resetting single sensors and assigning them a dynamic address.

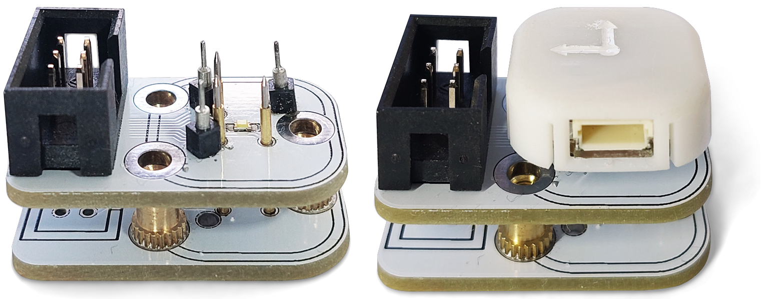


(a) Top **(b)** Bottom

Figure 3.5: Sensor module in enclosure with orientation marker on the top and the B-field sensor on the bottom

An enclosure protects the sensor module from damage.

To protect the sensor module assembly from damage, we designed an enclosure (Figure 3.5). Rounded corners prevent damage to and from the environment, bevels around the connectors help with plugging them in and a rim around the bottom of the PCB acts as a spacer for the sensor. We also added adhesive putty on the bottom of the PCB, which lets us place or remove sensors on about any surface. The PCB is installed in the case by just pressing it in from the bottom. The size of the complete module is 19mm×19mm×6.7mm.



(a) Without sensor **(b)** With sensor

Figure 3.6: Sensor module programming socket. The outer pins lock the module in place while the inner spring-loaded pins connect to pads on the underside of the module. The programming device can be connected to the shrouded header with standard pin configuration.

A programming socket simplifies programming.

The programming connectors for the ATtiny are on the bottom of the PCB to allow for programming the fully assembled and encased module. For this, we built a socket (Figure 3.6) with three outer pins to lock the sensor module in place and three spring-loaded pins to connect to three additional pads in the center, forming the six connections that are necessary for programming. An LED in the center of the socket indicates a successful connection. Due to the placement of the pads, it is not possible to install a module in a wrong way.

3.3 Controller

The controller has the task of coordinating, processing and bundling the measurements, and providing a clean API over USB. As of this writing, no evaluation boards for microcontrollers with I3C support are available, so we had to build our own (Figure 3.7).

We created an evaluation board for the microcontroller that also hosts short-circuit-proof voltage regulators, a breadboard, a fully compliant USB controller as well as USB, debug and I3C connectors.

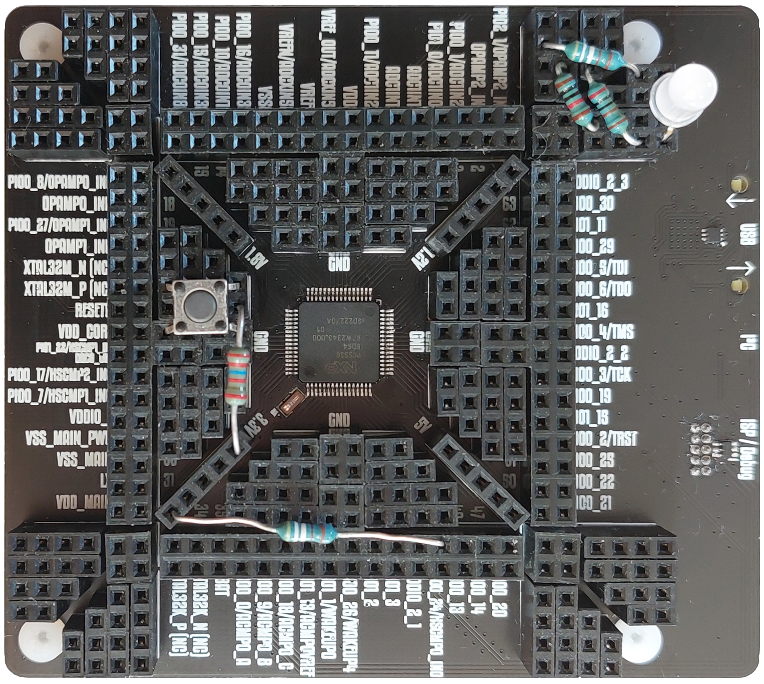


Figure 3.7: Assembled controller evaluation board. Pull-up resistors, a boot order change button and an RGB status LED, indicating whether the processor is idling, busy or in an interrupt routine are arranged on the breadboard.

We were only able to obtain samples of the 64 pin version of the microcontroller, which does not have the USB module bonded to any pins. An optional I3C pin is missing as well, so we had to compensate for that in our design (Figure 3.8).

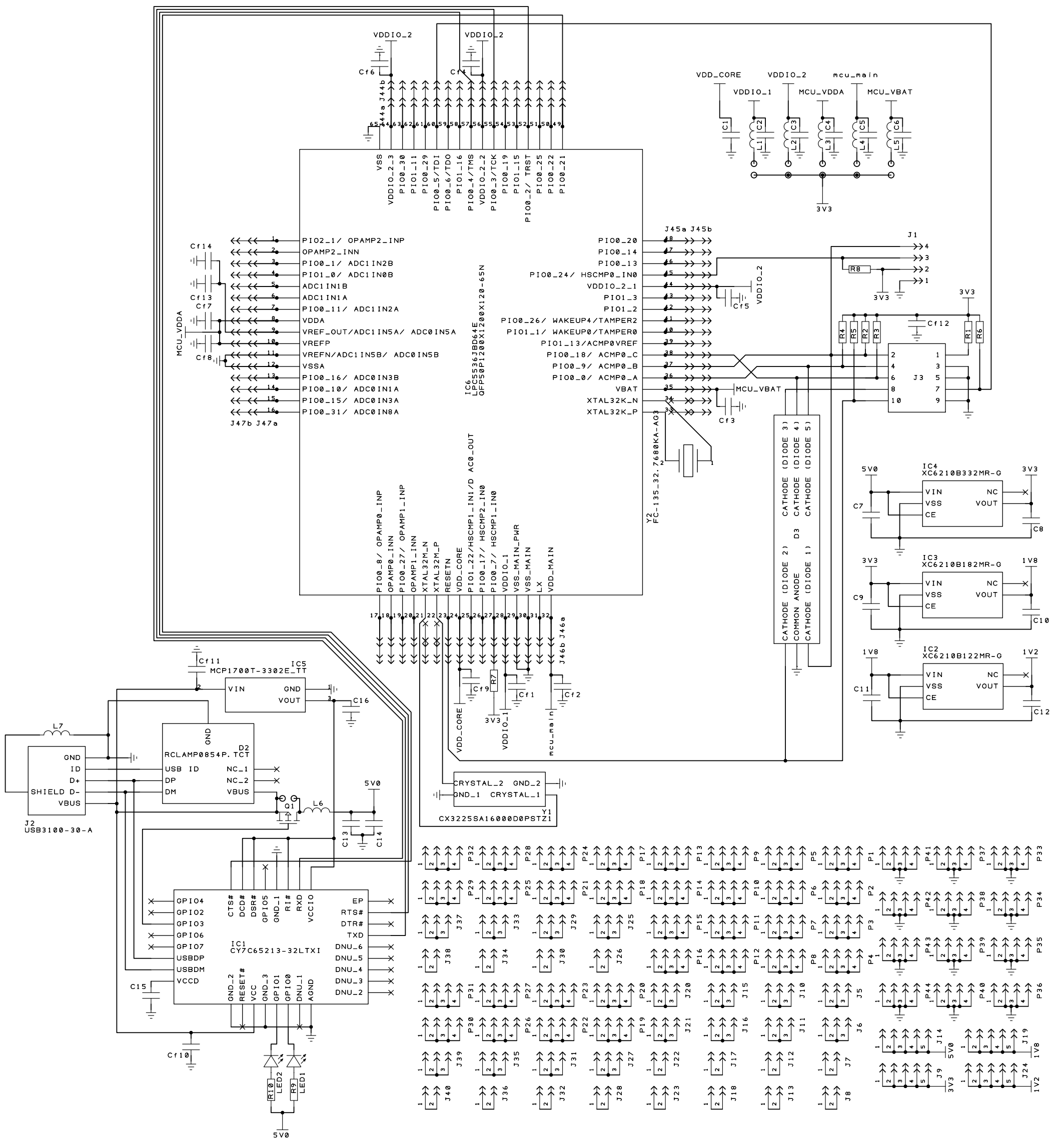


Figure 3.8: Controller evaluation board schematic

The controller board is powered by the host over the 5V USB supply rail. Most modern integrated circuits require much lower supply voltages of 3.3V, 1.8V or 1.2V, so we included three XC6210 linear regulators to lower the voltage and provide a very stable and noise-free power supply for the devices. The regulators also limit the operating currents to 800mA and short-circuit currents to 50mA, making the board safe to experiment with. For this purpose, an on-board breadboard is included on the PCB as well.

The controller board has three connectors: one USB port, a four pin JST-SH connector for I3C plus power supply and a 2×5 header for programming and debugging. Each pin of the microcontroller is connected to two sockets of the surrounding socket strip, except for the oscillator inputs, whose proper functioning would be at risk due to the added capacitance.

We use a CY7C65213 UART to USB bridge to provide USB functionality and make sure that the controller module is fully USB 2.0 compliant. The maximum baud rate is 3 Mbps, but we found that the maximum was limited by the flow control to about 2 Mbps in our setup. With efficient encoding, this should impose a similar limit on the data rate as I3C, so that should not be an issue.

Chapter 4

Localization algorithm

In this chapter, we describe the methods that we employ to estimate the magnet position, orientation and strength from B-field measurements. Unless noted otherwise, refer to Andrei [2022] for details on the optimization methods that we used.

4.1 Problem formulation

We assume magnets to be ideal dipoles, whose field can be calculated easily according to Equation (4.1). In this model, the dipole is located at the origin of the coordinate system and the field is calculated at position $r \in \mathbb{R}^3$ in Cartesian coordinates. The dipole moment $m \in \mathbb{R}^3$ points towards the north pole and its length is proportional to the field strength that the dipole produces. μ_0 is the vacuum permeability, which is approximately $4\pi \cdot 10^{-7}$ [NIST, 2018].

We use the dipole equation as approximation for the fields of permanent magnets.

$$B_r(r, m) = \frac{\mu_0}{4\pi} \left(3 \frac{\langle m, r \rangle}{\|r\|^5} r - \frac{m}{\|r\|^3} \right) \tag{4.1}$$

For a more elegant problem formulation, we prefer one stationary global coordinate space with a dipole at position p and multiple sensors at positions p_{s_k} instead. To this end, we reformulate Equation (4.1) by substituting r with $p_{s_k} - p$, so that the B-field at sensor k for a dipole at position p with a dipole moment m can then be calculated as follows:

A dipole equation parameter is substituted to allow for independent sensor and dipole positions.

$$B_k(p, m) = B_r(p_{s_k} - p, m) \tag{4.2}$$

With the six parameters $(p_x, p_y, p_z, m_x, m_y, m_z)$ of Equation (4.2), that we want to determine based on measurements, we need two 3D sensors that provide six equations to form a solvable equation system. More than two sensors lead to an over-determined equation system, which cannot be solved for p and m in the presence of measurement errors. To circumvent this problem, we formulate a loss function that assigns an error value to p and m, based on the deviations of the measurements from the fields that should have been measured. By minimizing that equation, we can obtain the values for p and m that have most likely lead to the measurements.

First, we stack all measurements $B_{s_k} \in \mathbb{R}^3$ of all n sensors into one column vector (4.4) and define a function $B(p, m)$ that maps the calculated fields at all sensor positions into a column vector as well (4.3).

$$B(p, m) = \begin{pmatrix} B_1(p, m) \\ \vdots \\ B_n(p, m) \end{pmatrix} \tag{4.3}$$

$$B_s = \begin{pmatrix} B_{s_1} \\ \vdots \\ B_{s_n} \end{pmatrix} \tag{4.4}$$

Next, we define the residual R as the vector of differences of the B-fields B_s that have, and the B-fields $B(p, m)$ that should have been measured:

$$R(p, m) \ = \ B(p, m) - B_s \ = \ \begin{pmatrix} B_1(p, m) \\ \vdots \\ B_n(p, m) \end{pmatrix} - \begin{pmatrix} B_{s_1} \\ \vdots \\ B_{s_n} \end{pmatrix} \tag{4.5}$$

We define a loss function and formulate a minimization problem to make the equation system solvable for the dipole position and orientation.

We then use the squared loss function (4.6) as the basis of our minimization problem, which is just the sum of the squared residuals [Dodge, 2008].

$$L(p, m) = \frac{1}{2} \|R(p, m)\|_2^2 \tag{4.6}$$

A loss based on squared residuals is easy to solve and should yield good results.

According to the Gauss–Markov theorem [Dodge, 2008], this error function is optimal under the assumption that the errors in the error vector $\epsilon \in \mathbb{R}^{3n}$ for measurements

$B_s = B(p, m) + \epsilon$ are uncorrelated and Gaussian distributed, with a mean of zero and equal variances. For real sensors, these assumptions should be met reasonably well, which is why we consider this kind of error function to be a good choice for the problem at hand. The factor of $1/2$ is optional and has no impact on the optimal values for p and m, but can be added to cancel out with a factor of 2 that appears in derivatives.

The solution for the magnet position p and dipole moment m can then be obtained by solving (4.7).

$$\arg\min_{p,m} L(p, m) \tag{4.7}$$

We perform the minimization by improving an initial guess step by step.

A large number of iterative methods exist to solve non-linear problems of this kind. They can be classified into direct methods, which only require the objective function that is supposed to be minimized or maximized, and into gradient-based methods, which require the objective function to be differentiable. All methods have in common that they locate the optimum, either by successively improving an initial guess or by shrinking an interval, that is expected to contain the optimum. We choose the former to solve the localization problem (4.7), i.e. starting from some values (p_0, m_0) and improving them step by step such that $L(p_{k+1}, m_{k+1}) < L(p_k, m_k)$. For this, we iterate according to (4.8), which leaves us with two subproblems to solve. The first subproblem is finding directions $\hat{p}$ and $\hat{m}$ to update the current values p_k and m_k in. We will deal with that in Section 4.3. The second subproblem is determining the optimal update step size h. We will have a closer look at that in Section 4.2. With this, the general form of the iteration equation is:

Determining the update direction and the update step size independently may improve the performance of the localization.

$$\begin{pmatrix} p_{k+1} \\ m_{k+1} \end{pmatrix} = \begin{pmatrix} p_k \\ m_k \end{pmatrix} + h_k \begin{pmatrix} \hat{p}_k \\ \hat{m}_k \end{pmatrix} \tag{4.8}$$

4.2 Determining the step size h

Once the update directions $\hat{p}$ and $\hat{m}$ are found (Section 4.3), we only need to search $L : \mathbb{R}^6 \to \mathbb{R}$ along a single line, so the loss function simplifies to $L_{line} : \mathbb{R} \to \mathbb{R}$,

For a fixed update direction, the loss function becomes one-dimensional.

$$L_{line}(h) = L(p - h\hat{p}, m - h\hat{m}) \tag{4.9}$$

and the minimization problem becomes

$$\arg\min_h L_{line}(h) \tag{4.10}$$

The one-dimensional loss function makes additional methods feasible.

with update step size h. This is called *line search*. Even though L_{line} only depends on h, it is not faster to evaluate than L. The advantage, that we are after, is the vast reduction of the search space from $\mathbb{R}^6$ to just $\mathbb{R}$, which makes using some methods possible, that would be too expensive in $\mathbb{R}^6$. Given an update direction that points towards the optimal solution, only one step would be necessary if we can also find the step size that is required to step directly into the goal.

The update step size h has to be determined on a logarithmic scale. The methods need to be adjusted to accommodate this.

We assume that $(\hat{p}, \hat{m})$ never points into the wrong direction, so we can assert that $h > 0$. The magnitude of h can vary by several orders, so it makes sense to search for this factor on a logarithmic scale. This means that all the methods, that we present in the following subsections, need to be adapted in this regard. We will examine three methods: Grid Search (Section 4.2.1), Golden Section Search (Section 4.2.2) and Newton-Raphson (Section 4.2.3). Some empirical examples of $L_{line}(h)$ are given in Figures A.1 to A.6.

4.2.1 Grid Search

Grid Search is fast, stable, simple and can distinguish global and local minima, but it scales poorly with the size of the search interval.

Due to its one-dimensionality, it is feasible to search for a value of h that minimizes $L_{line}(h)$ with a simple Grid Search. For this, we just evaluate $L_{line}(h)$ for a number of equidistantly spaced h in a chosen interval and keep the h that yielded the lowest result. Adapting it to search on a logarithmic scale is easily done by multiplying the current h with the Grid Search step size after each iteration, instead of adding it. This algorithm is easy to implement in general, but a few special cases have to be kept in mind when applying it to this particular problem.

As can be seen in Figures A.1 to A.6, all instances of $L_{line}(h)$ have flat regions that converge in direction of positive and negative infinity. Naive implementations, that start the search from the lower or upper end of the interval, will consistently choose either the largest or smallest h of the flat regions. This will either bring progress to a stop for small h, due to the tiny update step size, or foster parameter evaporation, i.e. large steps that are optimal in the given direction, but move orders of magnitude away from the optimal solution of $L(p, m)$. A good example, that leads to a very large step, can be seen in Figure A.1. A solution to the problem is searching from the center of the interval or some other preferred value outwards.

In addition, for instances of $L_{line}(h)$ that have no minimum, like the one seen in Figure A.1, it makes sense to check the loss at the interval bounds, independently of the chosen step size, as these positions are where the minimum is going to be in these cases.

Grid Search offers many benefits: There is no risk of divergence, as is often the case with gradient-based methods, and it is possible to locate the global minimum, as long as it falls into the sampled interval. Of the three methods, that we consider for the line search, only Grid Search can generally find the global minimum of $L_{line}(h)$, given a sufficiently small search step size.

The downside is that it is necessary to halve the search step size to double the amount of samples of $L_{line}(h)$ for each additional bit of precision of the resulting h. This means that getting a good estimate of h with this method is usually infeasible. It has to be noted that the final algorithm, that we found, is not very sensitive to errors in h though, so this is not too much of a problem.

4.2.2 Golden Section Search

A more efficient approach than an exhaustive search for solving the minimization problem (4.10) is dividing the search interval, similarly to the bisection method for finding roots of a function. In bisection, two interval bounds are be maintained, which need to be chosen so that they bracket a root. In each step, the interval can then be halved by replacing the appropriate bound with the center. To locate a minimum of a function instead, three values need to be maintained, and the interval has to be split in proportion of the golden ratio, hence the name Golden Section Search. The method can be adapted to search in a logarithmic space

Golden Section Search is stable, reasonably fast per step, of medium complexity and scales well with the search interval size, but is not guaranteed to find the global minimum.

pretty easily as well, by maintaining and working on the logarithm x of all h and calling $L_{line}(e^x)$ to evaluate the loss function.

As with the Grid Search, several special cases have to be considered in implementations of Golden Section Search: A step limit can be used as a stopping criterion, but it might make sense to also detect cases where the maximum precision is reached early to increase the execution speed. Traditionally, the result is chosen from inside of the interval. The minimum might be at the interval bounds as well though, so the loss of these needs to be evaluated and maintained as well, to be able to consider them for the final result. There are a few more details that can be considered, but these two are the most important ones.

