Towards UAV-assisted monitoring of onshore geological CO$_2$ storage sites

Florian Poppa

A thesis submitted for the degree of
Doctor of Philosophy
of The Australian National University

July 2014

College of Engineering and Computer Science
The Australian National University
This thesis is dedicated to my parents, 
Gudrun and Manfred Poppa, 
for their love, support and encouragement.
Statement of originality

The work presented in this thesis is an account of research undertaken at the College of Engineering and Computer Science at The Australian National University.

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

Parts of this thesis are described in the following publications which I completed while I was a Doctor of Philosophy student:

- Henry Berko, Florian Poppa, Uwe Zimmer and Andrew Feitz, Testing the application of an Unmanned Aerial Vehicle for CO₂ leak detection at the Ginninderra controlled release facility, CO2CRC Research Symposium, Sandy Bay, Australia, November 2013

Florian Poppa
I would like to thank my supervisor Uwe Zimmer for his valuable advice, guidance and support throughout my thesis. The interesting discussions about science in general, robotics research and photography were also much appreciated. Thank you for reliably piloting the helicopter throughout the experiments. Furthermore, thanks for the opportunity to tutor your concurrent and distributed systems course, and for providing me with the countless number of signatures that I needed throughout the course of my PhD research.

I extend my gratitude to my advisers Navida Kottege, Roy Featherstone, and Jochen Trumpf for proof-reading manuscripts and interesting discussions. Special thanks also goes to Timothy Lam for proof-reading parts of my PhD thesis. I am indebted to Alex Martin for helping me to 3D print various sensor and battery mountings, and for laser cutting the parts of the landing pad ensuring the correct position of the microphones. The tips and tricks you gave me along the way made my life so much easier. A very special thanks goes to Andrew Feitz for the opportunity to work for Geoscience Australia which allowed me to learn more about the topic of carbon capture and storage. The given task and the field trips to Toowoomba and Emerald let me experience what environmental monitoring really means in practice. Special thanks goes to Pamela Horyna at Geoscience Australia and Marie Katselas at the Australian National University for doing their magic in getting me the sheer endless number of signatures I needed for this adventure in an unprecedented time frame. A thanks also goes to Henry Berko and Jack Pittar as well as to all other helpful Geoscience Australia staff for setting up the release experiments, and for the friendly and open work environment. Furthermore, I greatly appreciate the extraordinary efforts of CO2CRC and CSIRO which also made the CO₂ trials possible.

A thank you also goes to the other PhD students in the lab, especially my room-mates at RSISE, for the friendly research environment and interesting discussions. Thanks also to all the friends I found during my trips with the ANU Mountaineering Club and while playing badminton in the Belconnen Badminton Club. You made my stay in Australia a great experience.

I want to take the opportunity to express my deepest gratitude to all the people whose guidance has helped me to pursue the path I have taken. To those who made me interested in math and science in the first place, and to the one who introduced me to robotics.
I would like to thank Chen Kong very much for supporting me throughout the last years, especially in the busy last few months before submission. Thank you for helping me with the initial experiments testing the CO$_2$ sensor and for all those late night snack and dinner deliveries. Thanks for all the little things that you do for me, and for being a friend I can always count on.

Most importantly, I would like to thank my parents Manfred and Gudrun Poppa. You fueled my interest in science and technology from early on, and always encouraged me to pursue the path I have taken. Thank you for having me taught to work hard for the things that I aspire to achieve, and to never give up. You truly have been a constant source of inspiration and support.
Abstract

Scientists all over the world look for solutions to reduce greenhouse gas emissions in an effort to achieve proclaimed emissions reduction targets. An intriguing candidate with the potential to make a substantial contribution to this attempt is **carbon capture and storage (CCS)**. The key advantage of CCS is that it provides the possibility to make a significant impact on the reduction of anthropogenic **carbon dioxide (CO₂)** emissions from power plants and carbon-rich industry processes while maintaining existing fossil fuel energy infrastructure. The technique could therefore be used as a transitional solution until fossil fuels can be eliminated from the energy generation mix, and the energy efficiency of industrial processes as well as appliances and products is further improved.

Like other technologies, CCS comes with its risks and rewards. To minimize possible negative impacts on humans as well as on the environment, it is necessary to understand the risks and to address them accordingly. A range of monitoring solutions for geological CO₂ storage sites is available. However, a cost-effective solution for the regular observation of atmospheric CO₂ concentrations (or tracer concentrations) of large areas above onshore geological CO₂ storage sites has yet to be developed.

This thesis discusses the use of a helicopter **unmanned aerial vehicle (UAV)** to fill this gap. The robot platform and its autopilot are designed to cope with ongoing sensor developments in addition to providing safety features necessary for the beyond line-of-sight operation of the UAV. The design focuses on the use of commercial off-the-shelf components for the aerial platform in order to shorten the development time and to reduce costs. The autopilot does neither enforce a specific helicopter model nor defines a set position estimation unit to be used. Access to the control loop enables low-level extensions like obstacle avoidance to be implemented. The developed solution allows the monitoring of an area of approximately 750 m² with one set of batteries in one altitude with a spatial resolution of 2 m by 2 m. Experiments show that point source leaks of as low as 100 kg CO₂ per day can be detected and their source located.

As opposed to autonomous take-offs of the helicopter UAV, autonomous landings on small dedicated helipads require an accurate localization system. A **time difference of arrival (TDOA)** based acoustic localization system which is based on planar microphone arrays with at least four microphones is proposed. The system can be embedded into the landing platform and provides the accuracy necessary to land the UAV on a helipad of the size of 1 m by 1 m. A review of existing TDOA-based approaches is given. Simulations show that the developed approach outperforms its direct competitors for the targeted task. Furthermore, experimental results with the developed UAV confirm the feasibility
of the introduced method. The effects of the sensor arrangement onto the quality of the calculated position estimates are also discussed.

In order to combine robotic-assisted monitoring solutions and other monitoring strategies (e.g. sensor networks and individual sensors) into a single solution, it is necessary to have a framework which allows next to the measurement data analysis also the management (path changes, robot behavior changes, monitoring of internal robot state) of possibly multiple heterogeneous mobile robotic systems. A modular user interface (UI) framework is proposed which allows robots from different vendors and with various configurations next to individual sensors and sensor networks to be managed from a single application. The software system introduces a strict separation between the robot control software and UIs. UI implementations inside the UI framework can be reused across robot platforms, which can reduce the integration time of new robots significantly. The end user benefits by being able to manage a fleet of robots from various vendors and being able to analyze all the measurement data together in a single solution.
# Table of contents

Statement of originality ................................. v
Acknowledgements ....................................... vii
Abstract .................................................. ix
Table of contents ......................................... xi
Abbreviations and acronyms ........................... xv
Notation and symbols ................................... xvii

1 Introduction and motivation ......................... 1
   1.1 Introduction to CCS ............................... 2
      1.1.1 Concept ...................................... 2
      1.1.1.1 Capture .................................. 2
      1.1.1.2 Transport ................................ 3
      1.1.1.3 Storage .................................. 3
      1.1.2 Costs ........................................ 5
      1.1.3 Risk of leakage ............................... 5
         1.1.3.1 The effects of atmospheric CO$_2$ onto the human and the environment ........................................ 6
         1.1.3.2 Leakage scenarios .......................... 6
      1.1.4 Risk mitigation ................................. 7
         1.1.4.1 Sub-surface monitoring ..................... 8
         1.1.4.2 Near-surface monitoring .................... 9
         1.1.4.3 Atmospheric monitoring .................... 9
      1.1.5 Summary ....................................... 11
   1.2 Problem derivation ................................. 11
   1.3 Robot based solutions ............................. 14
      1.3.1 Related work ................................ 14
      1.3.2 Summary ..................................... 16
   1.4 CO$_2$ sensor the determining factor .............. 16
      1.4.1 Sensor choice ................................ 17
      1.4.2 Platform type ................................ 17
1.5 Contributions of this thesis ............................................. 18

2 The rotorcraft UAV and its autopilot ..................................... 23
  2.1 An autopilot for helicopter UAVs ..................................... 24
    2.1.1 Drawbacks of existing solutions ................................ 24
    2.1.2 Design decisions .................................................. 25
    2.1.3 Architecture ...................................................... 28
      2.1.3.1 Hardware independent section .............................. 28
      2.1.3.2 Hardware dependent section ............................... 29
      2.1.3.3 Timing constraints ......................................... 30
    2.1.4 A helicopter UAV for aerial CO$_2$ monitoring ................. 30
    2.2 Autopilot adjustments ............................................... 35
      2.2.1 Sensor gimbal task ............................................ 35
      2.2.2 Sensor redundancy ............................................. 38
      2.2.3 Measurement FlightPattern .................................... 39
      2.2.4 Fallback FlightPattern ....................................... 41
    2.3 Ground crew ....................................................... 41
    2.4 Summary ............................................................ 48