With Golden Section Search, convergence is guaranteed and at a much faster rate than with Grid Search: In each step, the size of the interval is reduced to $\frac{\sqrt{5}}{2} - \frac{1}{2} \approx 61.8\%$ of its former size. So for one additional step, the precision of the result can almost be doubled. In comparison, Grid Search requires practically doubling the amount of steps to achieve the same.

The main disadvantage is the fact that convergence to the global minimum is not guaranteed. For functions with many minima, Golden Section Search might converge to any of them. For very few samples, Grid Search might actually be faster than Golden Section Search with an equivalent amount of steps as well, as Grid Search involves less overhead per step.

There are ways to generalize Golden Section Search to n-dimensional spaces [Rani et al., 2019], which makes it applicable to minimizing $L(p, m)$ directly, but comes at the cost of increasing the number of loss function evaluations, that are needed for each step by the power of n. It might still be useful, but this is not tested in this work.

4.2.3 Newton-Raphson

The Newton-Raphson method is used to approximate a root of a function $f : \mathbb{R} \to \mathbb{R}$ by iteratively improving an initial guess x_0. The intuition here is that, in every step, the current best guess x_k is replaced by the root of a tangent at that point. The iteration rule is:

$$x_{k+1} = x_k - \frac{f(x_k)}{\frac{d}{dx}f(x_k)} \tag{4.11}$$

Newton-Raphson can also be adapted to locate local minima by applying it to the first derivative of a function. In addition, the function can be transformed into logarithmic space by substituting x with e^x. The iteration step then becomes:

$$\begin{aligned} x_{k+1} &= x_k - \frac{\frac{d}{dx}f(e^{x_k})}{\frac{d^2}{dx^2}f(e^{x_k})} \\ &= x_k - \left(\frac{e^{x_k}f''(e^{x_k})}{f'(e^{x_k})} + 1\right)^{-1} \end{aligned} \tag{4.12}$$

We can use Equation (4.12) to solve the line search problem (4.10) by substituting f with $L_{line}(h)$, which yields the iteration rule for h:

$$\begin{aligned} h_{k+1} &= h_k - \frac{\frac{d}{dh}L_{line}(h)}{\frac{d^2}{dh^2}L_{line}(h)} \\ &= h_k - \frac{\frac{d}{dh}L(p - h_k\hat{p}, m - h_k\hat{m})}{\frac{d^2}{dh^2}L(p - h_k\hat{p}, m - h_k\hat{m})} \end{aligned} \tag{4.13}$$

The first and second derivatives of the loss function L with respect to h are given in Equations (4.14) to (4.18). To make them more concise, we substitute $p - h\hat{p}$ by P and $m - h\hat{m}$ by M.

$$\begin{aligned} \frac{d}{dh}L(P, M) =& \frac{d}{dh}\frac{1}{2}\|R(P, M)\|^2 \\ =& \left\langle R(P, M), \frac{d}{dh}B(P, M) \right\rangle \end{aligned} \tag{4.14}$$

$$\begin{aligned} \frac{d}{dh}B_k(P, M) = \frac{\mu_0}{4\pi} \\ \left(P\left(3\frac{c}{\|P\|^5} - 15\frac{\langle \hat{p}, P\rangle \langle M, P\rangle}{\|P\|^7}\right) + \hat{p}\left(3\frac{\langle M, P\rangle}{\|P\|^5}\right) \right. \\ \left. +M\left(3\frac{\langle \hat{p}, P\rangle}{\|P\|^5}\right) + \hat{m}\left(\frac{1}{\|P\|^3}\right)\right) \end{aligned} \tag{4.15}$$

$$\begin{aligned}\frac{d^2}{dh^2}L(P,M) =& \frac{d^2}{dh^2}\frac{1}{2}\|R(P,M)\|^2 \\ =& \left(\left\langle R(P,M), \frac{d^2}{dh^2}B(P,M)\right\rangle + \left\|\frac{d}{dh}B(P,M)\right\|^2\right)\end{aligned} \tag{4.16}$$

$$\begin{aligned}&\frac{d^2}{dh^2}B_k(P,M) = \frac{\mu_0}{4\pi} \\ &\left(P\left(-6\frac{\langle \hat{m},\hat{p}\rangle}{\|P\|^5} - 15\frac{\langle \hat{p},\hat{p}\rangle\langle M,P\rangle + 2\langle \hat{p},P\rangle\, c}{\|P\|^7} + 105\frac{\langle \hat{p},P\rangle^2\langle M,P\rangle}{\|P\|^9}\right)\right. \\ &+\hat{p}\left(6\frac{c}{\|P\|^5} - 30\frac{\langle \hat{p},P\rangle\langle M,P\rangle}{\|P\|^7}\right) \\ &+M\left(3\frac{\langle \hat{p},\hat{p}\rangle}{\|P\|^5} - 15\frac{\langle \hat{p},P\rangle^2}{\|P\|^7}\right) \\ &\left.+\hat{m}\left(-6\frac{\langle \hat{p},P\rangle}{\|P\|^5}\right)\right)\end{aligned} \tag{4.17}$$

$$c = -2a\langle \hat{m},\hat{p}\rangle - \langle \hat{m},r\rangle + \langle \hat{p},m\rangle \tag{4.18}$$

Even though the expressions are long, the fact that many of the operations are performed on scalars, and that many of the same dot products repeat, makes the evaluation of the derivatives sufficiently fast to be applicable.

Newton-Raphson converges very quickly, but may diverge if the starting position is too far from a minimum, the complexity is very high, due to the long derivatives and every step is very expensive.

As long as the initial guess is close to a root, it offers the advantage of converging much more quickly than Golden Section Search. In our simulations we found that three steps of Newton-Raphson is often enough to reach the full double floating point precision. However, if the initial guess is too far from a minimum it tends to diverge. It must also be ensured by some other means that the initial value lies within the attraction region of the global minimum. Lastly, every single Newton-Raphson step is very expensive. The time lost for the evaluation of the first and second derivatives is often compensated by the high convergence rate, but for the problem at hand, evaluations of $L_{line}(h)$ are very cheap in comparison, which gives the direct methods an edge.

Newton-Raphson could also be applied to the update direction subproblem or the localization problem as a whole, but that would require the second derivative of the loss function, which, as a tensor of order three, would be rather expensive to compute.

4.3 Determining the step direction

We consider three gradient-based methods for solving either the whole localization problem or just the update step direction subproblem. For non-linear equation systems the general idea is to approximate the function that needs to be minimized in every step with a linear or quadratic approximation to take a step into the downhill direction.

Since our objective is minimizing the loss function, we need the derivative of L

$$\nabla L(p,m) = \nabla \frac{1}{2}\|R(p,m)\|^2 = J(p,m)^T R(p,m) \tag{4.19}$$

which in turn requires the Jacobi-Matrix $J : \mathbb{R}^6 \to \mathbb{R}^{3n\times 6}$ of $B : \mathbb{R}^6 \to \mathbb{R}^{3n}$ for n sensors:

$$J(p,m) = \left(\begin{array}{c|c} \frac{\partial B_1}{\partial p} & \frac{\partial B_1}{\partial m} \\ \vdots & \vdots \\ \frac{\partial B_n}{\partial p} & \frac{\partial B_n}{\partial m} \end{array}\right) \tag{4.20}$$

$$\frac{\partial}{\partial p} B_k(p,m) = \frac{\partial}{\partial p} B_r(p_{s_k} - p, m) = \frac{\mu_0}{4\pi}\left(15\frac{\langle m,r\rangle}{\|r\|^7} rr^T - 3\frac{mr^T + rm^T}{\|r\|^5} - 3\frac{\langle m,r\rangle}{\|r\|^5} I\right) \tag{4.20a}$$

$$\frac{\partial}{\partial m} B_k(p,m) = \frac{\partial}{\partial m} B_r(p_{s_k} - p, m) = \frac{\mu_0}{4\pi}\left(3\frac{rr^T}{\|r\|^5} - \frac{I}{\|r\|^3}\right) \tag{4.20b}$$

The following methods could be used for the line search as well, but that would likely make little sense, as we would just constrain the methods to step into the initial update direction for multiple steps, instead of adjusting the update direction in every step as well. It would also require the rather expensive calculation of the derivative, albeit potentially at a reduced cost. Our goal with the line search is solving the problem faster, which is why we only consider two direct methods, that are less expensive to compute per step, or Newton-Raphson, which might make up for its high cost by converging much faster.

4.3.1 Gradient Descent

Gradient Descent is simple and rarely diverges, but often converges too slowly.

The simplest method that we evaluate is Gradient Descent. Starting from an initial guess, we take steps into the negative gradient direction of the loss function to successively reduce the loss.

$$\begin{pmatrix} p_{k+1} \\ m_{k+1} \end{pmatrix} = \begin{pmatrix} p_k \\ m_k \end{pmatrix} - h\, J(p_k, m_k)^T R(p_k, m_k) \tag{4.21}$$

This method converges only linearly towards a local minimum, which means that the error tends to zero like a geometric series. This is often very slow, even if an exact line search is used to determine h. On the other hand, this algorithm generally does not diverge, making it a safe choice. The cost and complexity per step is also very low, as it only requires one matrix multiplication on the highest level.

As mentioned before, we include the update step scaling factor h mainly to increase the rate of convergence, as Gradient Descent is very stable as-is.

4.3.2 Gauss-Newton

Gauss-Newton converges very quickly when starting close to a minimum, but has a tendency to diverge if it does not.

Gauss-Newton is based on a linear Taylor approximation of the loss function, which effectively solves a quadratic problem in every step. It iterates according to

$$\begin{pmatrix} p_{k+1} \\ m_{k+1} \end{pmatrix} = \begin{pmatrix} p_k \\ m_k \end{pmatrix} - h\, (J^T J)^{-1} J^T R \tag{4.22}$$

where $J^T J$ is an approximation of the Hessian. When L is represented reasonably well by the approximation, and when p_k and m_k are close to the optimal result, this method converges quadratically (i.e. very quickly). When these conditions are not met, Gauss-Newton is prone to making large steps, often increasing the parameters by several orders of magnitude in directions that only lead to little reductions in loss. This phenomenon is called parameter evaporation [Transtrum and Sethna, 2012] and turned out to be the main impediment when Gauss-Newton is applied to the dipole localization problem. It might also happen that J^TJ is singular, or at least ill-conditioned, which can happen for values for p and m that are close to zero or positions that are close to a sensor. In addition, if J^TJ is not positive definite, the step in the resulting direction might not lead to a reduction in loss at all.

Computing the inverse of a matrix is generally not advisable, so three methods of choice for solving (4.22) are presented in the following.

Making use of matrix decompositions is advisable when solving the inverse term of the Gauss-Newton equation.

Solution using Cholesky decomposition

We remember that the Cholesky decomposition decomposes a positive definite matrix A into the product of a lower triangular matrix L with its transpose:

The Cholesky decomposition is very fast, but fails for non positive-definite matrices and the condition number of J is squared, which can result in poor convergence rates.

$$A = LL^T \tag{4.23}$$

If A is not positive-definite, this is detected during the decomposition, which then has to be aborted. To remedy this, we decided to retry the decomposition after applying a regularization to A. An example would be adding the identity matrix to A.

We apply the Cholesky decomposition (4.23) to J^TJ in the Gauss-Newton iteration step in Equation (4.22) and rearrange it as follows:

$$\begin{aligned}
\begin{pmatrix}\hat{p}\\ \hat{m}\end{pmatrix} &= -(J^TJ)^{-1}J^TR && \text{(4.24a)}\\
\Leftrightarrow J^TJ\begin{pmatrix}\hat{p}\\ \hat{m}\end{pmatrix} &= -J^TR && \text{(4.24b)}\\
\Leftrightarrow LL^T\begin{pmatrix}\hat{p}\\ \hat{m}\end{pmatrix} &= -J^TR && \text{(4.24c)}
\end{aligned}$$

$$\Leftrightarrow \quad L^T\binom{\hat{p}}{\hat{m}} = y, \; L\,y = -J^TR \tag{4.24d}$$

Equations (4.24d) can then be solved efficiently by forward and backward substitution, respectively. While this is the fastest method of solving Equation (4.22), it has to be noted that calculating J^TJ explicitly squares the condition number of J, which may have a negative impact on the convergence speed when ill-conditioned equation systems are encountered.

Solution using QR decomposition

With the QR decomposition, we can avoid squaring the condition number, but it is slower and much more complex than the Cholesky decomposition.

As we know, the QR decomposition decomposes a matrix $A \in \mathbb{R}^{m\times n}$ into the product of an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ with a rectangular upper triangular matrix $R \in \mathbb{R}^{m\times n}$:

$$A = QR \tag{4.25}$$

We decompose J instead of J^TJ to avoid squaring its condition number and substitute it in the Gauss-Newton iteration rule (4.22). To avoid confusion with the residual $R(p, m)$, we rename the triangular matrix of the QR decomposition to R_J.

$$\binom{\hat{p}}{\hat{m}} = (J^TJ)^{-1}J^TR \tag{4.26a}$$

$$\Leftrightarrow \quad (J^TJ)\binom{\hat{p}}{\hat{m}} = J^TR \tag{4.26b}$$

$$\Leftrightarrow \quad ((QR_J)^T(QR_J))\binom{\hat{p}}{\hat{m}} = (QR_J)^TR \tag{4.26c}$$

$$\Leftrightarrow \quad ((R_J^TQ^T)(QR_J))\binom{\hat{p}}{\hat{m}} = (R_J^TQ^T)R \tag{4.26d}$$

$$\Leftrightarrow \quad R_J^TR_J\binom{\hat{p}}{\hat{m}} = R_J^TQ^TR \tag{4.26e}$$

$$\Leftrightarrow \quad R_J\binom{\hat{p}}{\hat{m}} = Q^TR \tag{4.26f}$$

We see that most of the terms cancel out, making the final result simple to compute via backward substitution.

Solution using singular value decomposition

We recall that the singular value decomposition (SVD) decomposes a real matrix $A \in \mathbb{R}^{m \times n}$ into a rectangular diagonal matrix $\Sigma \in \mathbb{R}^{m \times n}$ and two orthogonal matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$:

With the singular value decomposition, we can avoid squaring the condition number, use U for a convergence criterion and calculate the condition number of J in exchange for longer execution times.

$$A = U \Sigma V^T \tag{4.27}$$

By decomposing J instead of $J^T J$, we avoid squaring its condition number and get:

$$\begin{aligned}
&\binom{\hat{p}}{\hat{m}} = (J^T J)^{-1} J^T R && (4.28a)\\
\Leftrightarrow\ &\binom{\hat{p}}{\hat{m}} = ((U\Sigma V^T)^T (U\Sigma V^T))^{-1} (U\Sigma V^T)^T R && (4.28b)\\
\Leftrightarrow\ &\binom{\hat{p}}{\hat{m}} = ((V\Sigma U^T)(U\Sigma V^T))^{-1} (V\Sigma U^T) R && (4.28c)\\
\Leftrightarrow\ &\binom{\hat{p}}{\hat{m}} = (V\Sigma^2 V^T)^{-1} V\Sigma U^T R && (4.28d)\\
\Leftrightarrow\ &\binom{\hat{p}}{\hat{m}} = V\Sigma^{-2} V^T V\Sigma U^T R && (4.28e)\\
\Leftrightarrow\ &\binom{\hat{p}}{\hat{m}} = V\Sigma^{-1} U^T R && (4.28f)
\end{aligned}$$

We see that most terms cancel out and that the final equation can be computed very easily, with just a few matrix multiplications. Since Σ is a diagonal matrix, it can be inverted by just replacing every diagonal entry with its multiplicative inverse.

4.3.3 Levenberg-Marquardt

Levenberg and Marquardt proposed to add a damping term, consisting of a scalar scaling factor λ and a positive definite diagonal matrix D, to the Gauss-Newton iteration rule (4.22) [Transtrum and Sethna, 2012]:

Levenberg-Marquardt interpolates between Gradient Decent and Gauss-Newton, combining their robustness and high convergence rate, given that λ is chosen appropriately.

$$\binom{p_{k+1}}{m_{k+1}} = \binom{p_k}{m_k} - h\,(J^T J + \lambda D)^{-1} J^T R \tag{4.29}$$

For $\lim_{\lambda \to 0}$ the term $(J^T J + \lambda D)^{-1}$ is just $(J^T J)^{-1}$, which results in the Gauss-Newton iteration rule (4.22), while for $\lim_{\lambda \to \infty}$ the term $(J^T J + \lambda D)^{-1}$ becomes $(\lambda D)^{-1}$, which is the Gradient Descent step (4.21) divided by λ for $D = I$, where I is the identity matrix. This means that λ is effectively used to smoothly interpolate between Gauss-Newton and Gradient Descent, so, by choosing an appropriate λ, a compromise can be made between convergence rate and stability during each step.

Methods of choosing λ can be classified as direct or indirect:

A common direct method is decreasing λ by some factor after each step to increase the step size and speed up convergence. If the loss happens to increase after a step, λ is increased by another factor and the step is reevaluated until the loss decreases. This way, convergence can be enforced, however, potentially only at slow rates depending on the problem and the choice of D.

Indirect approaches are based on the idea of first choosing a step size δ in such a way that the approximation in that region is sufficiently accurate. After that, λ is determined such that $\|\binom{\hat{p}}{\hat{m}}\| \leq \delta$.

It has been reported that neither of these methods consistently outperforms the other one [Transtrum and Sethna, 2012]. Rather, the performance depends highly on the problem and the rate at which λ needs to be adjusted.

We combine Levenberg-Marquardt with the update step scaling factor h, as we did with previously mentioned methods, but it should be noted that this is usually not done, as choosing appropriate λ suffices to make Levenberg-Marquardt work as a so-called trust region method. However, if the line search can be performed sufficiently quickly, it might still lead to improvements in the convergence rate, which is why we added it here as well.

Solution using Cholesky decomposition

We apply the Cholesky decomposition (4.23) to $(J^T J + \lambda D)$ in the Levenberg-Marquardt iteration step in (4.29) and rearrange it as follows:

$$
\begin{aligned}
& \binom{\hat{p}}{\hat{m}} = (J^TJ + \lambda D)^{-1} J^T R && \text{(4.30a)}\\
\Leftrightarrow\ & (J^TJ + \lambda D)\binom{\hat{p}}{\hat{m}} = J^T R && \text{(4.30b)}\\
\Leftrightarrow\ & LL^T\binom{\hat{p}}{\hat{m}} = J^T R && \text{(4.30c)}\\
\Leftrightarrow\ & L^T\binom{\hat{p}}{\hat{m}} = y,\ L\,y = J^T R && \text{(4.30d)}
\end{aligned}
$$

The advantage in the Levenberg-Marquardt case compared to Gauss-Newton is that $(J^TJ + \lambda D)$ can be expected to be numerically positive-definite, so the decomposition should always succeed. The downside is that the decomposition will have to be repeated any time that the loss does not decrease during an optimization step, and λ needs to be adjusted, as it is part of the decomposed term.