3 An acoustic localization approach for precise landings on a helipad 51
  3.1 Introduction ............................................................ 52
    3.1.1 Motivation ....................................................... 52
    3.1.2 Related work ..................................................... 53
  3.2 Setting the stage ...................................................... 54
    3.2.1 Introduction to acoustic localization approaches .............. 54
    3.2.2 Data acquisition process ...................................... 55
    3.2.3 Error sources ................................................... 56
    3.2.4 Classification and quantity of TDOA estimates ................. 57
    3.2.5 The Cramér-Rao lower bound and the geometric dilution of precision ........................................ 58
    3.2.6 Optimal sensor placement ....................................... 59
      3.2.6.1 Uniform angular arrays (2D) ............................... 59
      3.2.6.2 Platonic solids and spherical code based array geometries (3D) ............................................... 60
      3.2.6.3 Remarks .................................................... 60
    3.2.7 Implications of J onto the accuracy of the position estimates ... 60
    3.2.8 The consequences of using a reference microphone .......... 62
    3.3 Review of passive TDOA-based position estimators for $\mathbb{R}^3$ .................................................. 63
      3.3.1 Approaches using the hyperbolic model ....................... 64
        3.3.1.1 Foy’s algorithm (Taylor-series) ......................... 64
        3.3.1.2 A misconception about iterative algorithms .......... 66
      3.3.2 Approaches using the linear model ............................ 67
        3.3.2.1 Schmidt’s algorithm: Plane intersection .............. 68
        3.3.2.2 Fang’s algorithm ....................................... 70
        3.3.2.3 Bucher and Misra’s algorithm ......................... 73
3.3.3 Approaches using the spherical model ........................................... 75
  3.3.3.1 Schau and Robinson’s algorithm: Spherical intersection . 76
  3.3.3.2 Smith and Abel’s algorithm: Spherical interpolation . . 77
  3.3.3.3 Friedlander’s algorithm ..................................................... 78
  3.3.3.4 Chan and Ho’s algorithm ................................................. 79
3.3.4 Approaches using the cone model .................................................. 81
  3.3.4.1 The algorithm of Brandstein et al.: Linear intersection . . 81
3.3.5 Summary ..................................................................................... 84
3.4 New TDOA-based position estimators .................................................. 87
  3.4.1 XY coordinates ........................................................................... 88
    3.4.1.1 Method XY-I ...................................................................... 88
    3.4.1.2 Method XY-II .................................................................... 89
    3.4.1.3 Comparison of method XY-I and XY-II ............................... 89
  3.4.2 Improving method XY-II ............................................................... 90
    3.4.2.1 Median ............................................................................... 90
    3.4.2.2 Outlier rejection ................................................................. 90
    3.4.2.3 Weighting .......................................................................... 91
    3.4.2.4 Comparison ........................................................................ 92
  3.4.3 Z coordinate ............................................................................... 93
    3.4.3.1 Method Z-I ......................................................................... 93
    3.4.3.2 Method Z-II ....................................................................... 94
    3.4.3.3 Comparison of method Z-I and Z-II .................................... 94
3.4.4 A matter of trust ............................................................................ 95
3.4.5 Case study: 120° layout ................................................................... 97
3.4.6 Summary ..................................................................................... 100
3.5 Optimal array geometry for the UAV localization task ......................... 101
  3.5.1 Maximum base distance uniform angular arrays .......................... 101
  3.5.2 Uniformly Cramér-Rao lower bound distributed maximum base
distance uniform angular arrays ......................................................... 102
3.6 Performance analysis: Simulation ......................................................... 105
  3.6.1 Direct competitors ...................................................................... 105
  3.6.2 General comparison ................................................................... 111
3.7 Performance analysis: Experiments ..................................................... 114
  3.7.1 Sensor array .............................................................................. 114
    3.7.2 Experiment I - Indoor, hand guided sound source .................... 115
      3.7.2.1 Experimental setup ......................................................... 115
      3.7.2.2 Time delay estimation ..................................................... 115
      3.7.2.3 TDE filtering .................................................................... 117
      3.7.2.4 Localization results ......................................................... 117
    3.7.3 Experiment II - Outdoor, intrinsic sound ................................. 119
      3.7.3.1 Experimental setup ......................................................... 119
      3.7.3.2 Time delay estimation ..................................................... 120
      3.7.3.3 Localization results ......................................................... 123
      3.7.3.4 Summary ........................................................................ 124
3.8 Conclusions .................................................................................... 124
Abbreviations and acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>analog-to-digital converter</td>
</tr>
<tr>
<td>AHF</td>
<td>Autonomous Helicopter Framework</td>
</tr>
<tr>
<td>API</td>
<td>application programming interface</td>
</tr>
<tr>
<td>BSP</td>
<td>board support package</td>
</tr>
<tr>
<td>CCS</td>
<td>carbon capture and storage</td>
</tr>
<tr>
<td>CEP</td>
<td>circular error probable</td>
</tr>
<tr>
<td>CH₄</td>
<td>methane</td>
</tr>
<tr>
<td>CO₂</td>
<td>carbon dioxide</td>
</tr>
<tr>
<td>COTS</td>
<td>commercial off-the-shelf</td>
</tr>
<tr>
<td>CRLB</td>
<td>Cramér-Rao lower bound</td>
</tr>
<tr>
<td>DAC</td>
<td>divide and conquer</td>
</tr>
<tr>
<td>DSP</td>
<td>digital signal processor</td>
</tr>
<tr>
<td>EOR</td>
<td>enhanced oil recovery</td>
</tr>
<tr>
<td>ERCP</td>
<td>Eclipse Rich Client Platform</td>
</tr>
<tr>
<td>FIM</td>
<td>Fisher information matrix</td>
</tr>
<tr>
<td>FPGA</td>
<td>field-programmable gate array</td>
</tr>
<tr>
<td>GCC</td>
<td>generalized cross correlation</td>
</tr>
<tr>
<td>GCS</td>
<td>ground control software</td>
</tr>
<tr>
<td>GDOP</td>
<td>geometric dilution of precision</td>
</tr>
<tr>
<td>GPS</td>
<td>global positioning system</td>
</tr>
<tr>
<td>GUI</td>
<td>graphical user interface</td>
</tr>
<tr>
<td>IDE</td>
<td>integrated development environment</td>
</tr>
<tr>
<td>IO</td>
<td>input / output</td>
</tr>
<tr>
<td>IPCC</td>
<td>Intergovernmental Panel on Climate Change</td>
</tr>
<tr>
<td>LD</td>
<td>linearly dependent</td>
</tr>
<tr>
<td>LI</td>
<td>linearly independent</td>
</tr>
<tr>
<td>LiPo</td>
<td>lithium polymer</td>
</tr>
<tr>
<td>LOCA</td>
<td>location on the conic axis</td>
</tr>
<tr>
<td>LWJGL</td>
<td>Lightweight Java Game Library</td>
</tr>
<tr>
<td>LX</td>
<td>linear intersection</td>
</tr>
<tr>
<td>MCU</td>
<td>microcontroller unit</td>
</tr>
<tr>
<td>ML</td>
<td>maximum likelihood</td>
</tr>
<tr>
<td>MMV</td>
<td>measurements, monitoring and verification</td>
</tr>
<tr>
<td>MSRDS</td>
<td>Microsoft Robotics Developer Studio</td>
</tr>
<tr>
<td>Abbr.</td>
<td>Term</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>MUAA</td>
<td>maximum base distance uniform angular array</td>
</tr>
<tr>
<td>NDIR</td>
<td>non-dispersive infrared</td>
</tr>
<tr>
<td>OS</td>
<td>operating system</td>
</tr>
<tr>
<td>OSGi</td>
<td>Open Service Gateway initiative</td>
</tr>
<tr>
<td>OSLS</td>
<td>one-step least-squares</td>
</tr>
<tr>
<td>PCB</td>
<td>printed circuit board</td>
</tr>
<tr>
<td>PSM</td>
<td>passive seismic monitoring</td>
</tr>
<tr>
<td>PWM</td>
<td>pulse width modulation</td>
</tr>
<tr>
<td>PX</td>
<td>plane intersection</td>
</tr>
<tr>
<td>RC</td>
<td>radio controlled</td>
</tr>
<tr>
<td>RD</td>
<td>range difference</td>
</tr>
<tr>
<td>RFIC</td>
<td>robotics framework interface component</td>
</tr>
<tr>
<td>RMLD</td>
<td>remote methane leak detector</td>
</tr>
<tr>
<td>RMSE</td>
<td>root-mean-square error</td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operating System</td>
</tr>
<tr>
<td>RTOS</td>
<td>real-time operating system</td>
</tr>
<tr>
<td>RUIC</td>
<td>robotics user interface component</td>
</tr>
<tr>
<td>SBC</td>
<td>single board computer</td>
</tr>
<tr>
<td>SI</td>
<td>spherical interpolation</td>
</tr>
<tr>
<td>SLAM</td>
<td>simultaneous localization and mapping</td>
</tr>
<tr>
<td>SMD</td>
<td>surface mounted device</td>
</tr>
<tr>
<td>SNR</td>
<td>signal-to-noise ratio</td>
</tr>
<tr>
<td>SWT</td>
<td>Standard Widget Toolkit</td>
</tr>
<tr>
<td>SX</td>
<td>spherical intersection</td>
</tr>
<tr>
<td>TDE</td>
<td>time delay estimate</td>
</tr>
<tr>
<td>TDLAS</td>
<td>tunable diode laser absorption spectroscopy</td>
</tr>
<tr>
<td>TDOA</td>
<td>time difference of arrival</td>
</tr>
<tr>
<td>TTL</td>
<td>transistor–transistor logic</td>
</tr>
<tr>
<td>UAA</td>
<td>uniform angular array</td>
</tr>
<tr>
<td>UART</td>
<td>universal asynchronous receiver / transmitter</td>
</tr>
<tr>
<td>UAV</td>
<td>unmanned aerial vehicle</td>
</tr>
<tr>
<td>UI</td>
<td>user interface</td>
</tr>
<tr>
<td>UIC</td>
<td>user interface component</td>
</tr>
<tr>
<td>UML</td>
<td>unified modeling language</td>
</tr>
<tr>
<td>UMUAA</td>
<td>uniformly Cramér-Rao lower bound distributed maximum base distance uniform angular array</td>
</tr>
<tr>
<td>URL</td>
<td>uniform resource locator</td>
</tr>
<tr>
<td>USB</td>
<td>universal serial bus</td>
</tr>
<tr>
<td>WGS84</td>
<td>World Geodetic System 1984</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Notation and symbols

\( a \) generic notion for a scalar
\( a \) generic notion for a vector
\( A \) generic notion for a matrix
\( \mathcal{A} \) generic notion for a set
\( \hat{\cdot} \) measured or estimated value
\( (\cdot)^T \) transpose of \( \cdot \)
\( (\cdot)^{-1} \) inverse of \( \cdot \)
\( (\cdot)^\dagger \) Moore-Penrose pseudoinverse of \( \cdot \)
\( (\cdot)! \) factorial of \( \cdot \)
\( \| \cdot \|_2 \) Euclidean norm of \( \cdot \)
\( \text{tr}(\cdot) \) trace of the matrix \( \cdot \)

\( p_i \) position vector of the \( i \)-th microphone
\( p_s \) position vector of the acoustic source
\( \tau_{ij} \) time difference of arrival between the sensors \( i \) and \( j \)
\( c \) signal speed
\( d_{ij} \) range difference between the \( i \)-th and the \( j \)-th microphone
\( D_i \) distance from the \( i \)-th microphone to the acoustic source
\( R_i \) distance from the \( i \)-th microphone to the origin
\( R_s \) distance from the acoustic source to the origin
\( N \) number of microphones of the sensor array in use
\( \Xi \) Cramér-Rao lower bound
\( \Gamma \) geometric dilution of precision
\( \sigma \) standard deviation
\( \sigma^2 \) variance
\( \mathcal{I} \) set of sensor pairs providing the TDOAs used for the position estimation
\( \mathcal{I}_0 \) set of sensor pairs specifying the \( N(N - 1)/2 \) distinct TDOA estimates
\( \mathcal{I}_{LI} \) set of sensor pairs which provide the \( (N - 1) \) linearly independent TDOA measurements
\( \lambda \) wavelength
Introduction and motivation

Chapter summary and structure

This chapter gives a brief introduction into the carbon capture and storage (CCS) process, before focusing on its risks, and currently available tools for the monitoring of onshore geological carbon dioxide (CO\textsubscript{2}) storage sites. The need for cost-effective, large area atmospheric monitoring systems is then discussed. An analysis of available robotics systems which could be used to tackle the task outlines the shortcomings of current approaches. It is argued that a helicopter unmanned aerial vehicle (UAV) can be used for the economical atmospheric monitoring of CO\textsubscript{2} concentrations over onshore geological CO\textsubscript{2} storage sites, if the system provides enough hardware flexibility to allow it to be improved with new CO\textsubscript{2} probes or tracer detectors. To embed the solution into a commercial setup, it is also necessary that sensor redundancy can be implemented on the UAV in order to improve the reliability of the aerial robotic system for beyond line-of-sight flights. Due to the limitations of available helicopter UAVs and their autopilot solutions, the development of a new system is proposed.

As opposed to autonomous take-offs of the helicopter UAV, autonomous landings of the aerial platform on small dedicated landing pads require a localization system with higher accuracy than global positioning system (GPS). An acoustic, time difference of arrival (TDOA) based localization system is proposed to solve this problem while avoiding additional payload on-board of the UAV which could potentially decrease the overall flight time.

In order to integrate the management of multiple robotic monitoring systems and other sensing solutions for the monitoring of onshore geological CO\textsubscript{2} storage sites into a single, comprehensive measurement data analysis system, a modular user interface (UI) framework is proposed. The chapter closes by outlining the contributions of these developments to the atmospheric monitoring of CO\textsubscript{2} at onshore geological CO\textsubscript{2} storage sites as well as to the robotics domain.
1.1 Introduction to CCS

1.1.1 Concept

More than 80% of the world’s energy is currently produced by burning fossil fuels (coal / peat 27.3%, oil 32.4%, and natural gas 21.4%) [1]. CCS is a by the Intergovernmental Panel on Climate Change (IPCC) recognized option to reduce the amount of anthropogenic atmospheric CO\(_2\) emissions. CCS involves the capturing of CO\(_2\) released from carbon-rich energy production and industry processes (e.g. cement manufacture, oil refining, ammonia production, and iron and steel manufacture [2, page 60]) and its subsequent storage in both onshore and offshore geological formations (see Figure 1.1). This work will focus on the onshore geological storage of CO\(_2\).

CCS is also commonly referred to as carbon dioxide capture and sequestration. The process can be split into three stages: capture, transport and storage.

![Figure 1.1: Schematic of CO\(_2\) sources and storage options for CCS (Image courtesy of CO2CRC).](image-url)

1.1.1.1 Capture

There are currently three main methods available to capture CO\(_2\) generated from a primary fossil fuel\(^1\):

- **Pre-Combustion Capture**
  This capture method uses a process to convert the fuel into a gas mixture which is principally composed of hydrogen, carbon monoxide, and oxygen. A second reaction converts the mixture into hydrogen and CO\(_2\). The hydrogen can be burned to produce energy. The resulting waste product is water.

---

\(^1\)For a detailed description of the processes, their limitations as well as a comparison see [3].
• **Post-Combustion Capture**
  Absorption-based, adsorption-based and membrane-based systems can be used for post-combustion CO$_2$ capture. Virtually all near-term solutions are absorption-based systems [4]. In an absorption-based system, a solvent is used to remove CO$_2$ from the exhaust gases after the combustion has taken place. The CO$_2$ is subsequently stripped out the solvent after the CO$_2$ absorption. The solvent can be mostly reused.

• **Oxyfuel Combustion**
  The fossil fuels are burned in oxygen instead of air. The resulting waste products are CO$_2$ and water, which are easily separable.

### 1.1.1.2 Transport

After the capture phase, the CO$_2$ is compressed until it reaches a supercritical state (above 31.1 °C and 7.39 MPa [5, page 179]). Supercritical CO$_2$ has a volume over 300 times less than its gaseous state yet still flows like a gas. The supercritical CO$_2$ is transported at pressures above 8 MPa [6] to a suitable storage site. This can be achieved via trucks, ships or pipelines. CO$_2$ is already moved in pipelines today, predominantly to be used for enhanced oil recovery (EOR). EOR techniques can be used to boost the oil extraction after the primary and secondary production phases have ceased. For CO$_2$-EOR, CO$_2$ is injected into the reservoir where it mixes with the oil, enabling it to flow better within the reservoir, allowing more oil to be recovered.$^2$

While pipelines are an effective way of transportation, they also pose concerns. In case of a release of CO$_2$ from a pipeline due to a crack or leak, a possibly large and highly concentrated CO$_2$ cloud could form and endanger nearby communities and wildlife, especially in low-lying areas (CO$_2$ is denser than air). According to the IPCC, the number of people potentially affected from CO$_2$ pipeline leaks may be greater than the amount of people exposed to risks from CO$_2$ capture and storage facilities [2, page 188]. This can be traced back to the fact that power plants, the primary target of CCS, are usually built close to energy consumers and therefore in proximity to populated areas. Sections of a pipeline transporting CO$_2$ from a power plant to a storage site will therefore also be located close to communities. In contrast, storage sites can ideally be chosen to be a safe distance away from densely populated areas.

### 1.1.1.3 Storage

To store CO$_2$, the supercritical fluid is injected into an underground storage formation. A storage site for CO$_2$ is usually in depths greater than 800 m [10]. In these depths the CO$_2$ is kept in its supercritical state due to the prevailing pressure. By permanently storing CO$_2$ underground, it is kept out of the atmosphere and cannot contribute to the greenhouse effect.

Not all types of rocks are suitable for the storage of CO$_2$. Sedimentary rocks which have been formed by an accumulation of sediments (e.g. minerals, rock fragments, or organic material) can provide pores which can allow air, water, natural gas, oil and CO$_2$ to be stored. Examples of common sedimentary rocks are sandstone, limestone, shale, and dolomite.

$^2$For a detailed introduction of EOR techniques see [7, 8, 9].
Chapter 1. Introduction and motivation

Not all sedimentary rocks have storage capabilities. This depends on the porosity and the permeability of the rock:

- **Porosity** is a measure of the amount of open space (e.g. between grains) contained within a rock. The higher the porosity, the greater the amount of open space.

- **Permeability** is a measure of how well a fluid can flow through a rock. The higher the permeability, the easier it is for the fluid to pass through the rock.

If oil or gas is found in sedimentary rock, it is often referred to as reservoir rock. These rocks have beneficial storage properties in that they provide high porosity. This allows gas or fluid to be stored inside the rock formation. To allow the gas or fluid to move through the formation, the rock needs to have high permeability. This property is necessary for the recovery of oil and natural gas as well as for the injection of CO\(_2\).

The supercritical CO\(_2\) is less dense and has greater buoyancy than oil or water in the reservoir and will migrate upwards through the reservoir rock. To prevent the injected CO\(_2\) from escaping to the surface, a barrier on top of the reservoir rock is necessary. Cap rock, a seal impermeable to CO\(_2\) (e.g. a thick marine mudstone which features low permeability), fulfills this role and locks the stored fluid or gas into the reservoir rock.

Depleted oil and gas fields which formerly held their contents captured for millions of years provide both a reservoir rock with high porosity and permeability and are covered by cap rock with low permeability. They are therefore good candidates for CO\(_2\) storage. Another type of reservoir suitable for CO\(_2\) storage is deep saline aquifers. These reservoir rocks have salty water stored in their pores. This fluid is also called brine and is unfit for drinking purposes. Deep saline aquifers have beneficial properties for CO\(_2\) storage, but only the ones sealed off by a suitable regional cap rock can be considered for CO\(_2\) storage.

It is estimated that CCS has the potential to store at least 80 years’ worth of current CO\(_2\) emissions [11]. However, the storage formations are not equally distributed over the globe. Some countries might therefore have to transport their CO\(_2\) to other countries for storage while others can choose storage sites close to CO\(_2\) producers.

After the CCS injection phase has finished, the injection well will be sealed. A method which is already common practice in natural gas processing can be utilized: cement and steel are used to form a plug [12] which seals the well. Afterwards, the well can be abandoned and followed up with periodic monitoring.
During and after the CO₂ is injected into the formation, additional mechanisms keep it stored safely underground:

- **Residual Trapping**
  As CO₂ is injected, some of it becomes trapped as small disconnected droplets in the porous rock (see Figure 1.2, left).

- **Dissolution Trapping**
  Some of the CO₂ dissolves in the brine. The resulting mix is heavier than the fluid surrounding it and therefore sinks to the bottom of the formation (see Figure 1.2, right).

- **Mineral Trapping**
  The CO₂ can chemically react with the rock of the storage site and form minerals. This process effectively binds the CO₂ to the rock. If such a reaction takes place and how long it needs to finish depends on the chemical properties of the reservoir rock and the brine found in the formation.

### 1.1.2 Costs

The overall costs of CCS are the sum of the individual costs for CO₂ capture, transport and storage. The first part of the CCS process, the capture and compression of CO₂, accounts for more than 75% of the overall costs [13]. The high price (and additional energy required) is expected to decrease with improvements in CO₂ capture technology.

The costs for transportation and storage depend on the distance between the CO₂ producer and the storage site as well as whether existing infrastructure can be reused or not. If e.g. a depleted oil or gas field can be used as storage site, infrastructure, geophysical exploration information, production history, wells and well data will be already available [14]. It is also possible to use existing, decommissioned natural gas pipelines to transport CO₂. Doing so can lower the costs of the transport and therefore the overall costs of CCS. It however depends on the pipeline design, the age of the pipeline and the degree of corrosion of the pipeline if such an undertaking is possible or not [15]. Using the captured CO₂ for EOR can reduce the overall expenditures of the oil production process. The gained savings can be used to offset the costs of CCS.

### 1.1.3 Risk of Leakage

"Open fractures provide pathways for rising carbon dioxide to bypass the impermeable caprock layers, returning to the surface and making CCS at best an expensive waste of time and at worst providing a danger to nearby population."

(From [16, page 86])

The quote by Verdon et al. illustrates that the repercussions of leakages of CO₂ from CCS storage sites can vary vastly. CO₂ could leak into the atmosphere at any stage throughout the CCS process: during capturing, transport, and storage. A leak during the capture phase will likely be detected in an early stage, with the capturing process taking place in an industrial setup with strict health and safety regulations [17]. Leaks in pipelines will likely also be detected in an early stage due to reduced pressure in the transporting system. The likelihood of a pipeline puncture or rupture as well as their effects close to densely populated areas are still subject to discussion (see [18]). The remainder of this
work will narrow its focus on the dangers, risk mitigation and monitoring options of onshore geological storage of CO₂.

### 1.1.3.1 The effects of atmospheric CO₂ onto the human and the environment

The danger of CO₂ to humans depends on its concentration. For low CO₂ concentrations in air, humans start breathing heavily, get headaches and experience a weak narcotic effect (1-3% CO₂ concentration). More severe impacts are dizziness, confusion, impaired judgment, and loss of consciousness (4-10% CO₂ concentration). If humans are exposed to a concentration of more than 10% CO₂ for a longer period of time, death will occur.

The environmental impact of CO₂ also varies with its concentration. Smaller concentration increases in the atmosphere actually increase plant growth, which is why CO₂ is sometimes used in greenhouses to enhance the growth rate of flowers. If the CO₂ concentration reaches a certain threshold, it is also destructive to plants. This threshold is species specific.

### 1.1.3.2 Leakage scenarios

There are two primary scenarios in which leakage can occur [2, page 34]. The scenarios are dependent on how the CO₂ escapes the storage formation. Figure 1.3 showcases and describes the possible CO₂ escape routes.

![Image of CO₂ escape routes](Image courtesy of CO2CRC)

**Figure 1.3**: Schematic representation of possible escape routes for stored CO₂ (Image courtesy of CO2CRC).

---

3 The presented figures are an excerpt from [2, page 392] in which a detailed summary of the consequences of specific CO₂ concentrations on the human body can be found. For further information see also [19] [20].

4 For an overview of CO₂ usage in greenhouses (benefits, concentrations, and problems) see [21].
1.1. Introduction to CCS

In the first scenario, CO$_2$ escapes through the injection well or leaks up an abandoned well. This kind of leakage will be detected quickly due to the rapid release of CO$_2$ and an associated hissing sound, the presence of bubbles, or formation of dry ice in high CO$_2$ flow situations. Workers in the vicinity of the well are endangered. The leak can be fixed with already established techniques used for well blow-outs [2, page 252].

CO$_2$ escaping via routes described by the seal, faults and migration options of Figure 1.3 can be summarized in a second scenario representing leaks which are more challenging to detect. CO$_2$ seeping from the storage site can migrate upwards and mix with drinking-water aquifers (the CO$_2$ itself as well as displaced brine). This may adversely affect groundwater quality, including increasing salinity and the mobilization of trace metals or organic contaminants. In addition, the soil can be subject to acidification and oxygen displacement [22]. If the leak is not detected, CO$_2$ can get into the atmosphere and pose a danger to people and the environment. Remediation measures range from stopping injection to reduce the pressure gradient, changing flow conditions in the reservoir by pumping water out of pressure management wells, pumping the CO$_2$ from the storage site and re-injecting it into an alternative formation, to groundwater extraction and purification [2, page 35].

1.1.4 Risk mitigation

The key to make CO$_2$ storage safe is to carefully choose the right geological storage site with ideally multiple regionally-extensive, thick seals. The best practices for the maturation process of a potential storage site consist of three steps: site screening, site selection and initial characterization [23]. Site screening identifies suitable sub-regions of a large area storage basin. The sub-regions have to provide suitable geologic properties, regional site data (e.g. evaluate distance to nearby population centers) as well as social data (e.g. demographic trends are considered). The sub-regions which provide the necessary properties are ranked and undergo the site selection process. This involves gathering more detailed subsurface geological data, a regulatory analysis, a site suitability analysis, a preliminary social characterization (concerns and benefits to the community) as well as the development of a site model. The potential storage sites which come out of this process are again ranked and form the list of qualified sites. The initial characterization of a qualified site involves a subsurface data analysis, a regulatory issue analysis, a model refinement, an outreach assessment and the development of an initial site plan. These steps ensure that only the most suitable potential storage sites are used for CCS.

After the storage site has been chosen and injection has been started, monitoring ensures that containment breaches are detected in an early stage to allow remediation measures to be taken as soon as possible. The most important monitoring tools are briefly introduced based on where the techniques can detect the CO$_2$ [5]. An overview of the discussed monitoring tools including their advantages and disadvantages is given thereafter in Table 1.1.

---

[5] Due to the extensiveness of the topic, this work focuses on an introduction of the most important monitoring tools available for the monitoring of onshore geological CO$_2$ storage sites. For a detailed review of available monitoring tools based on the properties of the storage site, the reader is referred to [24].
1.1.4.1 Sub-surface monitoring

A risk during the injection phase is that the pressure from the ongoing injection compromises the integrity of the cap rock. That would allow the CO\(_2\) to penetrate the reservoir seal. To avoid this, the injection pressure is always chosen far below the fracture pressure of the reservoir rock. Should however a fracturing occur, it is shown in [16] that passive seismic monitoring (PSM) can be used to detect such an event and its location in the cap rock or reservoir rock.

PSM uses an array of geophones in vertical boreholes or at the surface to detect microseismic events in the formation. The arrays deployed in boreholes can usually detect smaller events and have a better vertical resolution than ground-based arrays [25]. Surface arrays on the other side can cover a wider area.

Based on geomechanical models, parts of the reservoir which are most prone to failure can be determined [26]. The geophone arrays can then be placed to observe the weak points of the storage formation accordingly. Microseismic events in the cap rock will be of highest concern for storage security, signaling a possible fracture of the reservoir seal. In case PSM is used in real-time, it provides the operator of the storage facility with a timely warning in case of a fracture, due to the fact that the CO\(_2\) needs some time to rise towards the surface. PSM is able to detect possible fractures at the time of occurrence, giving this technique the edge over other methods. The time gained by PSM can be used to start appropriate remediation measures in an early stage.

For PSM to be effective, it is necessary that the geophone arrays are installed prior to injection to allow the collection of information about the background seismic activity. This baseline data can also be used to suggest potential spots in the reservoir with high seismic activity, allowing the operator to exclude these positions as possible injection well locations.

Seismic surveys, a similar technique to PSM, can be used to detect the CO\(_2\) plume in the reservoir. Seismic surveys also use geophone arrays on the surface (surface seismic) or in boreholes (borehole seismic, cross-well seismic), however in conjunction with artificial shock waves produced by a drop hammer or seismic vibrator (vibroseis) on the surface or by explosives placed in shallow depth. The shock waves propagate through the formations and get reflected, allowing the creation of an acoustic image of the subsurface based on the results delivered by the geophones. If seismic surveys are carried out repeatedly (time lapse seismic monitoring), it allows to monitor the development and motion of the CO\(_2\) plume in the storage formation. In addition, further seismic surveys after the injection phase can provide proof that the CO\(_2\) is still confined in the reservoir rock.

A further sub-surface monitoring technique which can be used for CCS is called wireline logging. This method involves lowering sensors down a well in order to collect information about the properties of the formations surrounding the well as a function of depth. In the context of CCS, wireline logging can be used to trace the vertical migration of CO\(_2\) in the immediate vicinity of the well. As with PSM, wireline logging is dependent on available baseline data that has to be collected before the injection starts [27].

The pressure put onto the reservoir increases and decreases with the CO\(_2\) injection rate. Pressure monitoring can help to gain insight about a CO\(_2\) leakage through the cap rock as well as about a failure of the cap rock [28]. Wellhead pressure, formation pressure, and above-zone pressure can be determined with sensors in the wellhead and via wireline downhole measurements.
Multiple wells are usually used to produce oil or gas from an oil field. If CO\textsubscript{2} is going to be stored in a depleted oil or gas field, this prior infrastructure has to be monitored. In case the condition of a well or its plug is doubtful, it has to be re-sealed before an injection can take place. During and after the injection phase, well monitoring can ensure that the well seal is intact.

1.1.4.2 Near-surface monitoring

Geochemical monitoring involves regular surveillance of ground water levels / pressures and its gas and water chemistry. Geochemical monitoring can also be applied to monitoring within the storage reservoir at dedicated reservoir monitoring wells. In case of a CO\textsubscript{2} leak in a storage site or displacement of brine from the reservoir, CO\textsubscript{2} or brine can migrate towards the surface and contaminate the ground water. This results in a change of the chemical properties of the water (for a detailed description see [29]). The regular sampling and analysis of the soil gas composition, soil flux measurements as well as gas sampling in the head space of water wells at dedicated locations around the injection site can provide evidence for a leakage. Baseline data has to be collected before the injection begins to allow these techniques to work.

Both satellite imagery as well as photos taken from airplanes can be used to detect subtle changes in the flora. However, natural changes in the vegetation due to environmental factors have to be considered. Increased growth or plant death does not necessarily mean a leak in the CO\textsubscript{2} storage site. An additional problem is that changes in the ecosystem need time. These indicators are therefore late indicators of a containment breach. In addition, the emitted CO\textsubscript{2} cannot be quantified based on the changes in the flora beyond a specific point.

1.1.4.3 Atmospheric monitoring

Atmospheric sensors can be used to measure the CO\textsubscript{2} concentration on the surface. Some technologies (e.g. non-dispersive infrared (NDIR) sensors) allow the operator to continuously gain measurements in real-time. These sensors are however only useful for spot checking, with measurements only taken at the deployment position.

Hyperspectral imaging systems working in the short-wave infrared range allow CO\textsubscript{2} to be detected. Such systems have already been used in practice, e.g. to map the CO\textsubscript{2} concentration of a volcanic plume [30]. However, this method suffers from potential spectral overlap (e.g. water vapor) and therefore requires additional environmental measurements for calibration [31].

Airborne sensing (low-level airplane surveys) can cover a large area and can still provide a good accuracy of the CO\textsubscript{2} concentration close to the ground [32]. The costs of this method however make this monitoring technique only useful for regular monitoring in large time intervals (see also [32]).

Remote sensing technologies can also be used to measure the CO\textsubscript{2} concentration. Satellites equipped with high-resolution spectrometers can deliver results for vast areas. However, the large surface footprint of each measurement in combination with the dilute, localized nature of a CO\textsubscript{2} surface leak currently makes the use of satellites unsuitable for routine leak detection from geological CO\textsubscript{2} storage sites. NASA launched the OCO-2 satellite on 2 July 2014 to monitor atmospheric CO\textsubscript{2} concentrations. OCO-2 has a surface
Table 1.1: Pros and cons of available monitoring tools for onshore CO\textsubscript{2} storage. A general disadvantage of near-surface and atmospheric measurements is that they are influenced by environmental changes (e.g. seasons, day and night). Exhaust gases and additional impacts from nearby industry and farming are also evident in the measurements. Extensive baseline data has therefore to be collected before the injection starts. It is important that the baseline data captures the full range of the natural variation [34].