Solution using QR decomposition

We apply the QR decomposition (4.25) to J in the Levenberg-Marquardt iteration step in (4.29) to avoid the increase in the condition number and also to avoid having to repeat the decomposition after adjusting λ:

$$
\begin{aligned}
& (J^TJ + \lambda D)\binom{\hat{p}}{\hat{m}} = J^T R && \text{(4.31a)}\\
\Leftrightarrow\ & ((QR_J)^T(QR_J) + \lambda D)\binom{\hat{p}}{\hat{m}} = (QR_J)^T R && \text{(4.31b)}\\
\Leftrightarrow\ & ((R_J^TQ^T)(QR_J) + \lambda D)\binom{\hat{p}}{\hat{m}} = (R_J^TQ^T) R && \text{(4.31c)}\\
\Leftrightarrow\ & (R_J^TR_J + \lambda D)\binom{\hat{p}}{\hat{m}} = R_J^TQ^T R && \text{(4.31d)}
\end{aligned}
$$

The right-hand side can be computed with just matrix multiplications. $(R_J^TR_J + \lambda D)$ is a triangular matrix, so the last term can be solved via backward substitution.

Solution using singular value decomposition

We apply the singular value decomposition (4.27) to J instead of $(J^TJ + \lambda D)$ in the Levenberg-Marquardt iteration step in (4.29) to avoid the increase in the condition number and to avoid having to repeat the decomposition after adjusting λ:

$$\begin{aligned}
&\binom{\hat{p}}{\hat{m}} = (J^TJ + \lambda D)^{-1} J^T R && (4.32a)\\
\Leftrightarrow &\binom{\hat{p}}{\hat{m}} = ((V\Sigma U^T)(U\Sigma V^T) + \lambda D)^{-1}(V\Sigma U^T)R && (4.32b)\\
\Leftrightarrow &\binom{\hat{p}}{\hat{m}} = (V\Sigma^2 V^T + \lambda D)^{-1} V\Sigma U^T R && (4.32c)\\
\Leftrightarrow &\binom{\hat{p}}{\hat{m}} = (V\Sigma^2 V^T + \lambda D)^{-1} V\Sigma U^T R && (4.32d)\\
\Leftrightarrow &\binom{\hat{p}}{\hat{m}} = (V\Sigma^2 V^T + V\lambda D V^T)^{-1} V\Sigma U^T R && (4.32e)\\
\Leftrightarrow &\binom{\hat{p}}{\hat{m}} = (V(\Sigma^2 + \lambda D)V^T)^{-1} V\Sigma U^T R && (4.32f)\\
\Leftrightarrow &\binom{\hat{p}}{\hat{m}} = V(\Sigma^2 + \lambda D)^{-1} \Sigma U^T R && (4.32g)
\end{aligned}$$

$(\Sigma^2 + \lambda D)$ is a diagonal matrix and can be inverted by replacing the diagonal entries with their multiplicative inverse. The remaining steps are just matrix multiplications.

Choice of D

We consider three different damping matrices, all of which offer different compromises between robustness and convergence rate.

We evaluate three possible choices for the damping matrix D in the Levenberg-Marquardt iteration step (4.29): The identity matrix, the diagonal elements of J^TJ, and the maximum diagonal elements of J^TJ that have been encountered during a localization up to that point [Transtrum and Sethna, 2012].

By choosing the *identity matrix* as the damping matrix, values that are lower in magnitude undergo higher damping than the others. Values in J^TJ, that are low in magnitude, have little influence on the final result and may thus be changed in rather large steps to counteract that. The additive damping equalizes the magnitudes of the values, making it an effective measure against parameter evaporation. The downside is that this choice of damping matrix gives the method trouble of traversing narrow canyons in the loss function.

A multiplicative damping can be implemented by choosing the *diagonal elements of* J^TJ as D. This enables Levenberg-Marquardt to traverse narrow valleys in the objective function much faster, at the cost of being less effective at preventing parameter evaporation, it has less of a damping effect on smaller values.

A compromise between these two choices for D is using the *maximum diagonal entries of* J^TJ that have yet been encountered during a localization. This way, the value scaling is largely preserved, but the fact that the values in D never decrease prevents inadequate damping of decreasing parameters.

4.4 Convergence, stop and abort criteria

After each iteration of the methods, that we discussed earlier, we need to decide whether to stop or to continue. In the following, we define three separate criteria to tell the algorithm that it should stop. It has to be noted, though, that this is not the focus of this work, so this aspect was only researched and evaluated very briefly.

We define three different criteria for stopping a localization, one that indicates success, one that detects failure and one that determines that we are not making any progress and should give up.

The one of the three that, is the easiest to define, is the *abort criterion*. The information that it conveys when it is true is that the algorithm takes too long to converge, and that it is unclear whether any more meaningful progress will be made. A step limit or a time limit are usually sufficient for this purpose.

The *stop criterion* is supposed to detect cases, in which the algorithm is unlikely to succeed, even when continued for longer. Detecting this reliably is much more difficult. Rules, that are based on the size of the update steps, on the reduction of the residuals or the reduction in loss do not work well for the localization problem, because phases of low reductions in these two metrics often occur even during eventually successful localizations. We have chosen to just put limits on the values of p, m and $R(p, m)$, respectively, as a simple heuristic for divergence.

The *convergence criterion* is not trivial to define either, for the same reasons that have been listed for the stop criterion. One could argue that putting a limit on the loss or residuals should be useful, but a certain value of loss does not translate directly into a certain localization accuracy, so a result that passes such a criterion would be unpredictable. In addition, the minimum achievable loss depends on the errors in the data, which are not known in advance, so there is no reliable way of choosing a suitable loss threshold to begin with. Our testing confirmed these assumptions. We did, however, manage to find a heuristic that performed almost as well as the ideal convergence criterion, which compares the current values of p and m with the true values:

$$cos\,\phi = \frac{\|UU^T R(p,m)\|_2}{\|R(p,m)\|_2} \tag{4.33}$$

Here, ϕ is the angle between the residual vector and the tangent plane of the model manifold, and U can be obtained by applying the singular value decomposition to J [Transtrum and Sethna, 2012]. In our tests, we managed to determine a threshold for $cos\,\phi$ that was almost always reached as soon as a positional deviation of less than 1mm was achieved, making this a very effective convergence criterion.

Chapter 5

Evaluation

In this chapter, we explain the condition under which we have run our simulations, use those to test several different aspects of our system and interpret the results in terms of two utility metrics. The most obvious metric is the amount of localizations that found a suitable result divided by the total number of localization attempts that were performed in a simulation:

We assess our simulation results based on two metrics: the convergence ratio and the expected computation time.

$$R_c = \frac{n_{converged}}{n_{converged} + n_{aborted} + n_{stopped}} \tag{5.1}$$

We will call this convergence ratio instead of convergence rate to avoid confusion with the usual meaning of the latter. In addition, we will call methods that are optimized to maximize the convergence ratio *successful*. Even though this is quite a simplification, we will treat the convergence ratio as the likelihood that a method converges within the limits that are set for the simulations.

None of the methods that we tested could achieve convergence ratios of 100% under any conditions, so a second metric that accounts for the implications might be in order. If a method fails, further localization attempts with different initial values will be necessary until it succeeds once. For an average execution time of successful localizations $t_{converged}$ and an average time of stopped or aborted localization attempts $t_{failure}$, the expected computation time until a localization is successful is given as follows:

The expected computation time is a more useful metric than the convergence ratio.

$$t_e = t_{converged} + t_{failure}\frac{1 - R_c}{R_c} \tag{5.2}$$

We will call methods, that are optimized to minimize the expected computation time, *fast*.

5.1 Simulations

Hardware

We avoid some pitfalls to obtain comparable results.

All simulations have been performed on an AMD Ryzen 9 3900X CPU (12-Cores/24 Threads) on 22 threads to leave some capacity for background tasks as well as the garbage collector. The process has been given a high priority, the CPU temperature was monitored and the clock frequency has been fixed at 3.725 GHz to ensure reliable benchmark results.

Convergence, stop and abort criteria

We use a convergence criterion based on the true values to eliminate it as a variable in our tests.

To avoid testing the adequacy of a convergence criterion and the performance of the solvers at once, we used the knowledge of the true values p_t and m_t to construct an ideal convergence criterion based on our requirements:

$$\|p - p_t\|_2 < 0.001m \;\wedge\; \frac{\|m - m_t\|_2}{\|m_t\|_2} < 1\% \tag{5.3}$$

As a stop criterion, we used a heuristic that is supposed to detect divergence:

$$\|p\|_\infty < 10^9 \;\wedge\; \|m\|_\infty < 10^{12} \;\wedge\; \|R(p, m)\|_\infty < 10^{24} \tag{5.4}$$

The abort criterion is just a step limit. The optimal amount of steps is subject to our evaluation when optimizing for speed. When optimizing for success, we used a somewhat arbitrary step limit of 50. Increasing the limit further only lead the diminishing improvements in convergence ratios.

Algorithm

All code has been implemented in Java 18 and was optimized in hot spots using the Java Microbenchmark Harness. We evaluate all three algorithms listed in Section 4.3 with all the possible decompositions, both with and without line search. For Levenberg-Marquardt, all damping methods will be tested as well, but we decided to leave out the QR decomposition, as it is unlikely to offer any advantages compared to the singular value decomposition in our setting while also taking a very long time to simulate.

To test whether Levenberg-Marquardt can lead to an improvement in convergence ratio over the other methods under optimal conditions, we determine λ with a logarithmic Grid Search, with a step multiplier of $\sqrt{10}$ over the interval $[10^{-15}, 10^{10}]$. For each sampled λ, we then additionally perform the line search if applicable, before evaluating the loss function with the given λ and h. This is very slow, so we excluded Levenberg-Marquardt from some speed benchmarks, as the comparison with methods, that are optimized for speed, would not make much sense.

We make sure to find the best λ for Levenberg-Marquardt to test its limits under optimal conditions.

We found that the optimal choice for λ often varies in several orders of magnitude between steps for the dipole localization problem, making direct methods of determining it a poor choice. We also confirmed this in tests that are not included in more detail, but are what led us to the decision to use an exhaustive search to guarantee the usage of suitable values for λ.

Determining λ with direct methods likely leads to subpar performance.

For the line search, we always start with a Grid Search, whose optimal initial starting interval will be determined in our tests. Skipping the Grid Search makes little sense, as we know from our examples of L_{line} (Figures A.1 to A.6) that we have to expect multiple local minima, which is something that Golden Section Search and Newton-Raphson cannot deal with. In addition, a coarse Grid Search is very cheap. The result of the Grid Search is the best value that was encountered for h and $[h/\delta, h{\cdot}\delta]$ is set as the new search interval, with δ being the Grid Search step size. We also intersect the new search interval with the initial one to make sure to not expand the search interval beyond the original bounds.

We perform the line search by executing a Grid Search, Golden Section Search and Newton-Raphson in that order.

The next step may or may not be a Golden Section Search, depending on the chosen maximum number of steps for this method, with the refined interval as input. The result will again be a new search interval and a new optimal value for h.

The optional last step is Newton-Raphson, which starts with the optimal h that was found in the previous step and provides a more accurate h if it started within the convergence radius of a local minimum in the loss function, but does not refine the search interval. The new value for h is again constrained to the search interval, to make sure that it remains in bounds.

Simulated system

All simulations are performed with four sensors and under ideal conditions without errors, apart from floating point inaccuracies.

We decided to place four sensors in a square pattern with 20cm distance to emulate a gesture input scenario where the sensors have been placed on the underside of a table, as shown in Figure 5.1. The calculated sensor measurements are left as-is without performing quantization or adding errors or noise. All calculations are performed on 64 bit floating point values.

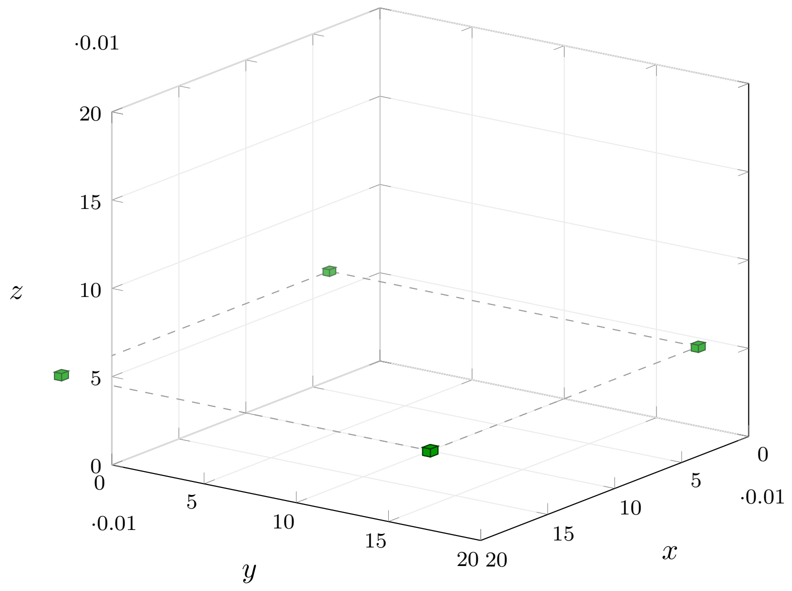


Figure 5.1: Sensor placement, units are in cm. Due to the symmetry, the simulations only need to sample positions inside of the box spanned by the axes.

A simulation consists of multiple localizations, with each having a distinct set of parameters. There are four vectors or twelve parameters in total that need to be adjusted for each localization, namely the initial values p_i and m_i and the true values p_t and m_t, to generate data that is sufficiently representative of the system as a whole. For parameters that do not affect the bin that a localization is sorted into, p and m are usually sampled according to Figures 5.3 and 5.5. For the simulations in Sections 5.2 to 5.4.1 these detailed sampling schemes have been used in all simulations for all four parameter vectors, resulting in a total of 9 434 112 different localizations after removal of cases where the initial values equal the true values. For some of the simulations it was necessary to randomly skip some localizations to shorten the simulation times. This was especially the case for Levenberg-Marquardt, as it is slow and has many variants that need to be simulated. No samples were dropped in the simulations that lead to the results presented in Table 5.1.

In each simulation, we test many combinations of initial and true values.

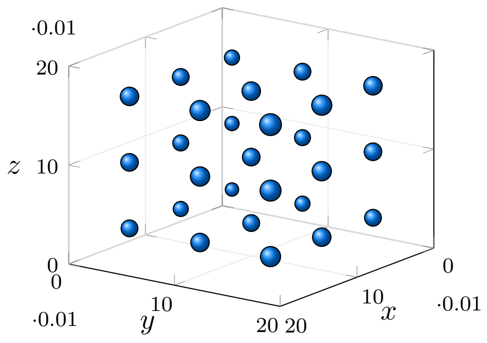


Figure 5.2: Default p samples
3D grid, 3 samples per axis

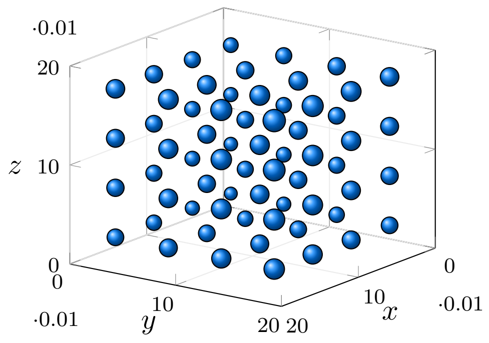


Figure 5.3: Detailed p samples
3D grid, 4 samples per axis

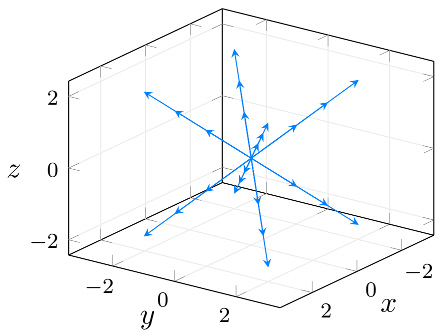


Figure 5.4: Default m samples
8 points on a sphere, 3 scales
Vector lengths scaled according to $log_{10} + 1$

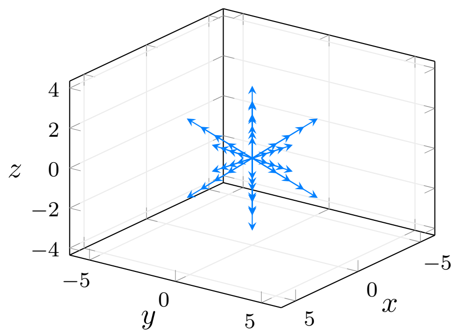


Figure 5.5: Detailed m samples
12 points on a sphere, 4 scales
Vector lengths scaled according to $log_{10} + 1$

The simulations in Section 5.5 require a higher resolution along the binned dimensions. The p and m distributions that were used for that purpose are displayed in Figures 5.6 to 5.15. Points and vectors with the same color belong to the same bin and simulation results within the same bin are averaged to obtain a single value. All other parameters are sampled according to Figures 5.2 and 5.4.

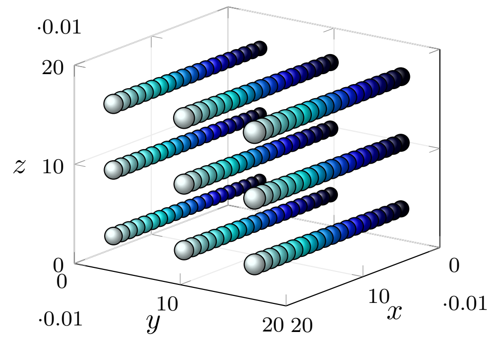


Figure 5.6: p with high x resolution
3D grid, 20 samples in x direction

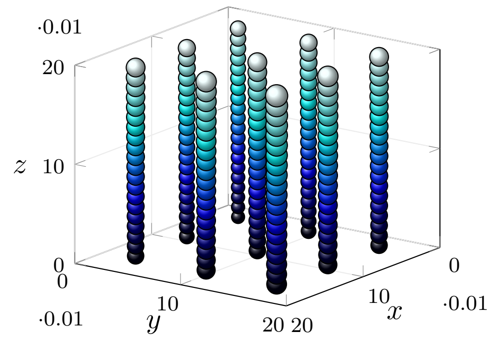


Figure 5.7: p with high z resolution
3D grid, 20 samples in z direction

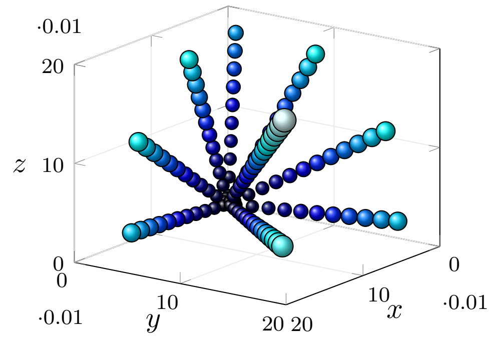


Figure 5.8: p with set distances from origin
9 points on a sphere segment, 20 scales

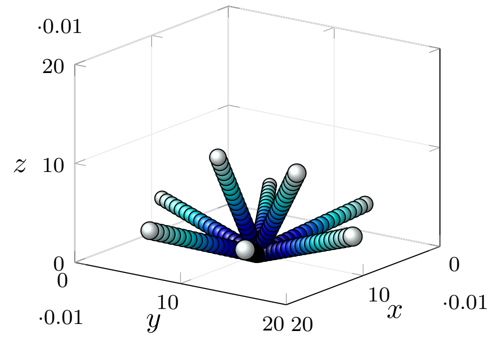


Figure 5.9: p with set distances from sensor
8 points on a half sphere, 20 scales

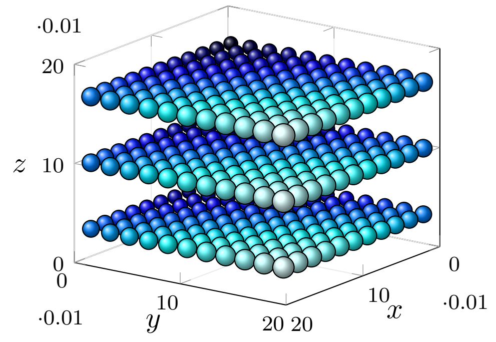


Figure 5.10: p with 21 lines in xy direction
3D grid, 11 samples in x and y direction

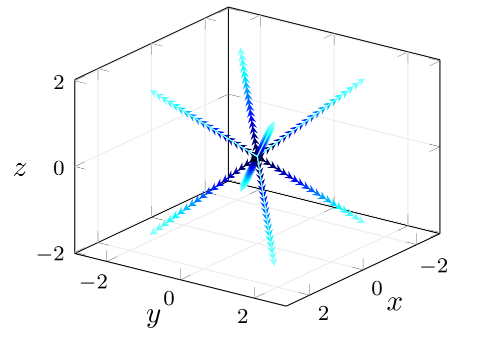


Figure 5.11: m magnitude
8 points on a sphere, 20 scales
Vector lengths scaled according to log_{10}

Some of the metrics, that are used in the diagrams in Section 5.5, depend on the corresponding initial or true value. The red arrow in the sampling schemes, displayed in Figures 5.12 to 5.15, signifies the input vector or point.

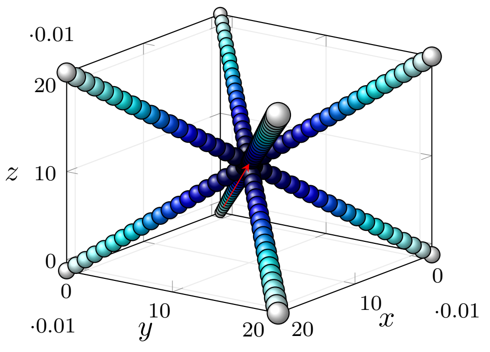


Figure 5.12: p distance
8 offsets on a sphere, 20 scales

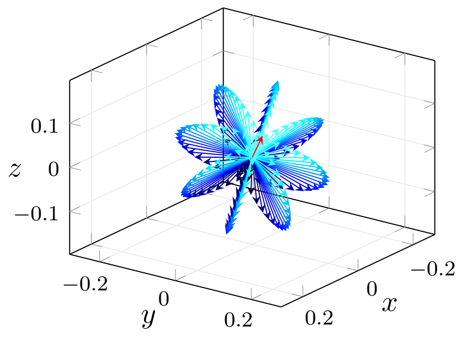


Figure 5.13: m angle
Angle changed in 8 directions in 20 steps

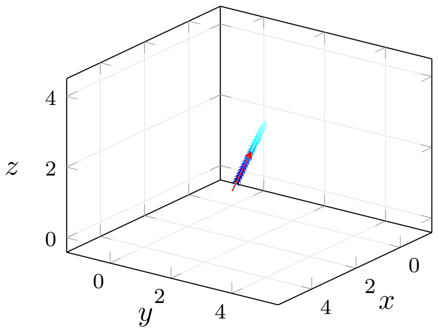


Figure 5.14: m scale
Input vector is scaled in 20 steps
Vector lengths scaled according to $log_{10} + 5$

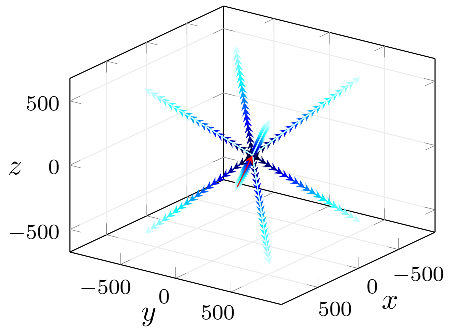


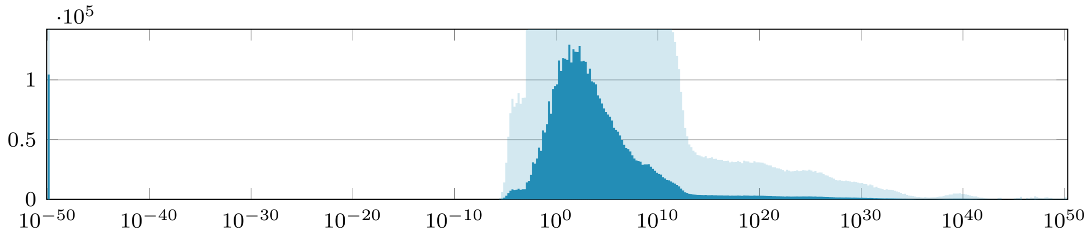
Figure 5.15: m distance
8 offsets on a sphere, 20 scales

5.2 Optimal h search interval

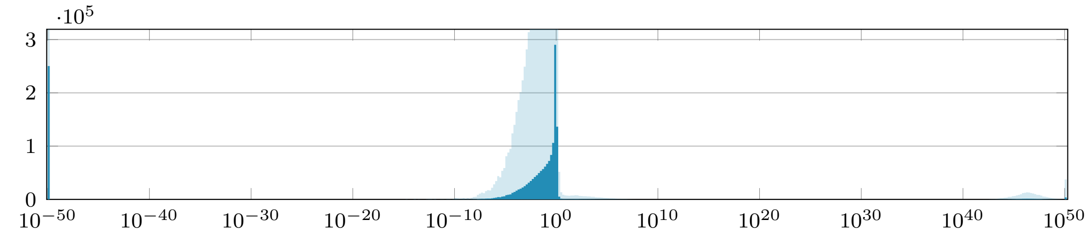
All the methods, that we evaluate for determining the optimal update scaling factor h, which is supposed to minimize the loss in the update direction, require a search interval or a starting value. The most obvious approach to determine a suitable search interval is running tests with a huge interval and sampling the distribution of the optimal h values over many simulations for each method and choosing an interval that contains most of the distribution. To determine the minimum of $L_{line}(h)$ reliably, we perform a very fine Grid Search with 100 steps per decade on the first and second

We need to define an interval to perform the line search in.

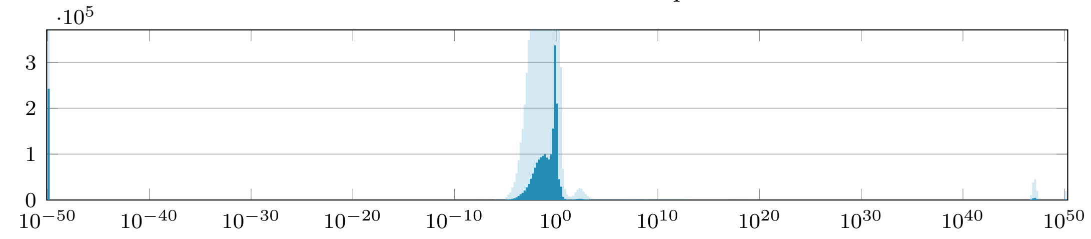
derivative to detect changes in the sign of the result. When a sign change is detected, we apply Bisection to locate the root. By doing this, we can record the locations of all extrema and inflections of $L_{line}(h)$. We then refine the results for minima further by applying a Golden Section Search, that based on a $L_{line}(h)$ implementation that uses floating point values with 96 bit mantissas, so that the final location as a double precision floating point value is within 1 ulp of the actual result.



(a) Gradient Descent



(b) Gauss-Newton, QR decomposition



(c) Levenberg-Marquardt, QR decomposition, diagonal damping

Figure 5.16: Histograms of optimal absolute h values in an interval of 10^{-50} to 10^{50}, as determined by a perfect line search, for a selection of the most successful methods of their class. The h bins are on the x-axis and the counts on the y-axis. The transparent blue bars are scaled by a factor of 10 to make bins with low counts more visible.

The results for a selection of methods are plotted in Figure 5.16. For Gradient Descent, the update step size usually needs to be increased, often by several orders of magnitude. For Gauss-Newton, h is mostly a damping term, with 1 being the most likely value and a distribution that rapidly decays and reaches zero at about 10^{-10}. The histogram for Levenberg-Marquardt looks like a mixture of three Gaussians, but is overall very similar to the Gauss-Newton distribution. For all methods quite a few scaling factors are at the lower bound of 10^{-50}, with the actual optimal values likely being less than that. This might indicate that the update vector points along a curved valley in the loss function, which results in the minimum of the loss function in that direction being just a tiny step away. While most values are more or less lumped in the middle of the search interval, for all three methods there are values that would have exceeded 10^{50} as well. While the large search interval most often guarantees that the line search algorithm finds the h that minimizes the loss in the given direction, these intervals are too large to search sufficiently quickly and include unreasonably large or small h values.

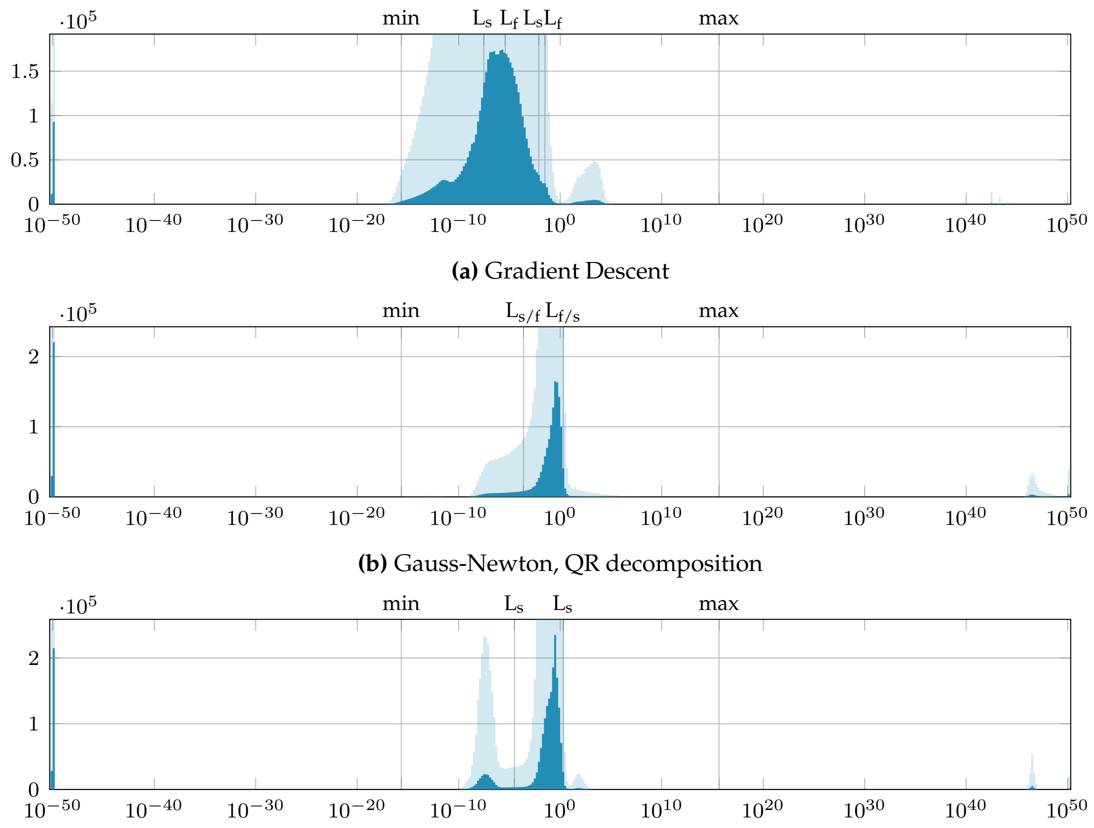
Optimal values for h are spread over large intervals.

The impact that an update has on the current values for p and m depends on the order of magnitude of the current values p_k and m_k, the update vectors $\hat{p}_k$ and $\hat{m}_k$ and h. For double precision floating point values the size of the mantissa is 52 bits, so the smallest value that we can add to 1 that results in an increase instead of being rounded away is 2^{-52}. Conversely, when adding a value larger than 2^{52} to 1, the value of 1 will be rounded away. In the 1D case of our update term (4.8), $d_{k+1} = d_k + h_k\,\hat{d}_k$, with double precision floating point scalars d, these limits will shift depending on the magnitude of the current value and update, d_k and $\hat{d}_k$ respectively. If $\hat{d}_k$ is smaller in magnitude than d_k, the interval for h in which both the update as well as the current value will contribute to the final result will shift towards higher values and vice versa.

Small h result in no progress being made. Conversely, large h completely override the result with the update.

To obtain a more meaningful representation of h that directly corresponds to the impact of the update, we scale the update direction vector to have the same magnitude as the current result vector before performing the line search. We will call these *h relative* and the h without normalization *absolute*. For vectors v, we modify the update term (4.8) to $v_{k+1} = v_k + h\,\frac{\|v_k\|}{\|\hat{v}_k\|}\hat{v}_k$. In this space we can limit the interval of reasonable values for h to lie between approximately 2^{-52} and 2^{52}. In contrast to the 1D case, this method is not perfect, as h values beyond the limits can still lead to changes in at least some components of the vectors, but it still opens up a more useful space to search for h values and search interval limits in. Similar to Figure 5.16, that was based on the absolute h space, Figure 5.17 shows the results of the corresponding simulations in the relative h space.

We can normalize the update direction vector, so that h corresponds to the impact of the update.



(c) Levenberg-Marquardt, QR decomposition, diagonal damping

Figure 5.17: Histograms of optimal relative h values, as determined by a perfect line search, for a selection of the most successful methods of their class. The h bins are on the x-axis and the counts on the y-axis. The transparent blue bars are scaled by a factor of 10 to make bins with low counts more visible. Values for h that are below *min* have no impact on the current values for p and m while scaling factors over *max* make the update override the current values completely. The other lines mark the limits that can be set to make the method as fast as possible (L_f) or optimize the robustness to initial values to make them as successful as possible (L_s).

The normalization leads to more compact distributions of optimal h.

As a result of the normalization, the distribution for Gradient Descent is much more compact. It is also noticeably skewed towards lower values, which hints at the fact that the update vector tends to point in a direction where only little progress can be made, hence the small steps. For Gauss-Newton, the variance of the distribution seems to have decreased slightly. Interestingly, there are some values far past the maximum marker, which would immediately trigger any sensible divergence criterion and stop the localization process. The Levenberg-Marquardt distribution is split into two parts. The values of the smaller distribution are a little bit concerning, as they will result in slow progress. Even worse, all methods chose update directions that lead to h that are smaller than 10^{50}, which usually results in no progress at all and lets the method repeat the same step until the step limit is reached and the localization is aborted. Based on this we can assume that limiting the range of permissible h values not only benefits the speed at which we can perform the search, but also prevents stalling and divergence.

Sufficiently many tiny or huge h are chosen to have a negative impact.

The result of limiting the search intervals can be seen in Figures 5.18 to 5.22. The limits have been determined as described in Section 5.4. Especially for Gradient Descent a large portion of lower h values have been cut off. This can prevent the algorithms from stalling too much and hitting the step limit, even if it means that the optimal reduction in L_{line} might often be missed. All the small distributions over 2 have been excluded as well. The reason for this might be that even h values that are not much greater than 2 are already sufficient to at least cause a detour on the way to the global minimum.

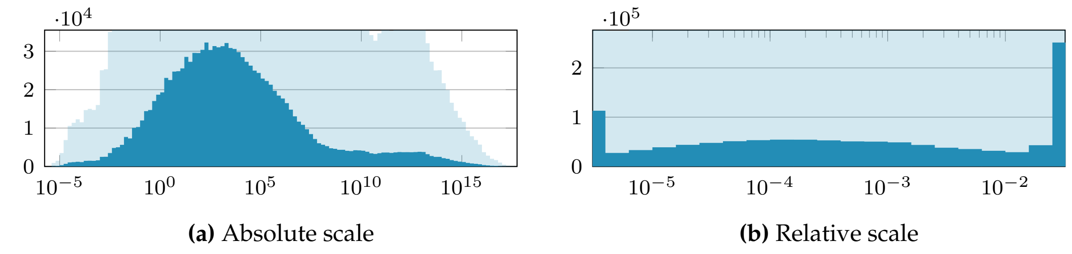


Figure 5.18: Optimal h distribution: Gradient Descent optimized for speed, h limited from $3.8 \cdot 10^{-6}$ to $3.1 \cdot 10^{-2}$

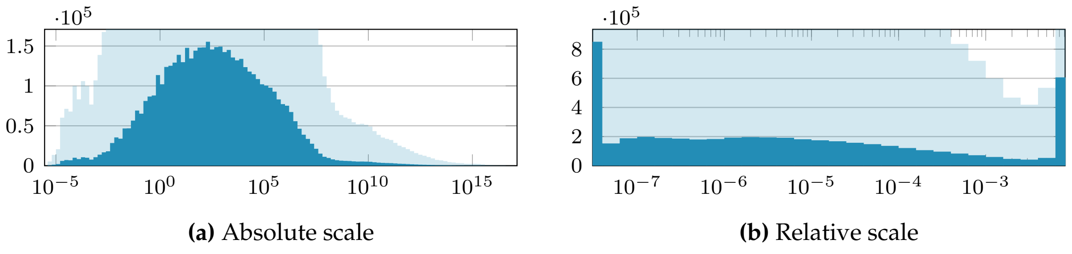
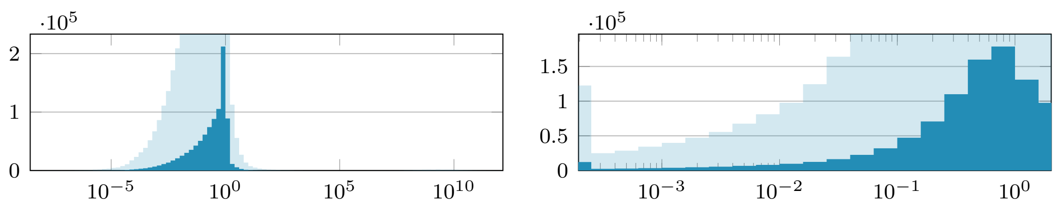


Figure 5.19: Optimal h distribution: Gradient Descent optimized for success, h limited from $3.0 \cdot 10^{-8}$ to $7.8 \cdot 10^{-3}$

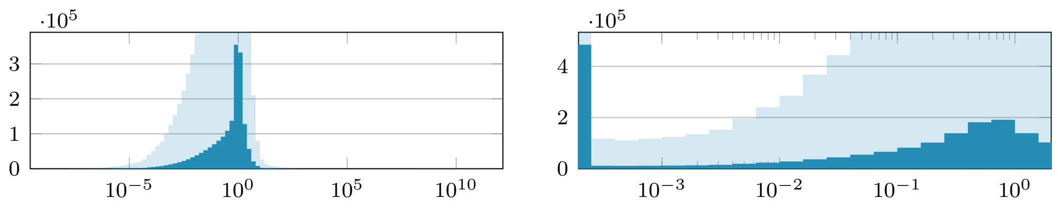
From Figures 5.18 to 5.22 it is evident that the choice of normalizing the update vector leads to very compact h spaces that only span about four to six orders of magnitude and can be searched very quickly. The corresponding distribu-

We found very small search intervals for h in the relative space that lead to high performance.