<table>
<thead>
<tr>
<th>CO\textsubscript{2} Location</th>
<th>Monitoring tool</th>
<th>Advantages (+) and disadvantages (-)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sub-Surface</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Passive Seismic Monitoring (PSM) |    | + Can be done continuously and in real-time [35]. Cost-effective after the initial installation.  
+ Allows fractures and their location to be detected at the time of appearance, allowing remediation actions to be taken early.  
- Not every microseismic event reported in the cap rock is synonymous with a fracture in the cap rock. Geomechanical models are necessary to interpret the detected events [16].  
- Baseline data has to be collected before the injection starts. |
| Seismic survey                |                 | + While a single survey can provide information about the location of the CO\textsubscript{2} plume in the storage formation, time lapse seismic monitoring can track the development and movement of the CO\textsubscript{2} plume.  
- Expensive and therefore the time intervals between consecutive measurements are large. |
| Wireline Logging              |                 | + Can be used to trace the vertical migration of CO\textsubscript{2}.  
- Baseline data has to be collected before the injection starts [27].  
- Only provides information within a short radius of the well. |
| Pressure monitoring           |                 | + Protection against over-pressure which could fracture the cap rock.  
+ Can be used to assess reservoir properties which in turn can be used to calibrate reservoir models.  
+ Can be done continuously and in real-time.  
- May not be sufficiently sensitive to detect small leaks. |
| Geochemical monitoring        |                 | + Impacts of CO\textsubscript{2} leaks as well as of displaced brine can be detected. Ensures safety of drinking water sources.  
- Localized point measurement that may not provide timely information.  
- Small leaks may not significantly change water chemistry. |
| **Near-Surface**              |                 |                                      |
| Soil gas composition, soil flux measurements, gas sampling in head space of water wells |             | + Measures indicators of a CO\textsubscript{2} leak.  
- Each measurement is only representative for a small area. |
| Satellite and airplane imagery |                 | + Coverage of a large area in a short period of time.  
- Subtle changes in the flora are late indicators for a CO\textsubscript{2} leak. Quantification not possible beyond a specific threshold. |
| **Atmosphere**                |                 |                                      |
| Atmospheric sensors           |                 | + NDIR sensors allow the CO\textsubscript{2} concentration to be measured continuously and in real-time.  
- Each measurement is only representative for a small area.  
- CO\textsubscript{2} leaks disperse rapidly in the atmosphere and quickly reach background levels.  
- Need to be near leak and measure with high precision measurements for a long time. Problem with timely detection of leaks. |
| Airborne sensing (low-level airplane surveys) |             | + Coverage of a large area in a short period of time.  
- Insensitive, expensive and therefore the time interval between the measurements are large. |
| Remote sensing (satellite)    |                 | + Coverage of a large area in a short period of time.  
- Can only detect large leaks and requires intensive data processing. |
footprint of three square kilometers [33], which when combined with the total CO$_2$ in the atmospheric column (approximately 2 km high), dwarfs the size of any suspected surface leak.

### 1.1.5 Summary

The advantage of CCS over other CO$_2$ reduction options is that it has the possibility of significantly reducing the amount of anthropogenic atmospheric CO$_2$ emissions while maintaining existing fossil fuel energy infrastructure. However, CCS can only be considered part of the solution to reduce global CO$_2$ emissions and must be supplemented with non-fossil fuel derived energy production and energy efficiency measures. CCS involves capturing CO$_2$ from carbon-rich energy production and industry processes, transporting it to the location of a suitable storage site, and injecting it into an underground geological formation where it ideally remains indefinitely.

To make geological storage of CO$_2$ effective and safe, ensuring that the CO$_2$ does not re-enter the atmosphere is essential. The probability of leakage to happen at a storage site is highest at the time of injection and decreases thereafter due to the positive effects of the residual, dissolution and mineral trapping [36]. Furthermore, the injection-induced pressure onto the reservoir rock and cap rock is not a factor in the long term storage.

Various monitoring methods are available to detect CO$_2$ leaks within the underground formation. A summary of the most important techniques together with their advantages and disadvantages can be found in Table 1.1. With risk mitigation strategies in place, there is according to [2, page 14] a 90-99% chance that more than 99% of the injected CO$_2$ will be kept in the storage formation for over 100 years.

The risk posed by CCS is comparable to other methods used by the oil and gas industry. For example, natural gas is stored in rock formations all over the world to balance out seasonal fluctuations in gas supply and demand. A prominent example for this is the natural gas storage facility located underneath the Olympic Stadium in Berlin [37]. Natural gas is in comparison to CO$_2$ highly flammable and explosive. Risks similar to the ones posed by CCS also apply to CO$_2$-EOR, which likewise involves CO$_2$ to be transported to and stored in underground rock formations.

It has to be noted that the previously described monitoring techniques will not only be used to ensure the safety of onshore geological storage of CO$_2$ during injection. The operator of the CCS facility must also provide evidence that no CO$_2$ has been released from the storage site for a certain period after the CO$_2$ injection has stopped. Only then, the legal obligations can be passed on from the site operator to a local or national authority. The monitoring techniques are therefore also necessary for the site operator to provide evidence about the successful and permanent storage of the CO$_2$.

### 1.2 Problem derivation

In the event of CO$_2$ escaping the storage formation, it is important that remediation measures are taken as soon as possible. Therefore, monitoring techniques detecting problems early on are crucial. The previously introduced sub-surface monitoring tools can detect possible fracture events or expose a CO$_2$ migration towards the surface while the CO$_2$ is still underground. They are therefore better suited for early problem detection than near-surface or atmospheric monitoring tools.
Chapter 1. Introduction and motivation

Sub-Surface monitoring techniques have an additional advantage over near-surface and atmospheric monitoring tools: Close to the surface or at the surface, the CO$_2$ concentration as well as other measured indicators (e.g. soil gas, soil flux, soil moisture content) are dependent on environmental fluctuations (e.g. seasons, day and night). Exhaust gases and additional impacts from nearby industry and farming also affect the measurements. It is therefore necessary to collect extensive baseline data before the injection starts in order to interpret collected measurements during the injection phase correctly.

The usage of tracers can help to make near-surface and atmospheric measurements easier to analyze. Tracers are chemical compounds that can be co-injected with the supercritical CO$_2$ into the storage formation and give the stored CO$_2$ a unique characteristic (fingerprint). The advantage of tracers is that they have low detection limits as well as a low and stable background concentration. Tracers measured in the soil gas or in the atmosphere above their background concentration reveal that CO$_2$ from the storage formation has migrated to the surface. In contrast, environmental fluctuations in CO$_2$ do not raise the concentration of the tracer. With the help of tracers, even very small quantities of escaping CO$_2$ can be detected [39, 40]. An additional benefit of tracers is that their concentration can be used to quantify the CO$_2$ leakage rate. CO$_2$ tracers which can be measured and analyzed in the atmosphere in real-time are under development [41, 42].

Each monitoring technique has its advantages and shortcomings. Therefore, a mix of sub-surface, near-surface, and atmospheric monitoring tools is necessary to detect all problem cases. This is also evident in the directive 2009/31/EC (Annex II, 1.1) [43] of the European Union, which explicitly includes both monitoring below and at the surface:

\begin{quote}
The choice of monitoring technology shall be based on best practice available at the time of design. The following options shall be considered and used as appropriate:

(j) technologies that can detect the presence, location and migration paths of CO$_2$ in the subsurface and at surface;

(k) [...]

(l) technologies that can provide a wide areal spread in order to capture information on any previously undetected potential leakage pathways across the areal dimensions of the complete storage complex and beyond, in the event of significant irregularities or migration of CO$_2$ out of the storage complex.
\end{quote}

(excerpt of directive 2009/31/EC (Annex II, 1.1) [43])

While the legal requirements regarding CCS vary between countries, the excerpt of the directive 2009/31/EC points out two fundamental points:

1. [...] shall be considered and used as appropriate [...]

Not all monitoring techniques are required to be implemented. This allows site specific constraints to be accommodated in the monitoring mix. It can be expected that

---

\textsuperscript{6}Natural tracers which do not have to be co-injected with the supercritical CO$_2$ (e.g. stable isotopes and noble gases) are currently under investigation [38] for measurements, monitoring and verification (MMV) purposes at onshore geological CCS storage sites.
the site operator will choose a set of monitoring tools which complies with the legal requirements and is cost effective.

2. [...] technologies that can provide a wide areal spread [...], [...] across the areal dimensions of the complete storage complex and beyond [...]

Suitable storage sites can ideally accommodate multiple injection wells and have a high storage capability. The storage formation can therefore spread over multiple square kilometers. Emissions have to be detected even beyond the extend of the storage complex. That requires a monitoring technique which can cover a vast area in a cost effective manner in order to keep the overall costs of CCS low.

Sub-surface and near-surface monitoring tools are well understood by the oil and gas industry due to their common use in hydrocarbon extraction projects. The remainder of this work will therefore focus on atmospheric monitoring tools for onshore geological CO$_2$ storage sites.

Atmospheric sensor networks are an option to monitor the CO$_2$ concentration at the surface over a vast areal extent. However, to stay within reasonable costs, each sensor node has to be inexpensive and easy to deploy [44, 45]. An additional problem which comes into play is the density of sensor nodes which is necessary to detect also smaller CO$_2$ leaks: Due to the rapid dilution and dispersion of CO$_2$ into the atmosphere, sensors need to be deployed in close proximity of the leak. CO$_2$ has already a significant background concentration in the atmosphere (i.e. ~390 ppm), which further exacerbates the problem. Statistical analysis of high precision CO$_2$ measurements and leak simulations show that for a CO$_2$ leak at a distance 1 km away from a high precision measurement station, a point source leak would need to be in the order 20 t/d before it could be detected [46]. A dense sensor network covering a large area would therefore be necessary to detect small containment breaches. Considering the current price of high-accuracy atmospheric CO$_2$ sensors in addition to the costs associated with their deployment and further infrastructure expenditure (e.g. power supplies, communication modules) results in a low benefit-cost ratio for a sensor network solution.

Remote sensing of the CO$_2$ concentration on the surface via satellites allows the coverage of large areas. However, satellites which cover the region of interest have to be available. In addition, the time when measurements can be taken as well as the time intervals between consecutive measurements are dictated by the design and elevation of the satellite and do not necessarily coincide with the monitoring schedule. Furthermore, the gained measurements are the sum of the CO$_2$ concentrations from the surface to the stratosphere, making the measurements also dependent on environmental factors which are decoupled from problems in the storage site.

Measurements taken from an airplane provide flexibility in time and frequency of the measurements. At the same time, large areas can be covered. Despite these advantages, the costs associated with this monitoring technique are high, especially if frequent measurements are needed. This can however be expected in the case of a containment breach.

In summary, the existing monitoring techniques do not allow a frequent and cost-effective solution for the atmospheric monitoring of CO$_2$ over large areas. The open problem can be summarized as follows:
Chapter 1. Introduction and motivation

Problem Statement:
A cost-effective solution for the surface detection and frequent monitoring of CO$_2$ (or CO$_2$ tracers) over and beyond the dimensions of an onshore geological CO$_2$ storage site in case of a containment breach needs to be developed.

1.3 Robot based solutions

In contrast to sensor networks, where the basic idea is to deploy multiple sensors around the area of interest, the core idea behind the use of robot-aided gas monitoring is to move a single sensor over the whole application area. Depending on the costs of the robot platform and the savings made by using only a single sensor, the overall costs can be drastically reduced. The coverage area of a mobile robot is dependent on the type of robot, the provided on-board battery power as well as the sensors and actuators of the platform.

Gas sensitive sensors have been used on mobile robots since the beginning of the 1990s to create gas distribution maps or to detect odor sources [47]. Most of the introduced approaches try to mimic the behavior of animals, notably moths [48, 49, 50], lobsters [51], crayfish [52], and dung beetles [48]. As discussed in [53, 54] most approaches have been designed for, and experiments have been conducted in controlled and simplified environments: Assumptions like a steady and constant airflow, an indoor environment, the presence of a single gas source, little or no obstacles, and a very limited size of the exploration area are usually made. In a real-world geological storage environment, these assumptions do not necessarily hold true: While there is a possibility that the gas enters the atmosphere at a single point, a leak along a natural line or crack in the ground as well as a diffuse CO$_2$ leak cannot be excluded. In addition, it can be assumed that the CO$_2$ will surface in a natural environment which can feature a complex terrain (hills, trees) as well as unpredictable wind behavior. Due to the constraints of the introduced problem statement, this work will further focus on the discussion of the available material regarding outdoor gas monitoring and source finding strategies with mobile robots.

1.3.1 Related work

In [55, 56] the authors present a ground-based system called GASBOT developed for landfill monitoring tasks. The platform is based on an ATRV-JR robot and can detect methane (CH$_4$) leaks indoors as well as outdoors. The system uses the SEWERIN REMOTE METHANE LEAK DETECTOR (RMLD), a tunable diode laser absorption spectroscopy (TD-LAS) sensor which reports gas concentrations as integral measurements over the path of the laser beam. The RMLD can measure CH$_4$ concentrations of as low as 5 ppm·m at distances up to 15 m. Larger distances can be covered with less accuracy [57]. Using the approach presented in [58] a gas distribution grid map can be calculated based on the TD-LAS measurements and their respective laser beam paths. To successfully construct a gas distribution map, one has to collect multiple measurements for each grid cell from various angles. Wind gusts negatively influence the results in outdoor setups due to the problem that the gas concentration of a cell cannot be assumed as constant between multiple measurements. Errors in the position estimation of the robot further degrade the
quality of the gas distribution map due to the faulty start and end position estimations of
the laser beams.