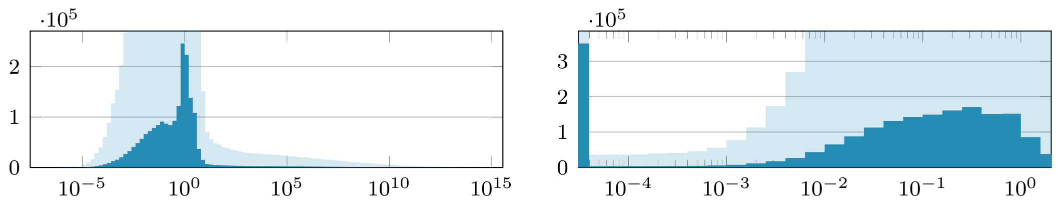
(a) Absolute scale **(b)** Relative scale

Figure 5.20: Optimal h distribution: Gauss-Newton with QR decomposition optimized for speed, h limited from $2.4 \cdot 10^{-6}$ to $2.0 \cdot 10^{0}$



(a) Absolute scale **(b)** Relative scale

Figure 5.21: Optimal h distribution: Gauss-Newton with QR Decomposition optimized for success, h limited from $2.4 \cdot 10^{-6}$ to $2.0 \cdot 10^{0}$



(a) Absolute scale **(b)** Relative scale

Figure 5.22: Optimal h distribution: Levenberg-Marquardt with QR decomposition and diagonal damping optimized for success: h limited from $3.1 \cdot 10^{-5}$ to $2.0 \cdot 10^{0}$

tions of the absolute values often span more than 20 orders of magnitude, which would take around four times longer to search. In addition, it is unclear whether limits in the absolute space would be feasible at all, as we might prevent the algorithm from making the appropriate updates depending on the current scale of the values and update.

5.3 Sensitivity to the h search interval

The conjectures, that we made in the last chapter, are supported by the following simulation data (Figures 5.23

to 5.34), in which we simulated the influence of the lower and upper h search interval bounds on the convergence ratio, expected computation time and average computation times for aborted and successful localization attempts. We did this for all three algorithms, with the fixed parameters chosen based on the best results in Table 5.1. The color gradient in the corresponding figures ranges from blue for good values over green to red for unfavorable values, i.e. longer simulation times and worse convergence ratios. The extra tick lines mark the optimum, which we determined according to Section 5.4.

There is a large plateau of acceptable search interval bounds, indicating that they are relatively uncritical, as long as the lower bound is not too high and the higher bound is not too low.

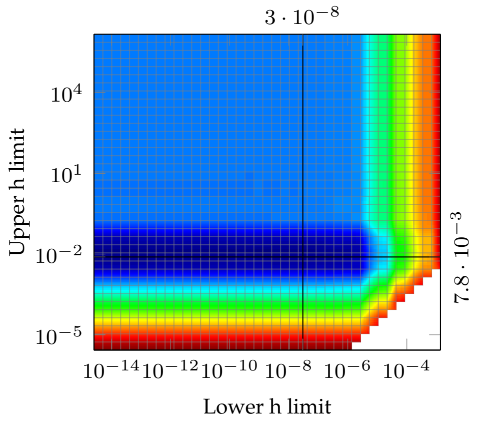


Figure 5.23: Successful Gradient Descent
h limits $\to$ convergence ratio
linear color gradient

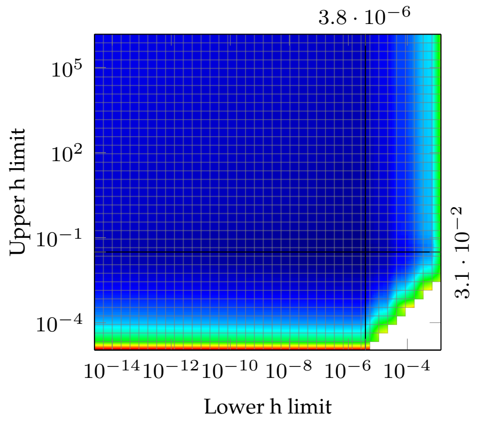


Figure 5.24: Fast Gradient Descent
h limits $\to$ expected computation time
logarithmic color gradient

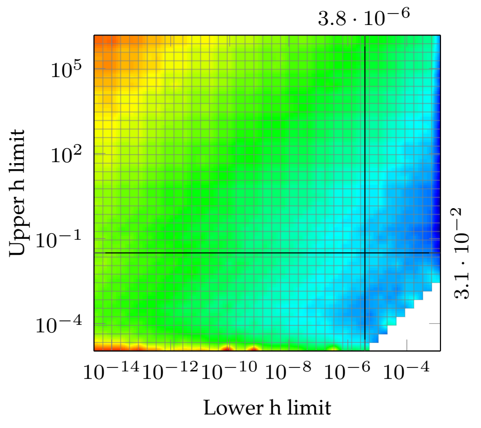


Figure 5.25: Fast Gradient Descent
h limits $\to$ average comp. time on success
linear color gradient

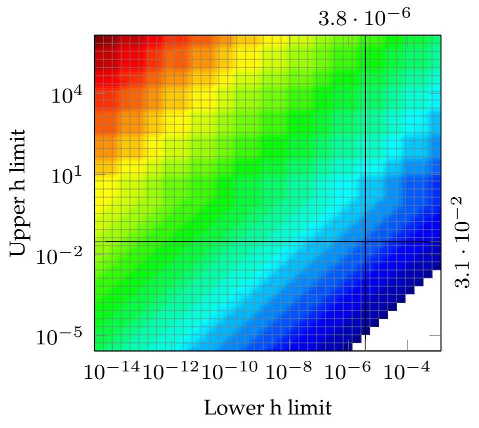


Figure 5.26: Fast Gradient Descent
h limits $\to$ average comp. time on abortion
linear color gradient

We can see the effects of the h search interval bounds on the convergence ratio for the best methods of their class in Figures 5.23, 5.27 and 5.31. For all methods, there are large plateaus of acceptable values with clearly defined borders parallel to the axes. We can deduce that the choice of the lower bound needs to be below a certain threshold to allow for the highest convergence ratio, but only has a slight negative impact on the execution time beyond that. The same is true for the upper limit for Levenberg-Marquardt within the simulated bounds. For Gauss-Newton, the convergence ratio gradually decreases again after a certain value and Gradient Descent has a sharply divided, somewhat small range of optimal values for the upper limit.

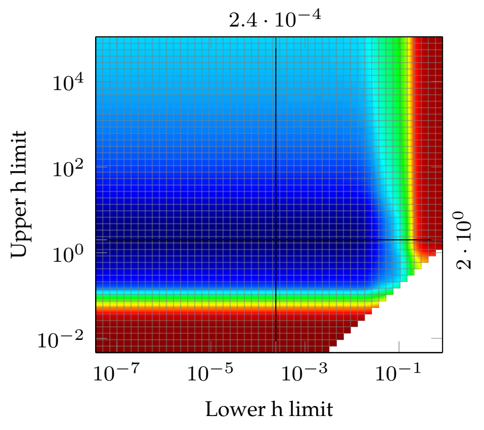


Figure 5.27: Successful Gauss-Newton
h limits $\rightarrow$ success rate
linear color gradient

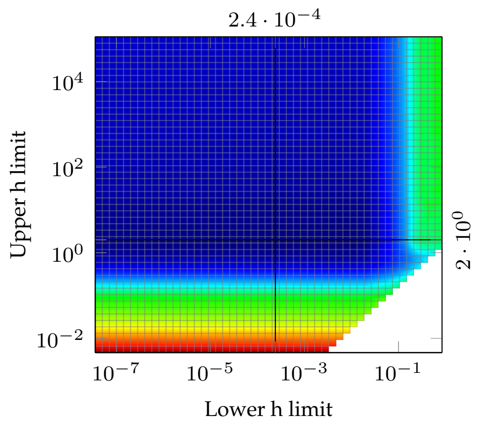


Figure 5.28: Fast Gauss-Newton
h limits $\rightarrow$ expected computation time
logarithmic color gradient

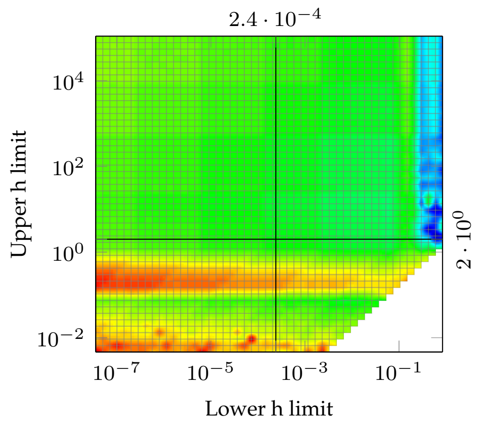


Figure 5.29: Fast Gauss-Newton
h limits $\rightarrow$ average comp. time on success
linear color gradient

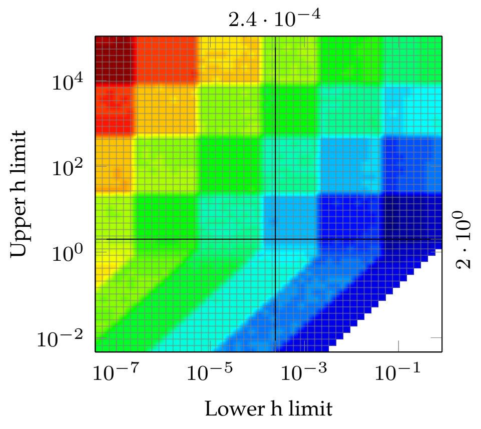


Figure 5.30: Fast Gauss-Newton
h limits $\rightarrow$ average comp. time on abortion
linear color gradient

The influence of the search interval on the expected computation time for the methods whose parameters have been chosen to optimize the speed is depicted in Figures 5.24, 5.28 and 5.32. While the values differ, the general result is the same for all methods in the sense, that there is a large plateau of suitable values. There is a slight tendency towards smaller interval sizes, but the impact on the expected computation time is minimal for all h bounds on the plateaus in the simulated regions.

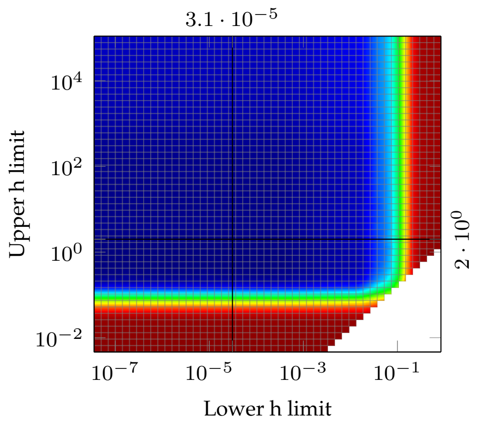


Figure 5.31: Suc. Levenberg-Marquardt
h limits $\rightarrow$ success rate
linear color gradient

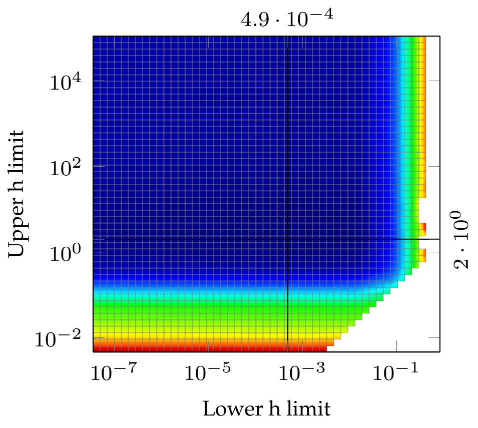


Figure 5.32: Fast Levenberg-Marquardt
h limits $\rightarrow$ expected computation time
logarithmic color gradient

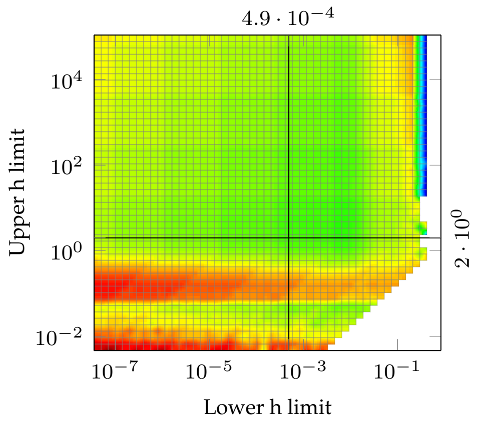


Figure 5.33: Fast Levenberg-Marquardt
h limits $\rightarrow$ average comp. time on success
linear color gradient

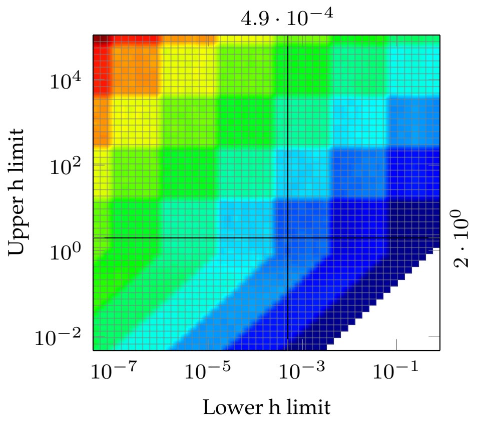


Figure 5.34: Fast Levenberg-Marquardt
h limits $\rightarrow$ average comp. time on abortion
linear color gradient

5.4 Parameter tuning

We use an optimization method to tune the parameters of our localization algorithm based on simulations.

We created a simple optimization program to find the optimal parameters in regard to our evaluation criteria (5.1) and (5.2). To optimize the methods for a high robustness to initial parameters, the h search interval bounds, the Grid Search step size, the number of Golden Section Search steps and the number of Newton-Raphson steps need to be chosen. The step limit is set to 50, as increasing it would essentially always result in slightly higher convergence ratios, making it unsuitable for evaluation criterion (5.1). For the optimization in terms of expected computation time, all line search parameters and the step size are subject to optimization.

We traverse the discretized parameter space in a steepest descent fashion.

The optimization algorithm works as follows: An initial set of parameters is chosen, simulations are run based on these with a wide variety of initial and true values, and the average result in terms of the success metric is recorded. After that, neighbors of the parameter set are generated by increasing or decreasing each individual parameter by one step. For one set of k parameters this results in $2k$ neighbors. The simulations are then performed and evaluated again for the neighboring parameter sets. If one performed better than the original, this set of parameters is chosen as the new optimum and the procedure is repeated from there from the start. The results of repeating this for every method can be seen in Table 5.1.

5.4.1 Results

The top section of the table lists the results for the methods without a line search, i.e. $h = 1$. The middle section shows benchmarks with line search parameters that are chosen to optimize convergence likelihood and the last sections was optimized in regard to the expected execution time.

The columns are, from left to right, the abbreviation for the method, the maximum number of steps until the localization is aborted, the lower and upper limits h_{min} and h_{max} for the line search using a normalized update vector as described in Section 5.2, the step size for the Grid Search GS, the number of steps of Golden Section Search GSS, the number of steps of Newton-Raphson NR, the expected time until a localization was successful t_e, the average amount of steps performed until the method converges $steps_c$, the convergence, abortion and divergences ratios R_c, R_a and R_d and lastly the average time until convergence, abortion or divergence t_c, t_a and t_d.

The methods are abbreviated as follows: Gradient Descent GD, Gauss-Newton GN and Levenberg-Marquardt LM. The matrix decompositions that can be used to solve the equation systems are the Cholesky decomposition CD, QR decomposition QRD and singular value decomposition SVD. The damping strategies for Levenberg-Marquardt are the identity matrix I, the diagonal entries of the Jacobian D and the maximum diagonal entries that have been encountered MD.

We will use the methods without line search as a baseline for the following comparisons. The displayed behaviors are very typical for the respective methods: Gradient Descent converges too slowly and has to be aborted most of the time. Each individual iteration is very fast, but many steps are needed on average to be successful, which effectively makes this the most expensive method of the ones that we evaluated. Gauss-Newton converges very rapidly, with less than seven steps on average until convergence. The convergence ratio is much better than Gradient Descent, but is still only about 30%. Each single step is still performed reasonably quickly though, making this a simple yet viable method for the localization of magnetic dipoles. Levenberg-Marquardt, as expected, boasts the highest convergence ratio of over 80% when solved with the Cholesky decomposition and dampened with one of the diagonal schemes. This comes at the price of needing almost three times as many steps on average compared to Gauss-Newton, though. While we published the execution time benchmarks, it has to be noted that Levenberg-Marquardt was implemented by determining λ with a Grid Search, making it unreasonably slow, but as successful as it can be with a greedy optimization for λ. As a consequence, the execution time benchmarks should not be used to compare Levenberg-Marquardt to the other methods. With that in mind, the baseline Levenberg-Marquardt looks very promising based on the high convergence ratio and low average step count, but including the line search turned out to have drastic effects on the results.

Gradient Descent is unfit for dipole localization.

Baseline Gauss-Newton diverges more often than not.

Levenberg-Marquardt is likely the best choice when used as-is.

method	limit	h_{min}	h_{max}	GS	GSS	NR	t_e	steps$_c$	R_c	R_a	R_d	t_c	t_a	t_d
LM CD MD	50						14.53 ms	18.05	81.68%	18.29%	0.02%	8.97 ms	24.82 ms	7.90 ms
LM CD D	50						12.87 ms	18.57	81.27%	18.57%	0.16%	7.97 ms	21.44 ms	5.04 ms
LM SVD D	50						17.86 ms	18.85	75.68%	23.51%	0.80%	9.78 ms	25.92 ms	2.58 ms
LM CD I	50						8.92 ms	18.97	73.87%	26.02%	0.11%	4.63 ms	12.18 ms	0.73 ms
LM SVD I	50						9.85 ms	18.97	73.87%	26.02%	0.11%	5.11 ms	13.44 ms	0.80 ms
LM SVD MD	50						20.28 ms	19.09	71.59%	26.45%	1.96%	10.28 ms	26.91 ms	2.29 ms
GN SVD	50						0.41 ms	6.82	29.53%	2.28%	68.19%	0.12 ms	0.89 ms	0.09 ms
GN QRD	50						0.28 ms	6.82	29.53%	2.28%	68.19%	0.08 ms	0.60 ms	0.06 ms
GN CD	50						0.33 ms	6.82	29.53%	2.28%	68.19%	0.10 ms	0.73 ms	0.08 ms
GD	50						1,071.40 ms	30.51	0.04%	99.96%	0.00%	0.28 ms	0.47 ms	0.28 ms
LM CD D	50	$3.1 \cdot 10^{-5}$	2	15.5	2	1	27.00 ms	14.78	84.92%	15.08%	0.00%	16.89 ms	56.94 ms	25.41 ms
LM CD MD	50	$2.4 \cdot 10^{-7}$	1	21.0	3	1	29.51 ms	15.67	84.85%	15.15%	0.00%	18.81 ms	59.94 ms	
GN QRD	50	$2.4 \cdot 10^{-4}$	2	15.5	2	1	0.51 ms	8.95	82.35%	17.65%	0.00%	0.24 ms	1.27 ms	
GN SVD	50	$6.1 \cdot 10^{-5}$	2	15.5	2	1	0.59 ms	8.95	82.35%	17.65%	0.00%	0.27 ms	1.46 ms	
GN CD	50	$6.1 \cdot 10^{-5}$	2	21.0	3	1	0.46 ms	8.97	82.32%	17.68%	0.00%	0.22 ms	1.16 ms	
LM SVD D	50	$4.8 \cdot 10^{-7}$	2	15.5	5	1	29.93 ms	15.58	81.78%	18.15%	0.07%	17.46 ms	56.19 ms	4.02 ms
LM CD I	50	$3.0 \cdot 10^{-8}$	2	28.6	4	1	27.51 ms	16.39	80.76%	18.88%	0.36%	16.06 ms	48.95 ms	1.97 ms
LM SVD I	50	$3.0 \cdot 10^{-8}$	2	28.6	4	1	27.25 ms	16.39	80.76%	18.88%	0.36%	15.91 ms	48.49 ms	1.95 ms
LM SVD MD	50	$2.4 \cdot 10^{-7}$	2	15.5	3	2	33.96 ms	15.67	80.68%	19.13%	0.19%	19.34 ms	61.51 ms	12.80 ms
GD	50	$3.0 \cdot 10^{-8}$	$7.8 \cdot 10^{-3}$	11.6	2		47.23 ms	13.23	1.71%	98.29%	0.00%	0.22 ms	0.82 ms	
GN CD	11	$2.4 \cdot 10^{-4}$	2	21.0	4		0.21 ms	6.95	66.52%	33.48%	0.00%	0.12 ms	0.19 ms	
GN QRD	12	$2.4 \cdot 10^{-4}$	2	21.0	4		0.22 ms	7.15	69.25%	30.75%	0.00%	0.13 ms	0.21 ms	
GN SVD	12	$4.9 \cdot 10^{-4}$	2	15.5	4		0.28 ms	7.14	69.23%	30.77%	0.00%	0.16 ms	0.27 ms	
LM CD D	22	$4.9 \cdot 10^{-4}$	2	15.5	2		14.31 ms	12.57	72.67%	27.33%	0.00%	8.65 ms	15.07 ms	8.32 ms
LM CD MD	20	$4.9 \cdot 10^{-4}$	4	15.5	2		15.31 ms	12.19	66.70%	33.30%	0.00%	8.42 ms	13.79 ms	11.84 ms
GD	11	$3.8 \cdot 10^{-6}$	$3.1 \cdot 10^{-2}$	6.6	1		16.20 ms	7.65	1.03%	98.97%	0.00%	0.12 ms	0.17 ms	
LM SVD I	24	$4.8 \cdot 10^{-7}$	2	21.0	2		17.61 ms	13.76	65.41%	34.14%	0.45%	9.23 ms	16.04 ms	1.36 ms
LM CD I	20	$1.9 \cdot 10^{-6}$	4	15.5	2		18.10 ms	12.66	56.86%	42.52%	0.62%	8.30 ms	13.10 ms	0.75 ms
LM SVD D	22	$1.2 \cdot 10^{-4}$	2	21.0	2		19.88 ms	12.83	64.02%	35.90%	0.08%	10.15 ms	17.36 ms	2.64 ms
LM SVD MD	20	$3.1 \cdot 10^{-5}$	4	15.5	3		21.01 ms	12.07	60.87%	38.73%	0.40%	10.21 ms	16.93 ms	4.45 ms

Table 5.1: Average simulation results. Simulated as described in Section 5.1 with about 9.4 million samples per dataset. **Top**: methods without line search, **middle**: line search optimized for success, **bottom**: methods optimized for speed.

When introducing the line search and optimizing the parameters for a high convergence ratio, h is mostly used as damping or very slight boosting term across the board. This is not surprising, as the optimal h in terms of L_{line} is often smaller than one (Figure 5.17). The Grid Search is rather coarse with step sizes between 15 and 30. Golden Section Search is used in all methods with 2 to 5 steps and Newton-Raphson is used in all methods but Gradient Descent to finish the line search. The convergence ratio of Gradient Descent improved markedly by a factor of approximately 40 to a 1.71%, but still leaves it impractical. The main drawback of Gauss-Newton is very effectively addressed by the damping that is introduced by the line search. The divergence ratio dropped from over 68% in the baseline variant to almost 0%, while the convergence ratio improved to over 82%, which is better than the one of the baseline Levenberg-Marquardt. This comes at the cost of a 25% increase in steps and 100% increase in time needed to converge, so there should be a better compromise. Levenberg-Marquardt improves as well, needing about 3 steps less to converge. The convergence ratio improved as well, but only very slightly. With it being only 2% ahead of Gauss-Newton but requiring almost twice as many steps and needing more time per step it is not really worth considering.

Gradient Descent is still impractical, even when combined with a line search. Gauss-Newton's convergence ratio improves, being on par with Levenberg-Marquardt.

Levenberg-Marquardt marginally benefits from a line search.

The most interesting, but at this point least surprising results are obtained when optimizing the line search for expected executing time per localization, as defined in Equation (5.2). With its low convergence rate, Gradient Descent is still completely outclassed, while our implementation of Levenberg-Marquardt is not optimized for speed, and does not provide any comparable data as a result. Considering that a Levenberg-Marquardt iteration will always be more expensive than one of Gauss-Newton, it would would need to converge in fewer steps for it to be faster than Gauss-Newton overall. Considering that λ shifts the update direction towards the one of Gradient Descent, which has been shown to be very ineffective, this seems pretty unlikely. So, the most effective approach according to our simulations is Gauss-Newton, which on average needs only $212\mu s$ to find the result when solved with a QR decomposition with a line search based on a Grid Search followed by a Golden Section Search. Newton-Raphson seemed to be too expensive to warrant its use when optimizing for speed, independently of the method.

Gauss-Newton is by far the most effective approach when paired with a well tuned line search based on a Grid Search followed by a Golden Section Search.

5.5 Sensitivity to initial values

In this chapter we examine the simulation data in respect to the influence that the initial values have on our success metrics (5.1) and (5.2) for the original implementations as a baseline and for the methods that we optimized. The choices of parameters for the latter can be found in Table 5.1. We denote all initial values with a subscript i and all true values with a subscript t. The function $d(x, y)$ is an alias for the Euclidean distance between two points.

5.5.1 Successful methods

The initial p should be chosen with some height over the grid. The choice of m hardly matters, as long as it is not zero.

The influence of the initial values on the convergence ratio is shown in Figures 5.35 to 5.37 for the original methods and in Figures 5.38 to 5.40 for the optimized methods. All of the latter have many things in common in this regard: The sensitivity to the initial x and y coordinates of p is very low. The z coordinate however should not be chosen too small, as this seems to have a strong negative impact on the convergence ratio. The purple graph suggests a dependence on the distance between p and the origin, but if this was the case we would expect a strong correlation with the red or yellow curves, which is not present. We assume that this effect is caused mostly by the dependency on the z coordinate of p. In addition p should not be very close to a sensor, with the bad impact of this likely being caused by the bad condition number of the Jacobian. The most influential factor by far is the distance between the initial and the true p. All methods are much more likely to converge if this distance is small, with Gauss-Newton even reaching a convergence ratio of over 99% if p_i is within less than 2.5cm of p_t.

Both variants of Gradient Descent fail for all initial m that are not identical to the true values. The sensitivity to initial p is reduced a lot by the line search, but the overall convergence ratio is still extremely low.

The sensitivity of Levenberg-Marquardt to the initial values is hardly affected by the addition of the line search. Both the baseline and the optimized variant perform pretty well overall. In contrast to the other methods, it essentially cannot localize dipoles that are less than 2cm away from a sensor and struggles for distances up to around 5cm. The

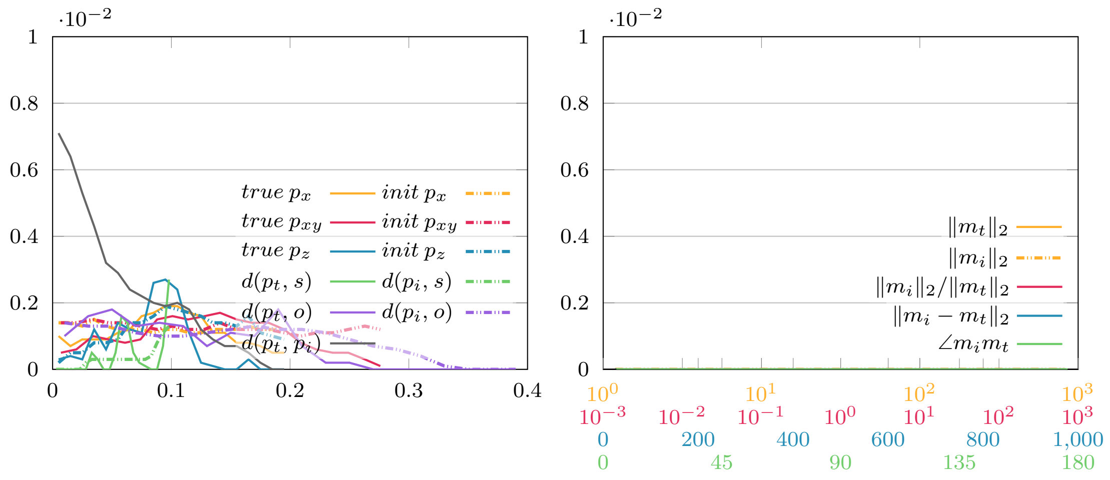


Figure 5.35: Baseline Gradient Descent, convergence ratio

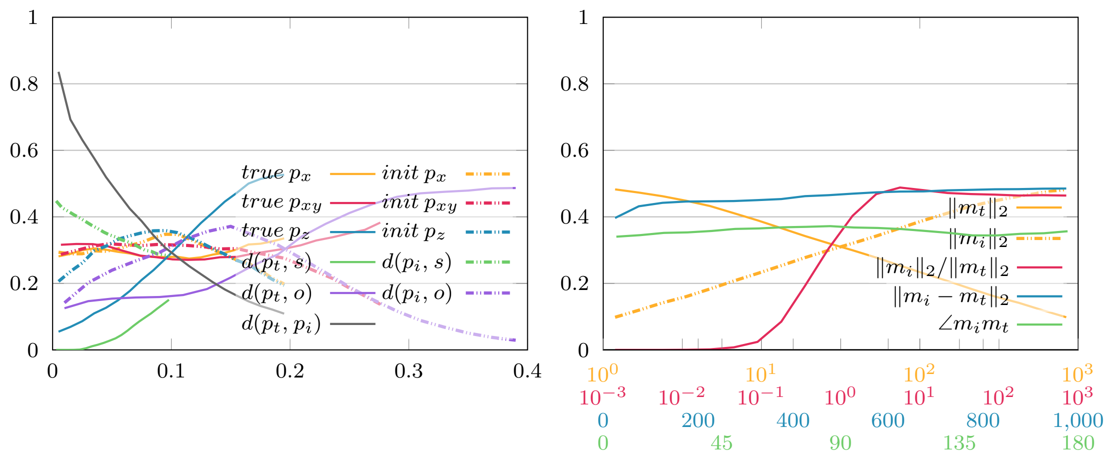


Figure 5.36: Baseline Gauss-Newton, convergence ratio

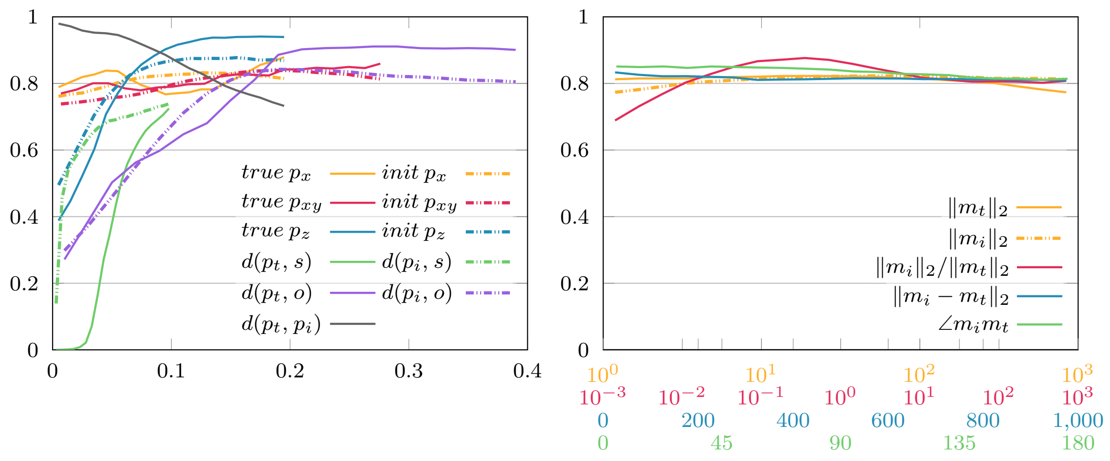


Figure 5.37: Baseline Levenberg-Marquardt, convergence ratio

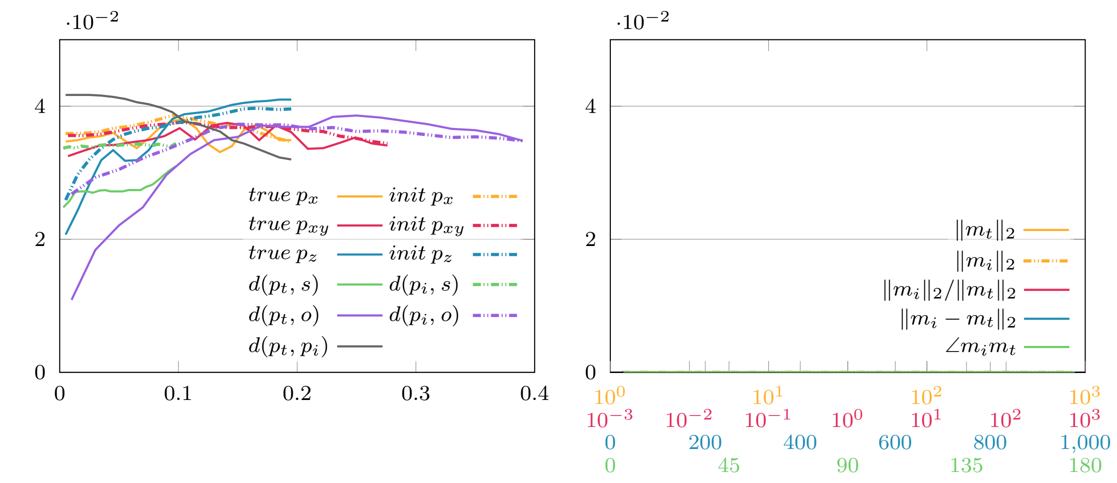


Figure 5.38: Successful Gradient Descent, convergence ratio

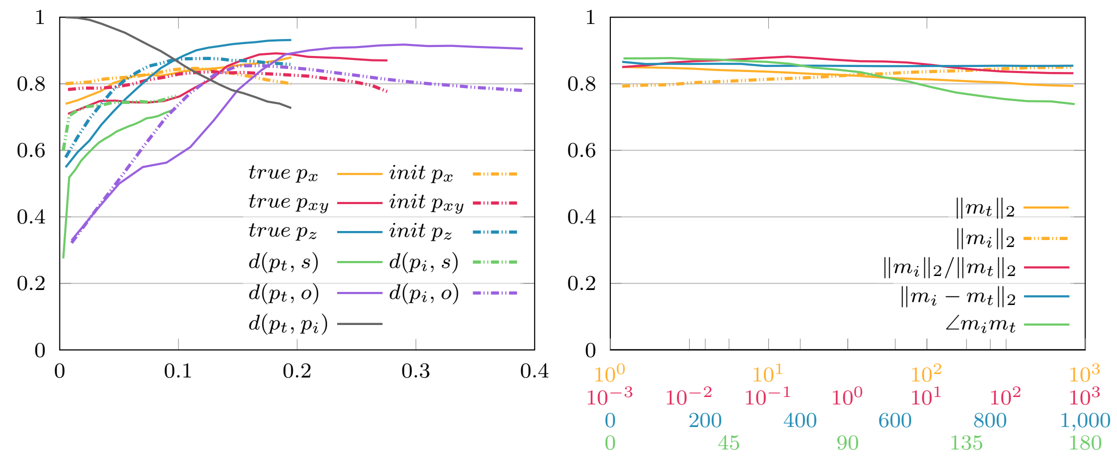


Figure 5.39: Successful Gauss-Newton, convergence ratio

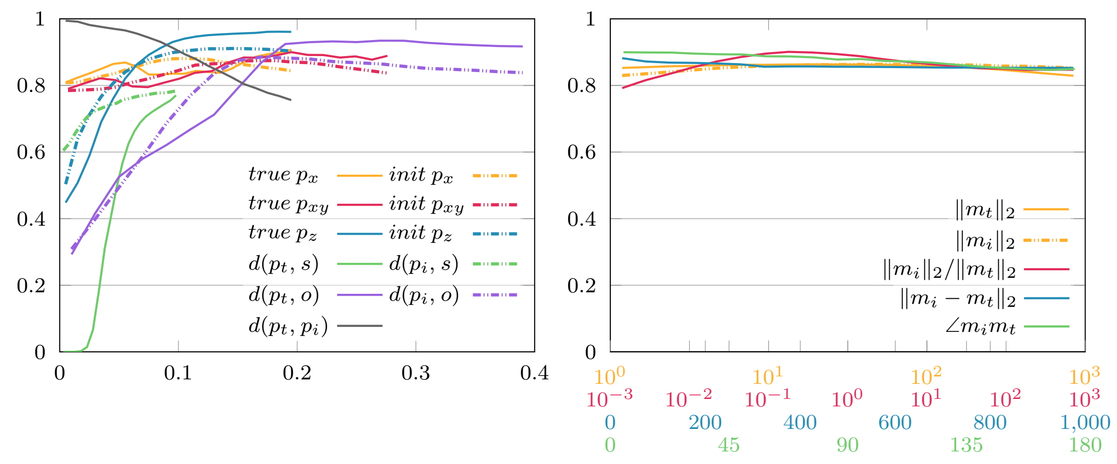


Figure 5.40: Successful Levenberg-Marquardt, convergence ratio

choice of the initial m has rather little influence on the convergence ratio. Only guessing the correct magnitude of m is slightly advantageous, while underestimating it is a bit of a handicap.

Gauss-Newton improved the most by the combination with the line search. For the baseline variant, the initial position had to be close to the true value, while the initial dipole moment had to be larger than or at least close to the true value in magnitude to work somewhat reliably. The improved version shows little sensitivity to most choices of initial parameters and even outperforms Levenberg-Marquardt by having much less trouble localizing dipoles close to a sensor. As with Levenberg-Marquardt, the initial m has very little result on the convergence ratio overall. There might only be a slight dependence on the angle between the initial and true m, with smaller deviations being beneficial.

Gauss-Newton shows little sensitivity to most initial values that are not too close to the grid.