A gas-sensitive quadcopter\footnote{Quadcopters are also referred to as quadrocopters in the literature.} for adaptive gas source localization and gas distribution mapping is presented in [54, 59]. The platform is based on the AIRROBOT AR100-B and is equipped with a DRÄGER X-AM 5600, a gas probe designed for personal monitoring applications which is able to sense up to six gases simultaneously. The unit uses infrared and electrochemical sensors to produce its measurements (for details see [60]). A modification of the sensor has been necessary to stay within the payload limits of the quadcopter [61]. In addition to the gas sensor, a humidity and temperature sensor has been integrated into the platform. To gain a gas concentration with the least amount of dilution caused by the rotors of the quadcopter, three approaches to transport the gas to the sensor have been tested in a wind tunnel: A passive approach (gas measurements are taken during normal flight), a semi-active approach (the suction effect of the rotors is used to transport the gas through a small pipe to the sensor) and an active approach (a dedicated fan inside a tube which protrudes from the radius of the quadcopter is used to push the gas towards the sensor). While none of the approaches reached the reference gas concentration, the active approach performed best with 66%, followed by the semiactive approach with 52% of the reference concentration. For their real-world CO$_2$ distribution experiments around the geochemically active Tuscany region (Italy), the semi-active approach has been chosen by Neumann et al. due to its applicability and still reasonably good sensitivity. During these experiments, 20 s worth of measurements have been taken for each predefined sensing location. Based on these measurements and the approach described in [62] a distribution map has been created. While the first run showed promising results with the calculated distribution map depicting the source location at its actual location, the following three runs estimated the source location with an offset of around 10 m. The authors give the destruction of the pre-experimental gas distribution caused by the first experimental run with the quadcopter in combination with a too short waiting period between the test runs as reason for that phenomenon. Neumann et al. also published a paper suggesting their UAV setup to be used for the monitoring of onshore geological CO$_2$ storage sites [63]. In their conclusion, the authors of the paper discuss a range of open problems which in combination make it problematic to adapt their introduced approach from being used for the atmospheric monitoring of onshore geological CO$_2$ storage sites: With the slow response rate of the currently available gas sensors with low weight which can be used on the quadcopter, it is necessary for the UAV to hover for a period of time to take a measurement. This further limits the area which can be covered by the aerial platform during the already limited flight time ($\sim$ 20 min) of the quadcopter. The stop-and-go measurement approach also further exaggerates the negative influence of the rotors onto the gas concentration measurements by constantly pulling “fresh” air from above the UAV which dilutes the pre-experimental gas concentration. In addition, Neumann et al. mention the negative effects of the position uncertainty introduced by the GPS receiver in use onto the accuracy of the gas distribution maps as a problem.

Adaptive sampling algorithms have been introduced (e.g. [64]) to speed up the search for a gas source over a wide area. These methods usually assume a gas source to be present as well as a single emission point. As previously discussed, a single location where the CO$_2$ is released into the atmosphere cannot be assumed if a leakage in an on-
shore geological CO\(_2\) storage site occurs. In case of a containment breach, all points where CO\(_2\) is released into the atmosphere have to be found, making the coverage of the whole area of interest necessary. An additional disadvantage of adaptive sampling algorithms in conjunction with an airborne setup is that the gas concentration is influenced by repeated passes of the aerial vehicle over the area, disturbing the original gas distribution.

### 1.3.2 Summary

Robot-aided gas monitoring tools have been developed, although not specifically for CO\(_2\) or the monitoring at onshore geological CO\(_2\) storage sites. Due to the possibility of multiple point sources or a diffuse release of CO\(_2\) into the atmosphere from a containment breach in a geological storage site, adaptive sampling algorithms cannot be used, but the whole area of interest has to be covered. Both ground-based as well as aerial robotic systems are available.

Due to the long response time of current CO\(_2\) sensors, the speed of the platform is constrained by the sensor rather than by the robot’s capabilities. This gives land-based systems the advantage that they can potentially cover larger areas than airborne systems due to their higher on-board power / consumption ratio. On the other hand, difficult terrain (e.g. patches of forest, fenced off or walled off areas) can be better monitored with aerial platforms which can take measurements above obstacles (e.g. forest) or simply fly over them (e.g. fences, walls) to access an area. In addition, airborne platforms provide the possibility to track a gaseous plume by monitoring the CO\(_2\) concentration in various altitudes.

With CO\(_2\) sensors expected to improve their response time and their accuracy / weight ratio as well as battery technology improving (increased power density), it can be expected that aerial solutions can boost their coverage while being able to provide both the basic monitoring as well as the tracking of gaseous plumes in the close future. The remainder of this work will therefore focus on the development of an airborne UAV to solve the introduced monitoring task, while trying to avoid the problems discussed by Neumann et al. in [63].

### 1.4 CO\(_2\) sensor the determining factor

In their described experiments, the group around Neumann used either a DRÄGER X-AM 5600 or an e-nose system based on the FIGARO TGS4161 to measure CO\(_2\) concentrations. The sensors have a response time of \(T_{90} \leq 10\) s and \(T_{90} \leq 90\) s respectively. To somewhat compensate for the long response time, the UAV is programmed to hover 20 s above each predefined measurement point and multiple measurements are taken. During this time, the air-CO\(_2\) mixture gets diluted by the four rotors of the quadcopter, degrading the measurements. The hovering further prohibits, next to the limited on-board battery power of the aerial vehicle, the coverage of a larger area. With the described setup, a maximum of 60 measurement locations are possible. On the positive side, the solution is flexible, cheap, and allows the frequent monitoring of the atmospheric CO\(_2\) concentration.

The problems experienced by Neumann’s group show that the choice of the CO\(_2\) sensor has great implications on all aspects of a robot-aided aerial CO\(_2\) monitoring tool: The

---

\(8\)The time a sensor needs to reach 90% of its final reading.
1.4. CO₂ sensor the determining factor

dimension of the UAV is bounded by the weight and size of the sensor, and the response
time of the probe dictates the maximum speed the aerial vehicle is allowed to fly (if one
targets a set spatial resolution). Even the technology used by the sensor has great implications onto the flying platform: If one would for example use a TDLAS sensor on the UAV, it would be necessary that each position one wants to have measured is covered by multiple individual measurements to allow the calculation of the CO₂ concentration at that point. On the other hand, if one would use a NDIR system, it would be necessary to fly over the whole area of interest because the sensor only measures at its current location. That again means that even the flight path of the UAV is dependent on the choice of the sensor and therefore also the path planning component of the aerial vehicle depends on this crucial decision.

1.4.1 Sensor choice

The advantage of TDLAS-based sensors that the CO₂ concentration can be measured
over a distance instead of only at a determined position is diminished due to the problem
that each grid cell has to be covered by multiple measurements in order to create a CO₂
distribution map. To ensure the detection of leaks along natural fault lines and of multiple
point leaks, a gas distribution map is however necessary. An additional problem with
TDLAS-based sensors is that they need a surface on which the laser beam bounces back
to the device to make a measurement. With increasing altitude the reflective properties
of a grassy surface will not be enough anymore to gain reliable measurements without
reflective beacons on the ground. But the distribution of such beacons over a large area
is not feasible. Therefore, the decision was made that TDLAS sensors will not be used on
the airborne vehicle.

<table>
<thead>
<tr>
<th>sensor model</th>
<th>response time</th>
<th>range</th>
<th>accuracy</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figaro TGS4161</td>
<td>90 s</td>
<td>350 .. 10 000 ppm</td>
<td>approximately ± 20% at 1000 ppm CO₂</td>
<td></td>
</tr>
<tr>
<td>Dräger X-am 5600</td>
<td>20 s</td>
<td>0 .. 50 000 ppm</td>
<td>± 100 ppm</td>
<td>250 g</td>
</tr>
<tr>
<td>Vaisala GMP343</td>
<td>≤ 2 s</td>
<td>0 .. 1000 ppm</td>
<td>± (3 ppm + 1% of reading)</td>
<td>360 g</td>
</tr>
<tr>
<td>Picarro G2401</td>
<td>≤ 5 s</td>
<td>0 .. 1000 ppm</td>
<td>≤ ± 0.15 ppm</td>
<td>27.2 kg</td>
</tr>
</tbody>
</table>

Table 1.2 gives an overview of non-TDLAS CO₂ sensors used by the previously introduced projects in addition to other commercially available sensors. One can see a clear weight increase for probes with higher accuracy. A good tradeoff between sensor weight, accuracy and response time is the Vaisala GMP343 CO₂ probe, which was therefore chosen to be used throughout this work.

1.4.2 Platform type

If a NDIR CO₂ sensor is moved by a robot through space, each collected measurement represents the average CO₂ concentration over the spatial area covered by the sensor
during the response time interval. Possible aerial robot platforms to move the sensor include blimps, fixed-wing airplanes, multicopter systems\(^9\), and helicopter systems\(^{10}\).

Blimps can stay aloft for a long time, however they provide only a limited amount of payload and are highly susceptible to wind. Fixed-wing airplanes can also cover a large area with their limited on-board power. The efficiency advantages of fixed wing UAVs can however only be used while maintaining a certain minimum speed. With the still long response time of the chosen CO\(_2\) sensor of around 2 s, the area represented by a single measurement would be large, limiting the detectability of smaller amounts of CO\(_2\) dispersed into the atmosphere against the existing CO\(_2\) background concentration. Both multicopter and helicopter platforms provide the ability to hover, fly with low to high speeds, and are also capable to fly in moderate winds.

The decision which UAV platform to use was heavily influenced by the fact that a platform change might be necessary later on to accommodate new sensors, with most likely different weight and size, to measure lower CO\(_2\) concentrations or to detect tracers. Quadcopters with waypoint-based flight out of the box are commercially available with a payload limit of up to 1.2 kg\(^{11}\). This limit provides enough room to accommodate the previously mentioned Vaisala GMP343 CO\(_2\) probe together with its battery, computing unit, wiring as well as a communication module for the measurement transfer to the operator. The addition of another sensor or the exchange of the CO\(_2\) probe with a heavier solution is however likely to exceed the payload limit.

Helicopter systems designed to be flown remote controlled form an economical base for a helicopter UAV. The platforms are available in various sizes, allowing payloads from just a few grams up to several kilograms. The designs of the helicopter systems remain relatively consistent throughout the spectrum (except slighter variations in the swashplate and tail-rotor design), allowing one to use the same underlying control algorithms after a platform change. Being able to use the same autopilot solution with an adapted parameter set on a range of helicopters with different payload capabilities makes possible adjustments to the UAV size later on less troublesome than upgrades from a quadcopter to another multicopter system, where the underlying control algorithms are different. With payload flexibility and extendability of the solution having the highest priority, the decision was made to use a helicopter platform as basis for the aerial CO\(_2\) monitoring tool.

1.5 Contributions of this thesis

Commercial autopilots for helicopter UAVs are available. They are generally all-in-one solutions comprised of a printed circuit board (PCB) including all sensors necessary for the position estimation of the UAV as well as a microcontroller unit (MCU) running the control software (hardware drivers, basic control algorithms).

A review of existing commercial solutions at the beginning of the project revealed high prices for the systems in combination with highly restrictive application programming interfaces (APIs). The MCUs used by the autopilots vary widely and range from

---

\(^{9}\)For the remainder of this thesis, multicopter is used as an umbrella term for quadcopters, hexacopters, octacopters, ...  
\(^{10}\)The term helicopter is used throughout this thesis to describe a helicopter with a main-tail rotor design.  
\(^{11}\)At the time of writing, a new quadcopter (MicroDrone MD4-3000) has been advertised, allowing up to 3 kg of payload. The system is however not commercially available yet.
1.5. Contributions of this thesis

**Contribution I (→ Chapter 2)**

**General description**
Development of a low-cost and reliable autopilot for helicopter UAVs. The autopilot provides waypoint-based flight and does not rely on a specific sensor suite for the position estimation nor a pre-defined helicopter model.

**Contribution to CCS**
The designed autopilot in conjunction with the developed helicopter UAV allows to monitor the atmospheric CO$_2$ concentration of an area of approximately 750 m$^2$ (with one set of batteries) in one altitude with a spatial resolution of 2 m by 2 m over an onshore geological CO$_2$ storage site. Point source leaks of as low as 100 kg CO$_2$ per day can be detected and their source located.

**Contribution to robotics**
A reliable, and real-time capable helicopter autopilot system which does not rely on a specific sensor or a pre-defined helicopter model. The system is based on top of commercially available stabilization systems for model helicopters.

- The autopilot software defines basic commands for waypoint-based flight and can be easily extended with additional flight patterns as well as measurement-driven flight routes. The software is written in Ada, a high-level and real-time capable programming language, which allows a seamless operation with the real-time operating system VxWorks (the targeted operating system)$^a$. The autopilot requires the global position and attitude of the UAV, a helicopter stabilization system as well as a motor controller with governor mode.

- The autopilot software has been implemented for x86 systems, allowing it to be run on a variety of hardware solutions. The user can pick a hardware which accommodates the input / output (IO) capabilities necessary and is compact and lightweight. The underlying helicopter hardware is not specified, which provides the user the flexibility to pick a model in the size which is capable of carrying the application specific payload.

The developed CO$_2$ sensing rotorcraft UAV can be seen as one implementation of the designed autopilot system for flights in the visual line-of-sight of the pilot$^b$. The hardware implements critical safety features, e.g. that the backup pilot can always gain control over the helicopter via the flick of a switch on the remote control. Separate power circuits for the autopilot components and the parts necessary for the remote controlled flight of the helicopter allow the backup pilot to land the machine even though the autopilot runs out of power.

---

$^a$General purpose operating systems are also supported (e.g. Linux-based systems).

$^b$The UAV is currently operated for educational, non-commercial purposes and is flown under the laws applicable for model aircraft in Australia. These laws mandate to keep flights within the visual line-of-sight of the pilot (see CASR Part 101). The UAV and its autopilot are however designed for beyond line-of-sight operations.
Contribution II (→ Chapter 3)

General description
Development of a 3D, relative acoustic positioning system prototype which allows one to determine the location of the UAV with enough accuracy, to enable autonomous landings of the rotorcraft UAV on a dedicated landing pad with a minimum size of 1 m by 1 m.