5.5.2 Fast methods

The influence of the initial values on the expected computation time is shown in Figures 5.41 to 5.43 for the original methods and Figures 5.44 to 5.46 for the optimized methods. Overall, the effects turned out to be the same as for the convergence ratio, so refer to Section 5.5.1 for more information.

The effects of the initial and true values on the expected execution time match the ones on convergence ratio.

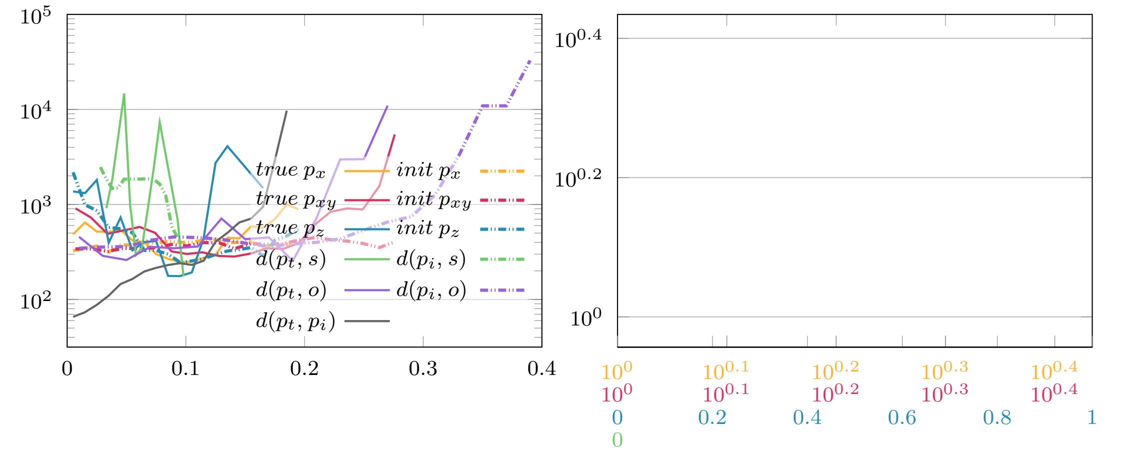


Figure 5.41: Baseline Gradient Descent, expected computation time

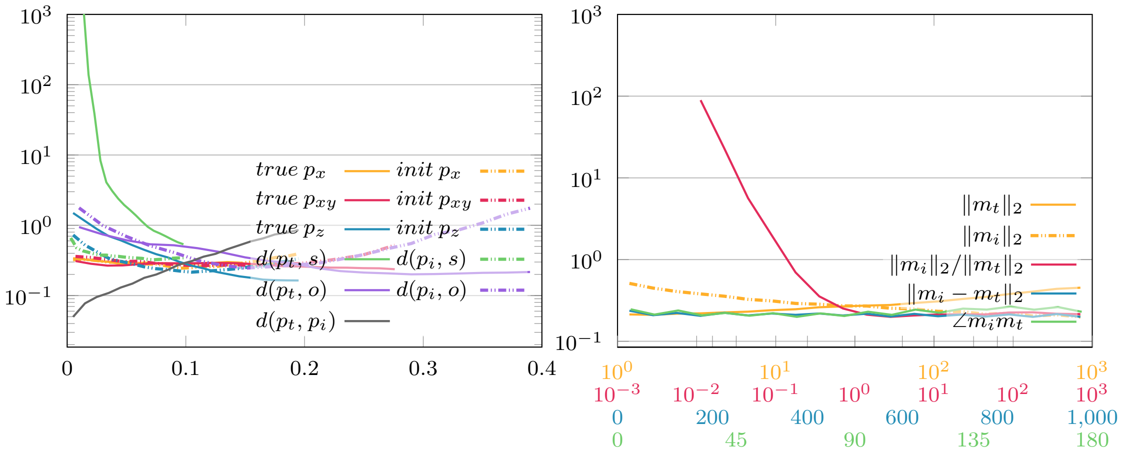


Figure 5.42: Baseline Gauss-Newton, expected computation time

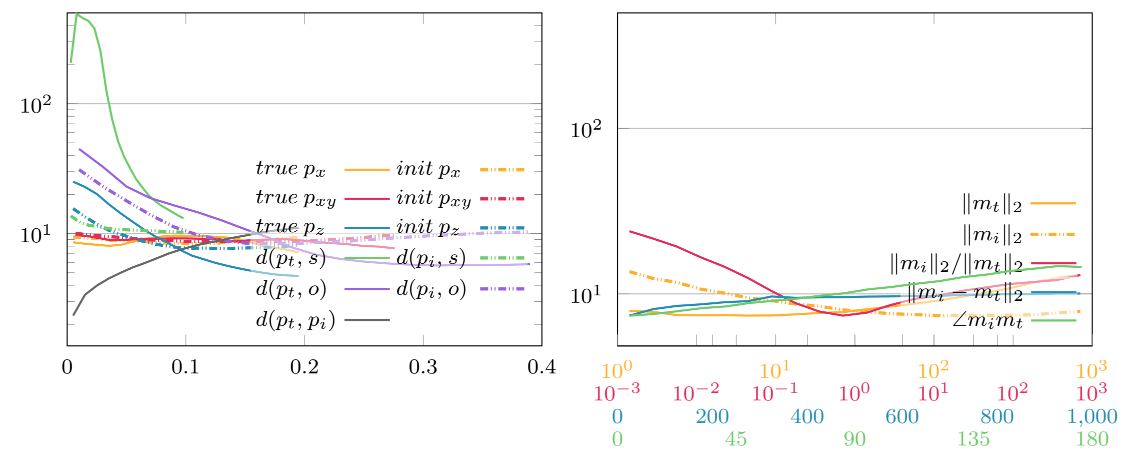


Figure 5.43: Baseline Levenberg-Marquardt, expected computation time

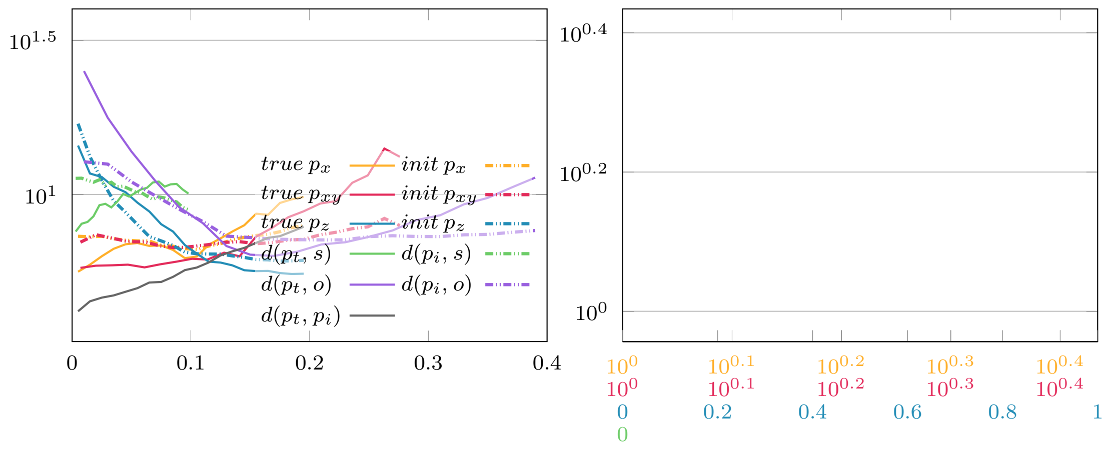


Figure 5.44: Fast Gradient Descent, expected computation time in ms

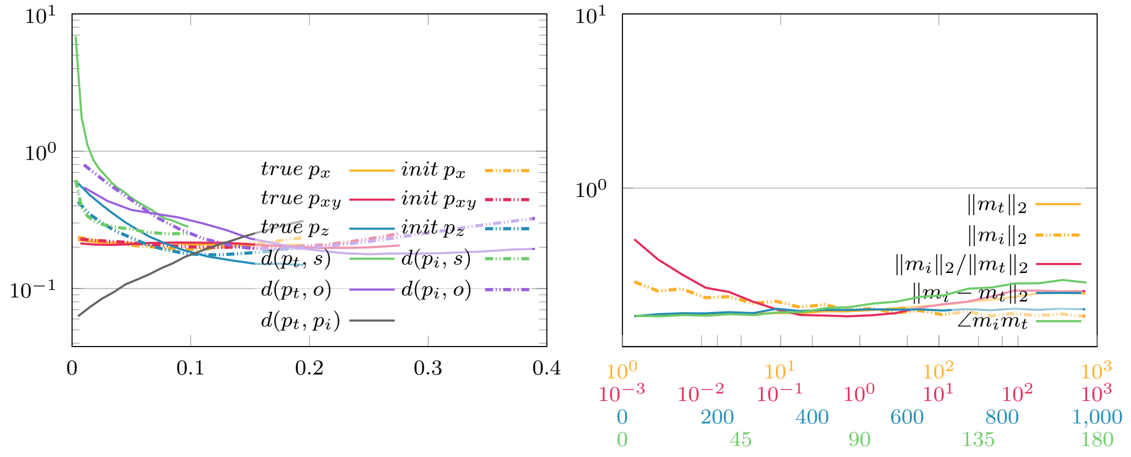


Figure 5.45: Fast Gauss-Newton, expected computation time in ms

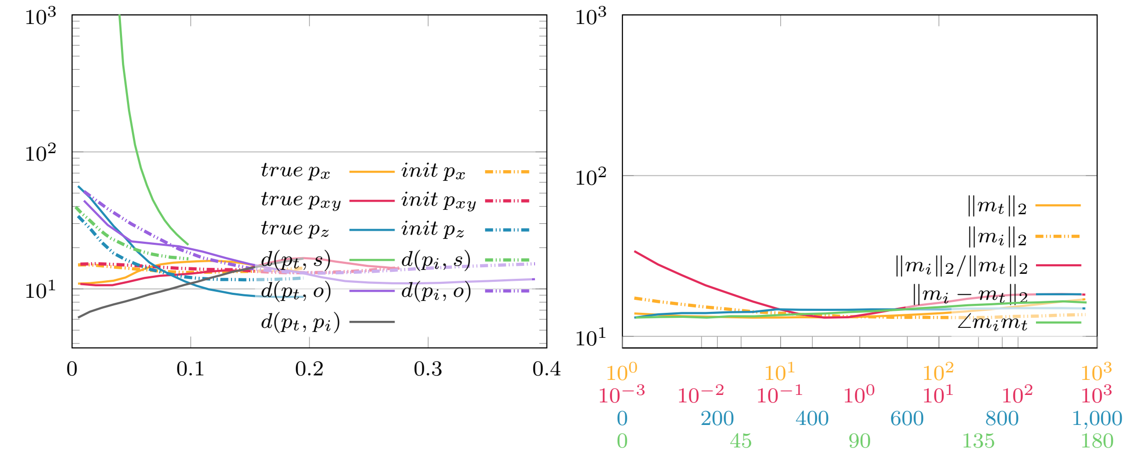


Figure 5.46: Fast Levenberg-Marquardt, expected computation time in ms

Chapter 6

Comparison with related work

In this thesis, we laid the groundwork for a localization system based on AMR sensors and permanent magnet markers. In this chapter, we will compare our system to the ones that we covered in Chapter 2 in terms of cost, size, weight, flexibility, applicability, sensing range, robustness and performance, among other things.

We achieve higher sensing ranges, more flexibility in sensor placement and lower costs than approaches based on graphical algorithms.

The approaches that rely on graphical algorithms [Liang et al., 2012, 2013, 2015, Steurer and Srivastava, 2003, Hook et al., 2009] are restricted to Cartesian grids and require a large number of sensors, making them prohibitively expensive. They are inherently good at multi-object tracking and identification, due to the high resolution, but have low sensing distances. We came up with a modular sensor design, that allows the composition of sparse sensor networks with almost arbitrary topologies, making our system cheaper and more flexible. The sensors can be of higher quality, without impacting the cost as much. This leads to increased sensing distances when combined with a localization algorithm that is based on solving the B-field equations. Multi-object tracking should be possible with our approach as well, but was not tested. The magnetization strength, that is estimated implicitly during the localization as the magnitude of the dipole moment m, might be usable for the identification of objects that are marked with a single magnet. However, whether this is feasible or thwarted by the distortion of the B-field of real magnets compared to ideal dipoles remains to be tested as well.

By using passive, permanent magnet based markers, we can offer more flexibility than approaches with active or externally excited alternatives, but the markers attract or repeal each other.

Fernandez et al. [2020] use active markers that contain the sensors for a wearable tracking system. Having to connect the markers with cables is not too much of a limitation for wearable systems, but limits the practicality for most other applications, which is why we and most related works favor passive markers. Using the sensors as markers also requires an external field source, in this case an excitation coil similar to the one that Hashi et al. [2011] use. The excitation coils typically need to be large, heavy and require a lot of energy, and thus mostly limit the systems that rely on them to static environments with sufficient space, a suitable power source, and no ferromagnetic or sensitive objects in the vicinity. As a consequence, most other works, as well as ours, use permanent magnets that are lightweight, small in size and need no external power source. On the other hand, the LC oscillator based markers used by Hashi et al. [2011] and the sensor based markers used by Fernandez et al. [2020] do have the advantage that they do not attract or repel each other.

The combination of sensors with integrated sampling circuitry and I3C allows us to take all measurements simultaneously.

All of the other approaches, that use more than two sensors, require multiplexers to take measurements sequentially [Schlageter et al., 2001, Hu et al., 2006, 2010, Hook et al., 2009, Liang et al., 2012, 2013, 2015, Steurer and Srivastava, 2003, Hashi et al., 2011]. This creates delay between the measurements, which leads to errors in the data when the markers are moving. Our system uses the I3C bus, which can be used to control over 100 sensors and offers broadcast commands that can be used to precisely synchronize all measurements. Since the sampling circuitry is integrated in our sensors, all measurements can be taken and sampled simultaneously.

There is not enough data for a meaningful comparison of the sensors and the corresponding signal processing chain.

Liang et al. [2013] use WSH138 Hall sensors, Hu et al. [2006] use HMC1053 AMR sensors and we use MMC5633 AMR sensors in our design. Some of the specifications are listed in Table 6.1. As is typical for the corresponding technologies, the Hall sensor has the highest measurement range at around $\pm$200G. As the WSH138 datasheet is very brief, this unfortunately is where our comparison with the Hall sensor ends. Compared to the HMC1053, the measurement range of our sensor is five times higher, which means that magnets can get closer to the sensors without overdriving them. The repeatability errors are the same, considering that the full scale range of the MMC5633 is five times higher than the one of the HMC1053. The linearity error of the MMC5633 is about 40% higher than the one of the HMC1053, but these values are only comparable to a limited extend, as they have been determined over different field ranges. We also lack the data to compare the arguably most important metric, the signal-to-noise ratio.

Part number	Meas. range	Linearity error	Repeatability error
WSH138	±200G	n.A.	n.A.
HMC1053	±6G	1.8%FS	0.10%FS
MMC5633	±30G	0.5%FS	0.02%FS

Table 6.1: B-field sensor specifications

Something, that is not mentioned in any related works, is that the localization can fail for some initial guesses. Hashi et al. [2009] use Gauss-Newton to solve the non-linear equation system, but only examine the accuracy of the results. We found that the original formulation of Gauss-Newton only convergences for around 30% of the initial guesses, or for about 84% of the initial values that are within 5mm of the true result. In fact, we examined the effects of the initial values on all the commonly used methods for solving the dipole equations in great detail and deem none of them robust enough for practical use on their own.

The common iterative approaches are not reliable enough to be used for localization on their own.

By implementing most of the code ourselves, we had the opportunity to optimize the performance and handle some special cases. Hu et al. [2005] used the trust region implementation of Levenberg-Marquardt provided by Matlab, which uses direct methods to determine λ. We implemented and tested this variant as well and found it to be comparatively unsuitable for this particular task, as the optimal λ tends to vary a lot between steps, frequently causing Levenberg-Marquardt to repeat the same optimization steps until a suitable λ is found.

We found that implementation details have a large impact on the overall performance.

Only few of the related works published benchmarks for their localization algorithm. For most of the works, we could find localization rates for the whole systems (see Table 2.1), but oftentimes it is not stated where exactly the bottleneck is. As a whole, we expect our system to be able to refresh the localization 1000 times per second, which is 10 times faster than the fastest related system that we found. Hu et al. [2005] found in ideal simulations with five 1D sensors that a localization with Levenberg-Marquardt can be performed in about $100ms$. On average, our simulations with comparable initial guesses finished in only $62\mu s$. It has to be noted though that our simulations were performed on faster hardware and assume four 3D sensors, which might or might not increase the convergence rate. In 2019, Lu et al. implemented the improved algorithm of Hu et al. [2008], that consists of a linear algorithm, whose result is used

Our system is extremely fast.

as initial guess for Levenberg-Marquardt. Like Hu et al., Lu et al. use the Matlab implementation of Levenberg-Marquardt, and publish benchmarks based on a real system with 24 3D B-field sensors and a permanent magnet that is moved along a space-filling curve by a robot arm. According to their results, with calculations performed on modern hardware, a localization with a combination of the linear algorithm and Levenberg-Marquardt still takes $62ms$, which would mean that our algorithm converges around 1000 times faster, when it does. They propose a faster alternative, based on a singular value truncated supervised descent method, which offers results of similar quality, but only needs $38ms$ on average per localization. This is still much slower than our proposed method.

Chapter 7

Conclusion

7.1 Summary

In this work, we designed and evaluated hardware and software as the basis for a permanent tracking system that is optimized for HCI applications such as gesture detection.

The hardware is compact, lightweight, energy efficient, affordable, fast and the sensors have a high sensitivity and resolution.

The hardware consists of a controller and a sensor network, that can be arranged in terms of size and sensor number as needed, to fit the requirements of the application. At 19mm$\times$19mm$\times$6.7mm, the sensor modules are rather small in size, so that they are even mountable in places where space is at a premium. Each sensor only needs 4mA of supply current for 100 measurements per second, or down to 1mA when noise in the measurements is of little concern, so it easy to supply the system with power via a standard USB 2.0 connection or a battery, even for large sensor networks. The maximum number of sensors is only limited by the 7-bit address space and the baud rates of I3C. This means that networks with around 100 sensors should be possible, as long as the measurement rate is chosen accordingly. In addition, the sensors can measure field strengths of up to 30G with a resolution of up to 20 bits and, RMS noise levels of under 2.5mG at sampling rates of 75Hz. If high sampling rates are more important than low noise, the latter can be increased up to 1kHz. The controller is very versatile and might be fast enough to perform the localizations itself. While our evaluation board turned out to be pretty large, it should be possible to design a non-prototype controller PCB for the 100-pin version of the LPC5536 and its integrated USB peripheral, that is almost the size of just the controller, with all other parts on the bottom side.

A localization based on a minimization problem using the dipole equations, solved with Gauss-Newton and a line search based on Grid Search and Golden Section Search is very fast.

For the localization, we assume the B-fields of permanent magnets to be sufficiently similar to that of ideal dipoles, so that we can use the corresponding equations to set up a quadratic loss function. We have attempted to solve the resulting non-linear optimization problem with three different iterative approaches: Gradient Descent, Gauss-Newton and Levenberg-Marquardt. The latter two require inverting a matrix, which is usually avoided by rearranging and solving the equation system with a matrix decomposition.

We tested three common methods and compared their performance: the Cholesky decomposition, the QR decomposition and the singular value decomposition. In addition, we have combined the methods with a line search to reduce their sensitivity to the starting conditions and to increase the convergence rates. The direction for the line search is generated by the general approach and normalized to be of the same magnitude as the current values. The scale is then determined by a combination of three methods: Grid Search, Golden Section Search and Newton-Raphson.

We ran several simulations for each combination of methods to determine the parameters for the line search that optimize either the robustness to the initial conditions or the expected time until the localization is successful when the possibility of repeated attempts after failures is taken into account. We found Gradient Descent to be unsuited for the localization problem under all conditions. Levenberg-Marquardt is applicable, robust to poor choices of initial values and sufficiently fast when used in its original form as a trust-region method, but the choice of the optimal λ parameter is difficult and varies a lot between steps, making direct methods of controlling λ a poor choice. Combining a Levenberg-Marquardt that always chooses the optimal λ with a line search lead to no improvements in expected execution time, and only little improvements in robustness to starting conditions. Gauss-Newton in its original form is rather fast, but very sensitive to the initial values. By combining it with a line search, we found that it almost matches the robustness of the ideal Levenberg-Marquardt, while needing more than 40% fewer steps to converge on average.

For starting conditions in the general vicinity of the sensor grid, the expected time until a successful localization is performed is just $212\mu s$. For initial positions, that are within 3cm of the actual position, the algorithm is even faster, needing only less than $100\mu s$ for a localization. This is relevant, because it can be expected to almost always be the case after the first localization in tracking applications.

7.2 Limitations and future work

To obtain a reliable system, a meta algorithm is needed in addition to our improved Gauss-Newton, which chooses the initial values and potentially restarts the localization with different parameters after failures. This might be a simple random restart scheme, something more complex that takes velocities and accelerations into account like the Unscented Kalman filter, or perhaps something as versatile as a particle filter.

A meta algorithm that uses Gauss-Newton must be chosen to ensure the success of localizations.

All our simulations have been performed without added noise, errors, offsets or quantization, and we used ideal dipoles instead of approximating the fields of permanent magnets more closely, so it is unclear whether the results are still valid under real conditions. In addition, we only performed simulations with four sensors in one particular configuration. Other placements and the addition of more sensors is likely to affect our results in various ways and should be looked into. On the same note, real-world tests on the actual hardware are pending as well.

More realistic simulations and tests with the real system are needed.

The line search can potentially be improved slightly by employing other (especially direct) methods. A 2D search with different scaling parameters for p and m might also be promising, considering that p and m often differ by two or three orders of magnitude.

It might be possible to speed up the localization by using a higher order Taylor approximation of the residuals. As this requires calculating a tensor with many values in each step, this does however seem unlikely. A definitely needed addition for real applications is the inclusion of the earth's magnetic field in the equation system. While it is possible to make an initial measurement for calibration and to subtract the measured fields as offsets, estimating the earth's magnetic field in every step makes the system more robust to orientation changes of the whole sensor grid. In addition, the measurement of the earth's magnetic field enables the estimation of the sensor orientation deviations, which are already significant due to the small size of the sensor,

The earth's magnetic field needs to be added to the minimization problem.

even if the modules as a whole were to be orientated perfectly.

The number of markers must be known in advance for multi object tracking.

Multi-object tracking and identification should be possible with our approach by adding additional pairs of p and m to the equation system. For this, the number of markers must, however, be known in advance.

Appendix A

Examples of $L_{line}(h)$

In the following, we present some examples of $L_{line}(h)$ that we found during our simulations. $L_{line}(h)$ is plotted in yellow, $\frac{dL_{line}(h)}{dh}$ in red and $\frac{d^2L_{line}(h)}{dh^2}$ in blue.

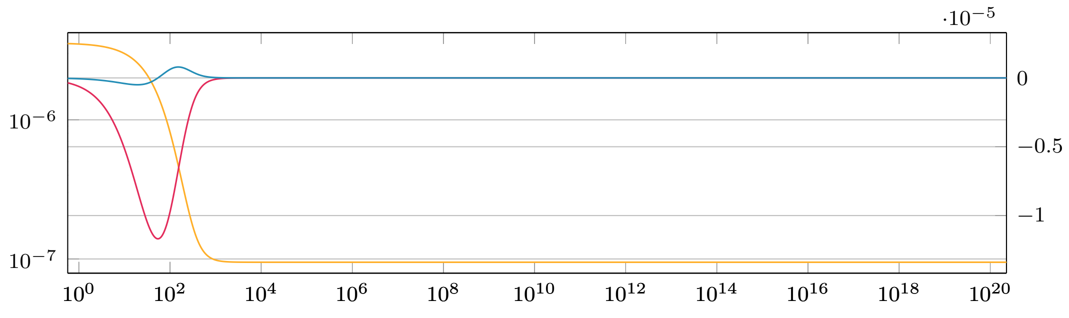


Figure A.1: Example of $L_{line}(h)$, no minimum

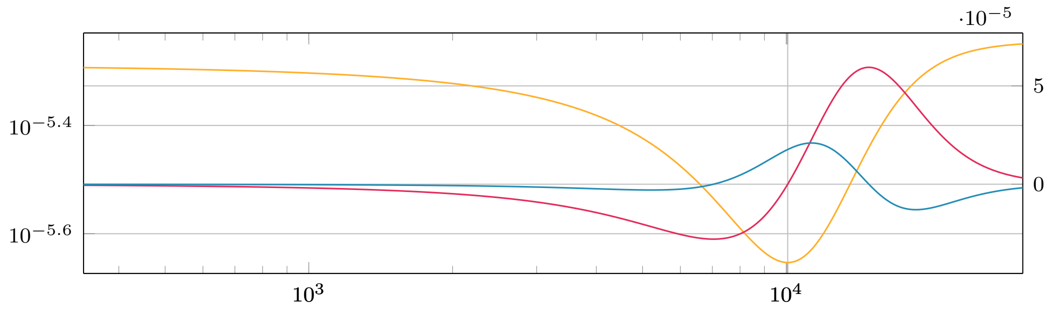


Figure A.2: Example of $L_{line}(h)$, one minimum

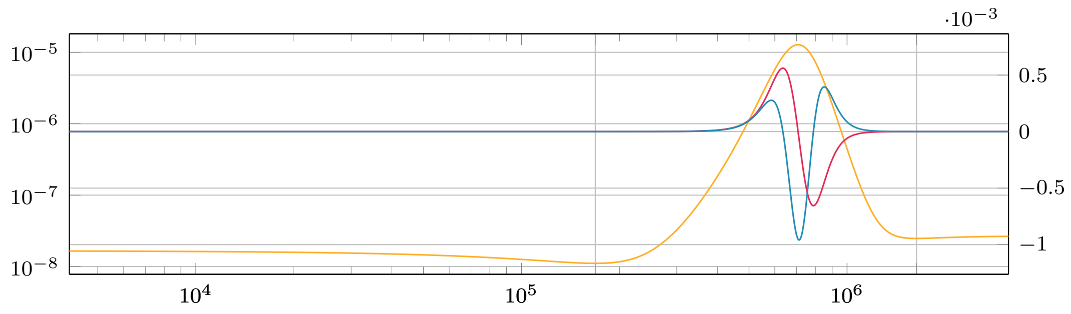


Figure A.3: Example of $L_{line}(h)$, two minima

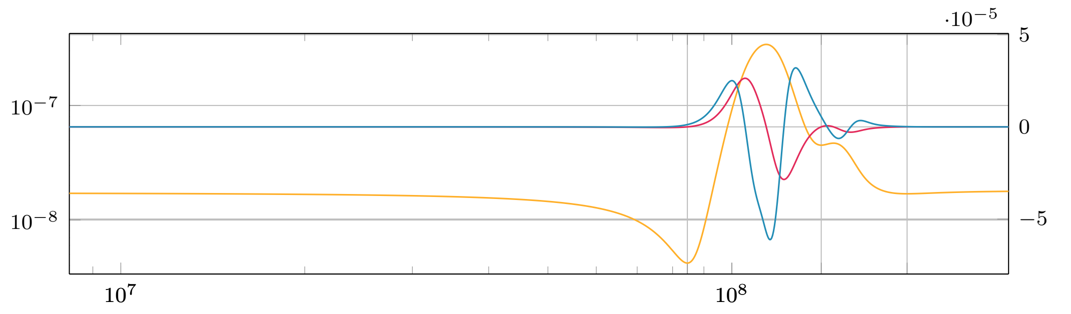


Figure A.4: Example of $L_{line}(h)$, three minima

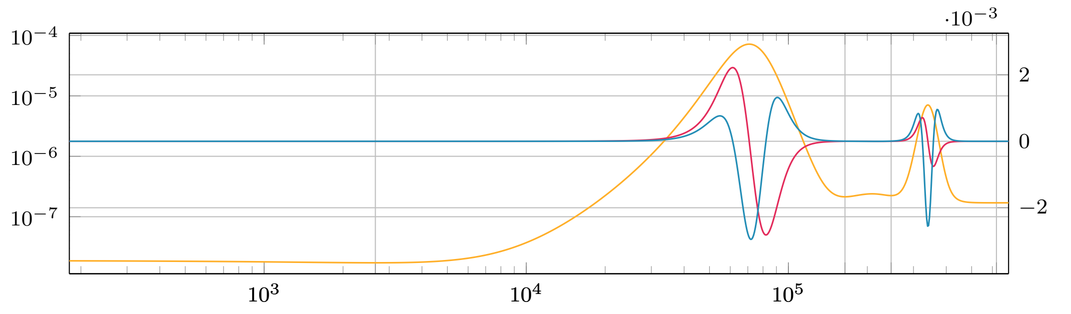


Figure A.5: Example of $L_{line}(h)$, four minima

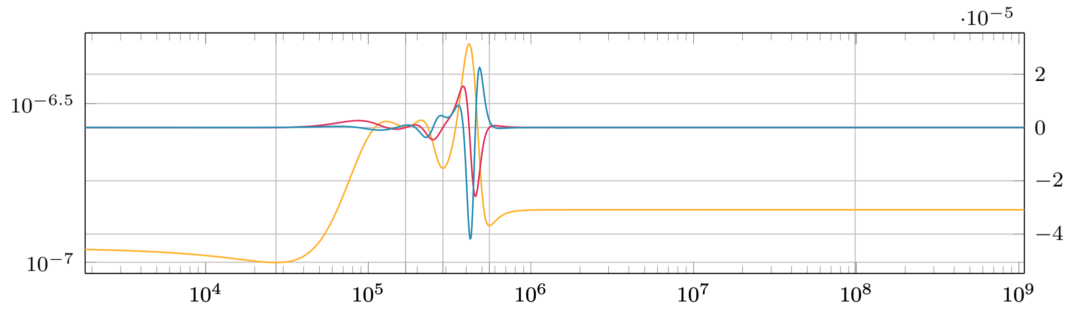


Figure A.6: Example of $L_{line}(h)$, five minima

Bibliography

Neculai Andrei. *Modern Numerical Nonlinear Optimization*. Springer International Publishing, Cham, 2022. ISBN 978-3-031-08720-2. doi: 10.1007/978-3-031-08720-2. URL `https://doi.org/10.1007/978-3-031-08720-2`.

H. Blanchard, L. Chiesi, R. Racz, and R.S. Popovic. Cylindrical hall device. In *International Electron Devices Meeting. Technical Digest*, pages 541–544, 1996. doi: 10.1109/IEDM.1996.554041. URL `https://doi.org/10.1109/IEDM.1996.554041`.

Universal Serial Bus Specification. Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC and Philips, 4 2000. Revision 2.0.

Yadolah Dodge. *The Concise Encyclopedia of Statistics*. Springer New York, New York, NY, 2008. ISBN 978-0-387-32833-1. doi: 10.1007/978-0-387-32833-1. URL `https://doi.org/10.1007/978-0-387-32833-1`.

Laura Drescher-Manaa. Designing a magnetic field sensor grid for 2d mid-air gesture recognition. Bachelor's thesis, RWTH Aachen University, Aachen, September 2022.

David Fernandez, Paolo Motto Ros, Danilo Demarchi, and Marco Crepaldi. A low-complexity 6dof magnetic tracking system based on pre-computed data sets for wearable applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, PP:1–14, 06 2020. doi: 10.1109/TCSI.2020.2998221. URL `https://doi.org/10.1109/TCSI.2020.2998221`.

Xinying Han, Hiroaki Seki, Yoshitsugu Kamiya, and Masatoshi Hikizu. Wearable handwriting input device using magnetic field: Geomagnetism cancellation in position calculation. *Precision Engineering*, 33(1):37–43, 2009. ISSN 0141-6359. doi: 10.1016/j.precisioneng.2008.03.008. URL `https://doi.org/10.1016/j.precisioneng.2008.03.008`.

S. Hashi, S. Yabukami, H. Kanetaka, K. Ishiyama, and K. I. Arai. Numerical study on the improvement of detection accuracy for a wireless motion capture system. *IEEE Transactions on Magnetics*, 45(6):2736–2739, 2009. doi: 10.1109/TMAG.2009.2020541. URL `https://doi.org/10.1109/TMAG.2009.2020541`.

S. Hashi, S. Yabukami, H. Kanetaka, K. Ishiyama, and K. I. Arai. Wireless magnetic position-sensing system using optimized pickup coils for higher accuracy. *IEEE Transactions on Magnetics*, 47(10):3542–3545, 2011. doi: 10.1109/TMAG.2011.2154313. URL `https://doi.org/10.1109/TMAG.2011.2154313`.

Jonathan Hook, Stuart Taylor, Alex Butler, Nicolas Villar, and Shahram Izadi. A reconfigurable ferromagnetic input device. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, page 51–54, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605587455. doi: 10.1145/1622176.1622186. URL `https://doi.org/10.1145/1622176.1622186`.

Chao Hu, Max Qinghu Meng, and M. Mandal. Efficient magnetic localization and orientation technique for capsule endoscopy. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 628–633, 2005. doi: 10.1109/IROS.2005.1545490. URL `https://doi.org/10.1109/IROS.2005.1545490`.

Chao Hu, M.Q.-H. Meng, M. Mandal, and Xiaona Wang. 3-axis magnetic sensor array system for tracking magnet's position and orientation. In *2006 6th World Congress on Intelligent Control and Automation*, volume 2, pages 5304–5308, 2006. doi: 10.1109/WCICA.2006.1714082. URL `https://doi.org/10.1109/WCICA.2006.1714082`.

Chao Hu, Max Q.-H. Meng, and Mrinal Mandal. A linear algorithm for tracing magnet position and orientation by using three-axis magnetic sensors. *IEEE Transactions on Magnetics*, 43(12):4096–4101, 2007. doi: 10.1109/TMAG.2007.907581. URL `https://doi.org/10.1109/TMAG.2007.907581`.

Chao Hu, Wanan Yang, Dongmei Chen, Max Q.-H. Meng, and Houde Dai. An improved magnetic localization and orientation algorithm for wireless capsule endoscope. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2055–2058, 2008. doi: 10.1109/IEMBS.2008.4649596. URL `https://doi.org/10.1109/IEMBS.2008.4649596`.

Chao Hu, Mao Li, Shuang Song, Wan'an Yang, Rui Zhang, and Max Q. H. Meng. A cubic 3-axis magnetic sensor array for wirelessly tracking magnet position and orientation. *IEEE Sensors Journal*, 10(5):903–913, 2010. doi: 10.1109/JSEN.2009.2035711. URL `https://doi.org/10.1109/JSEN.2009.2035711`.

Rong-Hao Liang, Kai-Yin Cheng, Chao-Huai Su, Chien-Ting Weng, Bing-Yu Chen, and De-Nian Yang. Gausssense: Attachable stylus sensing using magnetic sensor grid. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, page 319–326, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450315807. doi: 10.1145/2380116.2380157. URL `https://doi.org/10.1145/2380116.2380157`.

Rong-Hao Liang, Kai-Yin Cheng, Liwei Chan, Chuan-Xhyuan Peng, Mike Y. Chen, Rung-Huei Liang, De-Nian Yang, and Bing-Yu Chen. Gaussbits: Magnetic tangible bits for portable and occlusion-free near-surface interactions. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, page 2837–2838, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450319522. doi: 10.1145/2468356.2479537. URL `https://doi.org/10.1145/2468356.2479537`.

Rong-Hao Liang, Han-Chih Kuo, and Bing-Yu Chen. Gaussstarter: Prototyping analog hall-sensor grids with breadboards. In *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15 Adjunct, page 49–50, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450337809. doi: 10.1145/2815585.2835511. URL `https://doi.org/10.1145/2815585.2835511`.

Jun Lu, Manxi Xiao, Caibao Zhang, and Zhaoshui He. Robust and fast magnetic dipole localization with singular value truncated sdm. *IEEE Access*, 7:94300–94309, 2019. doi: 10.1109/ACCESS.2019.2928036. URL `https://doi.org/10.1109/ACCESS.2019.2928036`.

J. McFee and Y. Das. Determination of the parameters of a dipole by measurement of its magnetic field. *IEEE Transactions on Antennas and Propagation*, 29(2):282–287, 1981. doi: 10.1109/TAP.1981.1142569. URL `https://doi.org/10.1109/TAP.1981.1142569`.

MIPI. *MIPI I3C Basic Specification*. Mobile Industry Processor Interface Alliance, 6 2021. Specification Version 1.1.1.

NIST. *CODATA RECOMMENDED VALUES OF THE FUNDAMENTAL PHYSICAL CONSTANTS*. National Institute of Standards and Technology, 2018. URL `https://www.physics.nist.gov/cuu/pdf/wall_2018.pdf`.

Pedro Pinies, Juan D. Tardos, and Jose Neira. Localization of avalanche victims using robocentric slam. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3074–3079, 2006. doi: 10.1109/IROS.2006.282247. URL `https://doi.org/10.1109/IROS.2006.282247`.

G Sandhya Rani, Sarada Jayan, and K V Nagaraja. An extension of golden section algorithm for n-variable functions with matlab code. *IOP Conference Series: Materials Science and Engineering*, 577(1):012175, nov 2019. doi: 10.1088/1757-899X/577/1/012175. URL `https://dx.doi.org/10.1088/1757-899X/577/1/012175`.

Vincent Schlageter, PA Besse, Radivoje Popovic, and P. Kucera. Tracking system with five degrees of freedom using a 2d-array of hall sensors and a permanent magnet. *Sensors and Actuators A: Physical*, 92:37–42, 08 2001. doi: 10.1016/S0924-4247(01)00537-4. URL `https://doi.org/10.1016/S0924-4247(01)00537-4`.

P. Steurer and M.B. Srivastava. System design of smart table. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003).*, pages 473–480, 2003. doi: 10.1109/PERCOM.2003.1192772. URL `https://doi.org/10.1109/PERCOM.2003.1192772`.

Mark K. Transtrum and James P. Sethna. Improvements to the levenberg-marquardt algorithm for nonlinear least-squares minimization. *arXiv: Data Analysis, Statistics and Probability*, 2012. doi: 10.48550/arXiv.1201.5885. URL `https://doi.org/10.48550/arXiv.1201.5885`.

Typeset November 20, 2023