Contribution to CCS
The presented acoustic localization approach avoids additional payload on-board of the flying platform, which would reduce the overall flight time and therefore the area which can be covered by the CO₂ sensing UAV. The approach needs a planar microphone array which can be integrated into the helicopter landing pad without posing a threat for the UAV to be caught in.

Contribution to robotics
- The developed acoustic, TDOA-based approach allows one to localize a sound source in the vicinity of a planar microphone sensor array. The possible range depends on the design of the microphone array as well as on the acoustic properties of the signal source.
- Existing acoustic localization approaches are either restricted to a set number of microphones, or cannot be used in conjunction with planar sensor arrays. The developed approach has been developed specifically for the use of planar sensor arrays and allows the use of N microphones, where $N \geq 4$.
- An introduction and direct comparison of various TDOA-based acoustic localization approaches of various models (hyperbolic, linear, spherical, and cone model) can be found in this work. Advantages and restrictions of the models as well as of the individual approaches are discussed.

The algorithm is not restricted to the task at hand, but can also be used in other domains, e.g. for speaker localization and tracking.

8-bit to 32-bit units. The processing power provided by the autopilot hardware is generally powerful enough to run the control algorithms, however not all solutions provide the computational flexibility or API access to allow the software to be extended with new, demanding components.

An additional shortcoming of existing autopilot solutions for helicopter systems is that the all-in-one approach does not allow one to update or exchange the sensors used for the position estimation. Therefore, time-consuming hardware redesigns may be needed if sensors need to be changed or updated, e.g. to make use of sensors with better accuracy or higher update rates. Sensor redundancy can therefore also not be implemented. This however is necessary for the development of a reliable solution which can be used for beyond-line-of-sight operations. Furthermore, an additional computing unit is necessary to interact with the sensor suite used for the atmospheric monitoring task.
1.5. Contributions of this thesis

**Contribution III** (→ Chapter 4)

**General description**
Design and implementation of a UI framework which can connect to multiple sensors and robots simultaneously, allowing the management of all components comprising a comprehensive, robot-aided monitoring system. The framework also provides tools for measurement data representation and analysis.

**Contribution to CCS**
An all-in-one monitoring and robot management solution. The system can be extended with plugins to support new sensors and robots as well as data analysis and representation tools. The licensing of the system allows manufacturers to provide closed-source or proprietary plugins to integrate their robots or sensors into the existing framework.

**Contribution to robotics**
- The UI framework provides a one-stop solution for the management of mobile robots (individual robots and groups of robots) and for the graphical representation of the gathered measurements (both internal state of the robot as well as data about the environment).
- The system provides a clear separation of UIs from the sensor / robot control software (proprietary protocols or robotics frameworks). This allows one to use the same UI implementations for a variety of robots.
- The software was designed to provide a high grade of extendability while focusing on code reuse of both the connections to sensors / robots as well as of UIs.
- The framework core uses the Eclipse Public License which allows third-party groups and companies to provide framework extensions using their own license model.

due to the fact that all-in-one solutions do usually not provide IO ports for additional sensors.

In order to gain an autopilot solution which provides the hardware flexibility desired for the aerial monitoring of CO$_2$ over onshore geological CO$_2$ storage sites, a new system was designed and developed (see box Contribution I).

During landing maneuvers, uneven ground, stones or other rubble hidden in the grass around the landing area of the UAV can shatter the tail rotor or cause problems if one of the skids is placed on them. A landing pad can be used in order to prevent such unnecessary risks. The use of a helipad has also other advantages: It reduces the amount of dispersed dust during take-off and landing, and the planar surface of a helipad provides staff with a suitable platform to prepare and service the UAV. But most importantly, it is very useful for the backup pilot to know exactly where the UAV is sup-
posed to land, allowing him to identify possible problems early on and react quickly in case of such unforeseen circumstances. With the landing position of the UAV known, it is also possible to place necessary tools and accessories close by.

To be able to land on a dedicated helipad, an accurate position of the helicopter with respect to the landing pad is necessary. In order for the backup pilot to land the helicopter on the landing platform in moderate winds, the helipad should have at least the outline of 1 m by 1 m. However, to be easily portable, the size should be kept as small as possible. Commonly available GPS receivers do not provide the necessary accuracy for landing the UAV on a helipad of the specified size, making an alternative solution necessary (see box Contribution II).

For flight-planning and real-time status monitoring of the helicopter UAV, a ground control software (GCS) is necessary. Existing solutions can be found, most of them are open-source. These software packages however solely provide GCS functionality and do not provide support for a comprehensive measurement data analysis of data from various sources. Due to the fact that the developed UAV is only one of multiple monitoring solutions, the GCS should ideally allow the management of not only the UAV but also of other monitoring systems (e.g. continuous soil flux monitoring systems, fixed atmospheric monitoring stations, other robotic monitoring solutions). Only a software framework providing the user full access to and control of all measurement tools of the system allows one to respond to new developments informed and quickly. Therefore, a solution allowing the simultaneous management of multiple sensors and robotics systems (of different vendors with various configurations) in addition to measurement data analysis tools for the comprehensive interpretation of measurement data gained by these sources is necessary (see box Contribution III).

While the combination of the contributions targets to solve the introduced problem of large-scale, atmospheric CO$_2$ monitoring of onshore geological CO$_2$ storage sites, the individual contributions are different in their nature. Therefore, every contribution will be presented in its own chapter, which provides next to the presentation of the developed method a section about the related work which has been done in the particular field of each contribution. Furthermore, a discussion about the pros and cons of the developed system with respect to its field and to the introduced monitoring task is provided. The thesis concludes with a summary next to a discussion of possible improvements to allow the presented solutions to be incorporated into a monitoring program at a commercial, onshore geological CO$_2$ storage site.
The rotorcraft UAV and its autopilot

Chapter summary and structure

This chapter describes the design and development of a general purpose autopilot software for helicopter UAVs for non-acrobatic flight maneuvers. The focus of the design is on hardware flexibility (sensors, helicopter platform, computing unit), reliability (real-time design, sensor redundancy) and simplicity (architecture, extensions and adaptations).

The introduced autopilot distinguishes itself from other solutions by not relying on a specific sensor suite for the position estimation, and by not specifying the underlying helicopter hardware. The free choice of hardware components allows one to easily exchange sensor models as well as sensor modalities. This enables the user to make full use of the broad range of available commercial off-the-shelf (COTS) position estimators as well as of new developments in this field.

The presented autopilot also distinguishes itself from existing solutions by treating the stabilization of the helicopter and its waypoint-based flight component as two completely decoupled entities. This assumption allows one to use COTS products for the stabilization of the aircraft, which shortens the overall development time and reduces the focus of the autopilot software to drift compensation and the management and execution of flight patterns. The helicopter stabilization system is dual-used for the remote controlled flight and the waypoint-based flight of the autopilot.

After the introduction of the autopilot, the rotorcraft UAV developed for the atmospheric monitoring of CO\textsubscript{2} concentrations over onshore geological storage sites is presented. The solution can be seen as one implemented adaptation of the previously discussed autopilot. The chapter closes with a feasibility study which tests the usefulness of the developed rotorcraft UAV for the monitoring of atmospheric CO\textsubscript{2} concentrations.
Chapter 2. The rotorcraft UAV and its autopilot

2.1 An autopilot for helicopter UAVs

2.1.1 Drawbacks of existing solutions

At the beginning of this project, a study of available commercial autopilot systems has been undertaken to evaluate if one of these systems could be used for the monitoring of atmospheric CO$_2$ concentrations. At that point in time, the available solutions for helicopter autopilot systems were scarce and very expensive. The systems combined all hardware necessary for the position estimation and control of the helicopter on a single PCB to gain a small-sized and lightweight solution. This approach however restricts the user from exchanging or adding individual sensors without hardware redevelopments. Updated sensors with better accuracy or higher update rate as well as new developments can therefore not be made use of. In addition, sensor redundancy cannot be implemented.

Another problem of existing systems were restrictive software APIs which allowed one only minimal access to the control algorithms. Extensions that need direct access to the control loop (e.g. obstacle avoidance) can therefore not be realized. Even though the provided APIs were rather restrictive, most vendors only offered them as a separate purchase.

For the monitoring of atmospheric CO$_2$, one should follow the given flight path closely in order to disturb the pre-experimental CO$_2$ distribution as less as possible. The reviewed systems were however not designed to do so, and focused on smooth transitions between waypoints (resulting in a natural looking flight path) rather than on accurate flight path execution.

While the UAV and its autopilot are prototype systems developed to show that a flying robot platform can be used for the cost-effective atmospheric CO$_2$ monitoring of large areas over onshore geological storage sites, the systems are designed with the objective of being deployed at a commercial CO$_2$ storage site. Sensor redundancy plays an important role for reliable beyond line-of-sight operations in which a backup pilot cannot simply switch to remote controlled flight in case of unexpected errors on-board of the UAV. In addition, obstacle avoidance techniques on-board of the flying platform are necessary to avoid trees, fences and walls in the area of interest. With beyond line-of-sight operations and flights in low altitudes expected to be necessary for the UAV-assisted atmospheric CO$_2$ monitoring at commercial geological storage sites, all-in-one autopilot solutions are not suitable for the implementation of the UAV and its autopilot.

During the course of this work, UAVs received increasing attention by both hobbyist and research groups. That lead to the development and availability of multiple open-source and open-hardware autopilot systems. These developments have been made possible by advances in battery technology and a drop in prices for sensors used for the position estimation of UAVs (GPS modules, accelerometers, gyroscopes). Despite this progress, some problems still remain: Most systems available today have been developed for multicopters or fixed wing airplanes, with helicopter UAVs only receiving little attention. In addition, the projects are still focusing on all-in-one solutions making it hard or even impossible to add or change sensors used for the position estimation, and therefore to implement sensor redundancy without costly and time-intensive hardware redesigns.
2.1. Design decisions

The UAV used for the CO₂ monitoring task should allow the sensor suite for the detection of CO₂ leaks (e.g., CO₂ sensors, tracer detectors) to be exchanged. Ideally, the computational unit of the autopilot can also be used to record the measurements from these sensors. In addition, it should be possible to change the underlying helicopter platform to a larger size in case new payload requirements demand so. It should not be necessary to rewrite the autopilot software after such a step.

If one generalizes these requirements, one can see that this hardware flexibility is beneficial whenever the sensor suite for a given task may change, or new requirements demand the UAV configuration to be extended or adapted. Therefore, a new general-purpose autopilot software targeting non-acrobatic flight patterns (e.g., no flips or inverted flight) for helicopter UAVs was designed to overcome the problems of existing systems while providing hardware flexibility, reliability, and simplicity. The following paragraphs discuss the design decisions which lead to the autopilot architecture introduced in Section 2.1.3.

**Hardware flexibility**

All-in-one autopilots usually implement the device drivers and control algorithms on a MCU which is placed together with the sensors for the position estimation and necessary IO ports for the control of the actuators on a single PCB (see e.g., Figure 2.1). This results in small and lightweight solutions, but prohibits individual sensors to be added or exchanged. Furthermore, to manage payload sensors and actuators a second computational unit is usually required.

![Image](image.png)

**Figure 2.1:** The ARDUPILOT MEGA 2.5 is an example for an all-in-one autopilot solution. The PCB includes a 3-axis gyro, accelerometer and magnetometer as well as a barometer. A connector is provided to accommodate an external GPS. The all-in-one solution is compact and lightweight however restricts the user to the sensor suite provided (Photo by Jordi Muñoz, CC BY).

In contrast to this common design practice, separating the hardware necessary for the position estimation of the helicopter from the remaining hardware needed for the execution of the control algorithms, allows all sensors, including the basic sensors necessary...
for the position estimation, to be exchanged. This step makes a dedicated computational
unit for the execution of the autopilot code necessary. The hardware has to provide the
IO ports for all sensors and actuators of the system. A separate board for the execution
of the autopilot code has the disadvantage of adding more weight onto the UAV, and
therefore potentially reducing the overall flight time. In addition, the autopilot hardware
will be less compact, which may prohibit its use on small UAVs. For larger UAVs, the
payload will significantly outweigh the components of the autopilot system, and thus
the negative effects will be less significant.

The proposed separation of the position estimation from the remaining autopilot sys-
tem results in following advantages:

• Free choice from a variety of COTS position estimation units which come in differ-
ent sizes, weight, accuracy, and price.

• Possibility to exchange the position estimation unit with a better performing model,
if a new or improved sensor is available on the market.

• Exchange of the computational unit to satisfy changing hardware requirements (IO
capabilities, computational power).

• Sensor redundancy for enhanced reliability.

• Possibility to use multiple sensors with a different sensing modality to deliver the
same quantity (e.g. position). This is especially interesting in combination with re-
dundancy where the backup sensor is ideally using a different sensing modality
than the primary sensor to ensure that the backup sensor can deliver the needed
quantity and does not suffer from the same shortcomings than the primary sensor
(e.g. a vision-based approach could be used to deliver the position in case of lost
GPS reception).

Reliability
The autopilot system of an UAV supporting beyond line-of-sight operation is a safety-
critical system. A failure of the system can not only lead to financial damage for the
operator (due to the loss of the UAV platform itself as well as due to the unfinished
mission), but can also result in people on the ground being harmed or even killed. It is
therefore necessary to make the system as reliable as possible. The proposed autopilot
targets to increase reliability by implementing a set of measures:

• Operating system (OS)
The real-time operating system (RTOS) VxWORKS (WIND RIVER) is targeted to
run the control software of the autopilot. The RTOS features deterministic timing
behavior (e.g. bounded interrupt latency and time for OS calls) and provides a fixed
priority scheduling system.

In order to prevent compromising the gained hardware flexibility, x86-based sys-
tems are targeted as basis for the computational unit. A wide range of such systems
with board support packages (BSPs) for VxWORKS are available in various sizes
and with different IO capabilities, allowing the user to select the hardware which
best suits the task at hand.
2.1. An autopilot for helicopter UAVs

- **Programming Language**
  The autopilot software system is designed to be implemented in a programming language providing inter-task communication, synchronization and real-time operations.

- **Sensor Redundancy**
  Sensor redundancy plays an important role for beyond line-of-sight operation, in which remote controlled flight is not a valid backup alternative. Sensor redundancy comes at the cost of increased overall weight and power requirements while improving the reliability of the overall autopilot system by making it (to a certain degree) fault-tolerant to the failure of individual sensors.

- **Functional Isolation**
  The autopilot software is designed to run completely on-board the helicopter. No communication to the GCS is necessary for the solution to work. Even a complete communication break-down between the autopilot software and the GCS does not influence the autonomous flight. Solely the operator loses the ability to monitor the UAV (position, battery voltages, ...) as well as the possibility to change its previously given flight path (sending new waypoints). The decision to run the autopilot exclusively on-board the UAV makes the system more reliable, being independent of unstable wireless communication links and time lags which would be introduced through off-line computing.

**Simplicity**

The control of a helicopter is complex due to the fact that it resembles a multivariable non-linear underactuated control system [65]. A large amount of research has been done to investigate suitable control methods for helicopters. The developed solutions range from learning-based methods [66] and fuzzy logic approaches [67] to model-based control methods [68].

In order to hide the complexity of the helicopter control from the user, the design decision was made to split the stabilization of the helicopter and the waypoint-based flight into two decoupled entities. This assumption allows one to use COTS helicopter stabilization systems which are commonly used by pilots of radio controlled (RC) helicopters to simplify the control of helicopter models which are not mechanically stabilized e.g. by a flybar stabilizer. Electronic stabilizer systems generally provide a heading hold feature and bring the helicopter model into the horizontal plane if the stick positions are neutral. They do however not compensate for drift. Depending on the stabilizer system in use, acrobatic flight-maneuvers might not be possible anymore. For most applications, including the atmospheric CO$_2$ monitoring task, this is however not a restriction.

With a stabilizer on-board, the autopilot software itself has to solely implement drift compensation (position hold) and waypoint-based flight, resulting in an easy to understand autopilot architecture. To build the autopilot system on top of a stand-alone stabilization unit has also the advantage that the stabilization system can be dual-used for both remote controlled and waypoint-based flight. Remote controlled flight is also necessary for beyond line-of-sight operations of the UAV to test new or changed hardware setups, and to tune the parameters of the autopilot.
2.1.3 Architecture

The autopilot architecture can be logically split into a hardware independent and a hardware dependent part. The former section of the software system can be used across various UAV configurations, allowing the developer to make use of existing flight pattern implementations without the need to copy source code fragments or to adapt source code.

The CO₂ sensing helicopter UAV developed in this thesis employs an implementation of the autopilot system described in this subsection. The UAV platform itself as well as the hardware and task dependent autopilot structures implemented for the aerial measurement platform are discussed in Section 2.2.

The chosen programming language for the reference implementation is Ada 2005. Ada 2005 is a high-level and real-time capable programming language which also seamlessly integrates with VxWORKS.

2.1.3.1 Hardware independent section

Concurrent tasks read the sensor measurements and communicate with the GCS as well as the stabilization system. Sensor measurements, parameters, and waypoints are stored in protected objects, which allow multiple simultaneous read operations and manage mutually exclusive write operations.

![Diagram of autopilot architecture]

Figure 2.2: Structure of the autopilot architecture. The diagram is based on the unified modeling language (UML) class diagram notation but has been adapted to allow Ada specific structures (Tasks, Protected Objects, Tagged Types) to be modeled (type of structure is specified in brackets).

The autopilot task - the heart of the system - executes and manages flight patterns. A flight pattern can implement e.g. that the UAV follows the shortest flight path between...
waypoints (Euclidean distance), focuses on a smooth flight path (fly along the path created by fitting a polynomial curve through the waypoints), or circles around a given waypoint. Flight pattern transitions can be triggered by the autopilot software internally or by the user via the GCS.

Each FlightPattern depends on the current global position of the UAV (latitude, longitude, elevation above ground level, yaw with respect to true north) and the waypoints provided by the user. Four PID controller (one each for transitioning forward/backward, left/right, up/down and one for rotating around the yaw axis) form the basis of each FlightPattern. The PID parameters can be set and changed by the GCS operator during flight. Extensions which need direct access to the control loop, e.g. obstacle avoidance techniques, are also implemented inside a FlightPattern. To keep FlightPattern implementations independent of the underlying hardware (sensors, stabilization system), each implementation makes use of an InputDataManager to gain the necessary information and an OutputDataManager to forward the calculated commands to the hardware (see Figure 2.2).

2.1.3.2 Hardware dependent section

During flight, the velocities of all rotor blades (main and tail) are kept constant. The autopilot changes the direction of flight as well as the speed of the UAV by changing the pitch of the main and tail rotor blades. The collective pitch is changed to produce more or less thrust while changes in the cyclic pitch provide angular acceleration. The collective as well as the cyclic pitch are controlled by servos which change the level (collective pitch) and the orientation (cyclic pitch) of the helicopter swashplate located on the main rotor shaft. Changes in the angle of the tail rotor lead to a change in yaw of the UAV. The governor mode of the motor controller senses changes in the motor speed (and therefore indirectly changes in the load) and counterbalances these changes in order to keep the speed of the rotor blades constant.

The OutputDataManager receives commands from the FlightPattern in percent for each direction (forward/backward, left/right, up/down, yaw). A 50% value represents the neutral position (no acceleration). The OutputDataManager converts the percentages to values which can be forwarded to the stabilization system (hardware dependent). The stabilization system sets the final positions of the servo motors on the main and tail rotor based on its internal state as well as the inputs received by the autopilot. All swashplate setups supported by the stabilizer system (e.g. 3 point linkage 120°/135°/140°, or 4 point linkage 90°) can be supported by the autopilot.

Sensor redundancy can be implemented in the InputDataManager. If e.g. two inputs for the position are available, the InputDataManager verifies the validity of both measurements (are values too old or marked as invalid by the sensor) and provides the FlightPattern with the position best suited at the current time, which can be the only valid measurement or the more accurate one (Section 2.2.2.2 discusses how sensor redundancy is implemented on the CO2 sensing UAV). In case of any unexpected error or unavailable position information, a fallback FlightPattern is executed. This special FlightPattern can e.g. implement neutral stick positions to stabilize the helicopter (hovering, yet drifting) in addition to an acoustic notification for the backup pilot in case of line-of-sight operations.
2.1.3.3 Timing constraints

The autopilot ensures that timing constraints are met by employing preemptive fixed priority scheduling. Two priority bands are implemented: The higher priority band contains all hard real-time tasks which can only be blocked while accessing a protected object (e.g. autopilot task). In contrast, the lower priority band contains all tasks which gain measurements from the sensors, communicate with the stabilizer system and generally tasks which can be blocked due to IO operations. The fact that no protected object provides any entries and their implemented functions and procedures are all non-blocking in addition to the utilization of priority ceiling inside all protected objects in combination with preemptive fixed priority scheduling guarantees a deadlock free system. A schedulability analysis for the tasks of the higher priority band can be done by determining the worst case execution times of all tasks of this band. This ensures the schedulability of the tasks in a given environment (RTOS, hardware).

2.2 A helicopter UAV for aerial CO\textsubscript{2} monitoring

2.2.1 Hardware

Helicopters are available in a large variety of sizes, ranging from small remote controlled models up to full-scale manned systems. This variety is also reflected in helicopter UAVs. Full-scale systems like the UNMANNED LITTLE BIRD (BOEING), the K-MAX UNMANNED MULTI-MISSION HELICOPTER (LOCKHEED MARTIN), and the UNMANNED FIRE SCOUT (NORTHROP GRUMMAN) have been developed for reconnaissance and supply missions in military setups. In contrast, small to medium scale helicopter UAVs target mostly non-military tasks including, but not limited to mapping tasks [69], agricultural tasks (e.g. YAMAHA R-MAX for crop dusting), inspection tasks [70], fire detection and monitoring [71, Chapter 8], environmental monitoring [72], as well as law enforcement tasks (e.g. TACTICAL ELECTRONICS RAPTR for tactical reconnaissance). The aerial platforms are either remote controlled or feature some level of autonomy, which can range from waypoint-based flight to the autonomous detection and selection of a landing site (e.g. [73]).

Figure 2.3: The CO\textsubscript{2} sensing rotorcraft UAV based on the ALIGN T-REX 700E RC helicopter model.
2.2. A helicopter UAV for aerial CO\textsubscript{2} monitoring

Figure 2.3 shows the developed UAV during a remote controlled test flight. The model helicopter ALIGN T-REX 700E has been chosen as basis for the aerial vehicle due to its payload capacity, the availability of a sensor gimbal for the model, as well as being battery powered. The flight duration of electric powered helicopter models is significantly less than the one provided by their petrol-powered counterparts, reducing the overall area which can be covered by the aerial vehicle. A battery-powered solution is however necessary for the monitoring of geological storage sites to ensure that the CO\textsubscript{2} measurements are not affected by any exhaust gases of the UAV platform itself.

A single motor (ALIGN RCM-BL700M 510KV) is used to power both the main and tail rotor, which are coupled via a fixed-ratio gear set. The governor mode of the motor controller (CASTLE CREATIONS PHOENIX EDGE HV 120) ensures that the speed of the main and tail rotor is kept constant under varying loads. The helicopter features a two-bladed main rotor with direct drive (also known as flybarless rotor head). The swashplate position of the ALIGN T-REX 700E is controlled by three servo motors in a 120° setup (see Figure 2.4). The electronic stabilization system LF-TECHNIK GYROBOT 900 is used on the helicopter. Without any stick movement on the remote control, the electronic stabilization system keeps the UAV in a horizontal position, but does not compensate for drift. The LF-TECHNIK GYROBOT 900 is shared between remote controlled and waypoint-based autonomous flight.

To switch between the two modes of operation (remote controlled, autopilot), an EMCOTEC DPSI TWIN MAXI V2.0 is used. The system allows to change between the two modes via a switch on the remote control. It is therefore always possible for the backup pilot to gain control over the helicopter, even if the autopilot fails due to hardware failure or unexpected exceptions. In addition to this function, the EMCOTEC DPSI TWIN MAXI V2.0 also provides power to the servos (up to 16 channels) from two lithium polymer (LiPo) batteries connected to its inputs. The system acoustically warns the operator about almost empty batteries on its inputs and incorporates a logger, collecting information about power usage and errors (e.g. number of received invalid pulse width modulation (PWM) signals).

For the position estimation, a XSENS MTI-G attitude and heading reference system is employed, which fuses the measurements of a GPS receiver, a three-axis accelerometer, a three-axis gyroscope, a three-axis magnetometer, and a pressure sensor to provide the user with the global position of the UAV as well as with the attitude of the UAV with 100 Hz update rate. To gain the relative elevation of the helicopter above ground, a HOKUYO UTM-30LX laser range finder is used.

A KONTRON PITX-SP single board computer (SBC) with an INTEL ATOM Z530 processor with 1.6 GHz provides computational flexibility in a compact form factor and is used as computational unit for the autopilot (see Figure 2.7). The RTOS WINDRIVER VXWORKS was targeted to run on the platform. Unfortunately, a BSP for the KONTRON PITX-SP is only available for version 6.8 of the OS. Version 6.9 is however necessary to support sensors and actuators with built-in universal serial bus (USB) to transistor–transistor logic (TTL) serial universal asynchronous receiver / transmitter (UART) converters, which are used by some of the on-board sensors. A BSP for WINDRIVER VXWORKS V6.9 for the KONTRON PITX-SP is at the time of writing still not available. In the meantime, a minimal version of DEBIAN SQUEEZE is used as OS. Not using a RTOS can lead to cases in which the processing power is too less to fully execute all tasks before their
Figure 2.4: A closer look onto the rotor head delivers insight into the mechanic used to change the collective and cyclic pitch of the main rotor blades: Three servos control the orientation of the swashplate. Lever arms between the main rotor blade mountings and the swashplate then change the pitch of the blades. Swashplate movements parallel to the ground (up and down) change the collective pitch, while the orientation of the swashplate changes the cyclic pitch. The motor controller (CASTLE CREATIONS PHOENIX EDGE HV 120) can be seen on the left, the electronic stabilization system (LF-TECHNIK GYROBOT 900) on the right.

Figure 2.5: Fully equipped sensor gimbal with VAISALA GMP343 CO₂ probe, power distribution board, payload battery, HOKUYO UTM-30LX laser range finder, and DIGI INTERNATIONAL MCQ-XBEE-XSC communication module (cannot be seen in the picture, mounted underneath the laser range finder in the front of the gimbal).
2.2. A helicopter UAV for aerial CO\textsubscript{2} monitoring

**Figure 2.6:** The GPS receiver EAGLE TREE GPS EXPANDER V3 which is used as alternative sensor for the position estimation can be seen on top of the sensor gimbal. To its right, the main receiver (SPEKTRUM AR9000) is mounted on top of the EMCOTEC DPSI TWIN MAXI V2.0 which switches between the outputs of the receiver (remote controlled flight) and the autopilot. The antenna of the communication module DIGI INTERNATIONAL MCQ-XBEEXSC can be seen on the left side protruding from the sensor gimbal.

**Figure 2.7:** Computational unit KONTRON PITX-SP SBC (right) next to the position estimation unit XSENS MTi-G located on the tail boom of the UAV. The main batteries including the battery tray extension can be seen on the left side.
Chapter 2. The rotorcraft UAV and its autopilot

deadline. Low priority tasks will then miss their deadlines. The priority of the tasks should therefore be chosen next to their timing constraints according to their importance for the usability of the overall system, e.g. tasks managing payload as well as logging tasks should have low priority.

To output the commands calculated by the autopilot to the stabilization system, PWM signals have to be generated. The COTS development module ALVIDI AL-XSLED ATXMEGA featuring an ATMEL ATXMEGA256A3U MCU is used in conjunction with a custom made PCB to allow the simultaneous generation of 16 PWM outputs. The developed PCB provides the level converters from 3.3 V (MCU) to 5 V (LF-TECHNIK GYROBOT 900), a connector for each servo channel (outputs), plugs for the monitoring of the battery voltages (inputs), as well as a FTDS RS232RL USB to TTL serial UART converter for the communication between the KONTRON PITX-SP and the MCU. The PWM signals are created with 50 Hz while the battery voltages monitored by the built-in MCU analog-to-digital converters (ADCs) are sent with 10 Hz to the autopilot. A custom-made PCB was necessary to support all 16 PWM channels provided by the EMCOTEC DPSI TWIN MAXI V2.0.

The CO₂ sensor VAISALA GMP343 is mounted together with the payload battery and a power regulation PCB on the gimbal. In case the CO₂ sensor needs to be exchanged and the new gas probe has different power requirements, only changes on the sensor gimbal are necessary. The power distribution board is the second custom-made PCB. This was necessary because a surface mounted device (SMD) is used for the power regulation (POWER TRENDS 78SR112VC). The PCB features next to the power regulator only connectors for the sensors (power, USB). Next to the CO₂ sensor, the power distribution PCB, and the payload battery, the HOKUYO UTM-30LX laser range finder as well as the DIGI INTERNATIONAL MCQ-XBEXSC wireless link, used for the communication with the GCS, are mounted on the sensor gimbal (see Figures 2.5 and 2.6).

To compensate for the weight on the gimbal, a battery tray extension (see Figure 2.7) has been created with a 3D printer. Shifting the main batteries further to the back allows to counterbalance for the payload on the sensor gimbal. In case the helicopter platform cannot cope with the weight of a new or additional sensor, it is possible to change the helicopter platform itself. A bigger version of the ALIGN T-REX 700E (ALIGN T-REX 800E) is available. In addition, if the overall weight of all on-board electronics can be reduced, a smaller version of the model helicopter (ALIGN T-REX 600E) can be used to create a more streamlined solution.

Problems in the autopilot system are propagated acoustically to the backup pilot via the on-board speaker mounted between the landing skids. Protection against the loss of power during flight due to drained batteries is provided by the continuous monitoring of all battery voltages. The measured voltage levels are forwarded to the GCS operator who can react to low voltage levels accordingly. All autopilot parts and payload components are powered by a different battery circuit than the components necessary for remote controlled flight. Therefore, if the autopilot/payload battery runs out of power, all systems necessary for the backup pilot to land the helicopter are still available. A schema of the power distribution on-board the helicopter is depicted in Figure 2.8. The separation of the power circuits is also beneficial for the components of the autopilot system because the input power to the sensors is not negatively influenced by fluctuating voltages caused by the main motor or the servo motors.
All tasks of the autopilot software run concurrently with different update rates. The position updates from the XSENS MTi-G are with 100 Hz the most frequently received measurements used by the autopilot. The autopilot task itself therefore runs with 200 Hz. The control commands to the stabilizer are send to the Atmel ATXMEGA256A3U MCU with 100 Hz which then creates the PWM signals (with a period of 20 ms) for the stabilization system. The servos changing the angles of the main and tail rotor blades are set by the stabilization system. The information flow on-board of the helicopter is depicted in Figure 2.9.

The commonly found component setup of helicopter UAVs is to centralize the sensors as well as the computational unit in an enclosure which is mounted between the landing skids. The setup has the advantage that the box can be detached, and the components can be tested without the helicopter. In addition, due to the small distance between the components, the length and therefore the weight of the cables is reduced. This weight reduction is however counterbalanced by the weight of the enclosure. The disadvantages of the setup are that a higher landing gear is usually necessary to accommodate the enclosure and that the center of gravity is shifted towards the ground and away from the center of rotation leading to a less linear roll and pitch characteristic (commonly described as a pendulum effect). To avoid this problem, the author decided to use a different approach: in order to keep the center of gravity as close as possible to the center of rotation, the components of the system have been distributed over the whole helicopter platform, including the tail boom. The setup has the benefits that the flight dynamics of the helicopter do not change significantly compared to the original model, and that the pendulum effect is avoided. These advantages benefit the pilot during remote controlled operation, and make the UAV platform more stable. The setup does however not allow an operator to run all components without the helicopter or to test the helicopter without the autopilot hardware.

### 2.2.2 Autopilot adjustments

#### 2.2.2.1 Sensor gimbal task

During flight, the sensor gimbal is pointed towards the ground. This behavior is controlled by an additional task in the autopilot software and ensures that the laser range finder measures the relative distance of the UAV to the ground. The antenna of the communication module, which is also mounted on the sensor gimbal, benefits from this by being kept in the same direction than its counterpart connected to the GCS. This significantly improves the connectivity during the experiments.

The CO\textsubscript{2} probe has been mounted on the bottom of the sensor gimbal, so that when the gimbal is pointed towards the ground, the main rotor downwash actively pushes air through the sensor. This ensures a constant flow of air through the Vaisala GMP343, allowing one to use the CO\textsubscript{2} sensor with its 2 s response time setting. With the minimum response time of the CO\textsubscript{2} sensor, the helicopter does not have to hover during a monitoring run to take a measurement. Instead, the UAV can fly with a low and constant velocity, allowing a larger area to be covered with less disturbances to the pre-experimental CO\textsubscript{2} distribution than it would be possible with a stop-and-go approach.

Each measurement of the Vaisala GMP343 can be understood as the average CO\textsubscript{2} concentration of the air volume above the main rotor. The gas in this volume gets sucked
Figure 2.8: Power sources and power distribution on-board of the helicopter. There are three decoupled power circuits: the motor circuit (blue), the UAV electronics circuit (red), and the payload and autopilot circuit (green). Each circuit is powered by one or more LiPo batteries (bold boxes). The voltage is controlled or regulated (dashed boxes) before it is distributed to the components (all other boxes) in the directions indicated by the arrows. Remote controlled operation of the helicopter is possible with only the UAV electronics circuit (red) and the motor circuit (blue).
Figure 2.9: Information flow on-board of the UAV. There are two possibilities to control the helicopter (blue boxes): remote controlled by the backup pilot or via the autopilot implemented on the on-board computer. The behavior of the autopilot can be influenced by the GCS (purple box). The backup pilot can choose between remote controlled and autopilot operation by flicking a switch on the remote. The switching between the two modes is done in hardware by the DPSI Twin Maxi (bold red box). Dashed links indicate wireless communication, while solid edges depict wired communication links. The information is flowing in the direction shown by the arrows. All receivers on-board of the helicopter (dotted box) receive the same information from the remote control. The SPEKTRUM AR9000 with its two satellite receivers is used for redundancy, while a separate satellite receiver is used to allow the autopilot to observe the stick positions of the remote control. Green boxes depict hardware components solely used for the waypoint-based flight of the UAV.
into and mixed by the main rotor blades before it is pushed through and measured by the CO$_2$ sensor. The airflow created by the main rotor can therefore be seen as beneficial for the monitoring, allowing the average CO$_2$ concentration of a volume to be measured, which is more representative than a CO$_2$ measurement at a specific position.

2.2.2.2 Sensor redundancy

To increase the reliability of the helicopter UAV, key quantities are measured by multiple sensors. The global position delivered by the XSENS MTi-G can be marked as invalid due to low GPS signal reception, an insufficient amount of satellites found, or due to problems of the Kalman filter used inside the unit. Therefore, a second GPS receiver (EAGLE TREE GPS EXPANDER V3) has been mounted on top of the sensor gimbal and can be used as a backup. An alternative sensor for the laser range finder measuring the relative distance above ground is also available: An ultrasonic sensor (MAXBOTIX MB1320) which is mounted underneath the helicopter chassis. The implemented sensor redundancy has following limitations:

- Both the XSENS MTi-G as well as its backup sensor are based on GPS technology. Low GPS signal reception can lead to the problem that both the primary as well as the secondary sensor cannot receive a stable GPS fix at the same time. Ideally, one would like to use a secondary sensor which uses another sensing modality than the primary sensor. A non-satellite based global positioning alternative to GPS is however not commercially available at the moment.
- The EAGLE TREE GPS EXPANDER V3 is more prone to GPS position jumps than the XSENS MTi-G due to the fact that it only uses the raw GPS position and does not fuse multiple sensors for its position estimation. Furthermore, the backup GPS receiver has a much lower update rate (5 Hz raw GPS position) than the primary sensor (100 Hz inertia-interpolated GPS position).
- The MAXBOTIX MB1320 reduces the possible flight elevation drastically (maximum range of the ultrasonic sensor is 7.65 m compared to 30 m guaranteed range of the HOKUYO UTM-30LX). On the positive side, the backup sensor uses a different sensor modality than the primary sensor. Therefore, if e.g. direct sunlight is causing the laser range finder to experience problems, the backup sensor can deliver the distance above ground because it is immune to lighting conditions.
- The EAGLE TREE GPS EXPANDER V3 does not replace the XSENS MTi-G completely. In case of communication problems with the XSENS MTi-G, no attitude information will be available.

The implemented setup allows only flights up to an elevation of 7.65 m due to the restrictions of the ultrasonic sensor. This disadvantage is of minor concern for the detection of CO$_2$ leaks over onshore geological storage sites due to the fact that the helicopter is targeted to fly close to the ground (see Section 2.3) in order to detect even small CO$_2$ leakages.

It has to be noted that it is not intended to use the secondary sensors over an extended period of time e.g. to fly a complete mission. The switch from the primary sensors to their backup solutions is signaled to the operator of the GCS as a critical problem. The GCS operator should abort the operation and give the UAV the command to fly to
2.2. A helicopter UAV for aerial CO\textsubscript{2} monitoring

Possible problem: Current position is invalid or out of date.
Reasons: No GPS reception; invalid position: position is not available, attitude still valid; Problem with hardware, unexpected error in task communicating with Xsens MTi-G; data too old: position and attitude invalid.

Figure 2.10: Sensor redundancy implementation in the InputDataManager: UML Activity Diagram for the function getPosition().

the landing position if the primary sensors do not recover after a short period of time. Figure 2.10 shows how the discussed sensor redundancy setup is implemented in the InputDataManager of the developed helicopter UAV.

2.2.2.3 Measurement FlightPattern

The stabilization system used on the developed UAV supports multiple modes of operation (beginner, beginner acrobatic, and standard). The OutputDataManager implementation for the previously introduced UAV setup is only implemented for one of the modes (beginner mode, targeting non-acrobatic flight maneuvers). With the switching between the standard and beginner modes during flight being discouraged by the manufacturer, it has to be ensured at the start of the autopilot that the expected mode is set. A SPEK-
TRUM SPM9545 satellite receiver is used to gain the stick positions from the remote control. A dedicated task in the autopilot software communicates with the device and retrieves the state of the flight mode switch as well as of the switch which is used to choose between manual and autonomous flight. The check if the right flight mode is set is implemented in the procedure `initEarly()` of the `OutputDataManager` which is called during the startup phase of the autopilot. Errors in the procedure will result in an error message and the immediate stop of the autopilot system, ensuring that a flight is only possible if all necessary preconditions are met.

Changes in the payload require the rotor speed of the helicopter to be adjusted. This governor mode setting of the motor controller can be altered via the remote control. Because the payload can change between flights, the current rotor speed (as set by the backup pilot on the remote control) is read the first time the user switches from remote controlled operation to autopilot mode (after remote controlled take-off). This is implemented in the `initLate()` procedure of the `OutputDataManager` which is (next to the `initLate()` procedure of the `InputDataManager`) called by the `initLate()` procedure of the `FlightPattern`, which in turn is called when the `FlightPattern` is executed for the first time. The splitting of the initialization code into two separate procedures enables an immediate stop of the autopilot if settings not allowed to be changed during runtime are not set correctly during startup (`initEarly()`), and allows the initialization of necessary internal data structures with measurement data from sensors which need some time to provide correct values (e.g. GPS) later on (`initLate()`).

In order to disturb the pre-experimental CO\textsubscript{2} distribution as less as possible, the UAV is planned to fly along measurement lanes (blue path) upwind over the measurement area (gray area). In order to avoid destroying the pre-flight CO\textsubscript{2} distribution, the helicopter is planned to fly in lanes upwind over the measurement area (see Figure 2.11). These lanes should be followed as closely as possible. Therefore, following strategy is implemented (see Figure 2.12):

1. The helicopter turns its nose towards the next waypoint (set yaw towards goal). During this phase, the current latitude, longitude and altitude is maintained.
2. The UAV changes the altitude to be level with the next waypoint (go to goal altitude). During this phase, the current latitude, longitude and yaw angle is maintained.
3. The aerial platform flies towards the next waypoint (fly to goal). During this phase, the elevation and heading is maintained, while the UAV is controlled to stay on a
2.2. A helicopter UAV for aerial CO\textsubscript{2} monitoring

Step 1:
Set yaw towards goal

Step 2:
Go to goal altitude

Step 3:
Fly to goal

Step 4:
Position Hold (last waypoint) or go to Step 1 (next waypoint available)

Waypoint n or initial position

Waypoint n+1

Intermediate waypoint

Figure 2.12: FlightPattern used for the atmospheric CO\textsubscript{2} monitoring.

direct line which can be drawn between the last waypoint (or the initial position) and the next waypoint.

4. If the helicopter reaches the waypoint, the next waypoint is targeted. If no new waypoint is available, the position of the last waypoint is kept (position hold).

Figure 2.13 and 2.14 show how the FlightPattern is implemented in the developed helicopter UAV.

2.2.2.4 Fallback FlightPattern

In case of an error in the measurement FlightPattern, the fallback FlightPattern is activated. This sets all stick positions to neutral which signals the stabilization system to keep the aerial platform level. An acoustic warning signal is used to indicate the backup pilot about the switch to the fallback FlightPattern. Hearing the warning signal, the backup pilot can decide to let the stabilization system do its job or directly take control of the UAV by flicking a switch on the remote control. The implemented fallback FlightPattern is possible due to a current limitation\textsuperscript{1} restricting the developed UAV to be used only for in line-of-sight flights of the backup pilot. The implemented fallback FlightPattern is not dependent on any position or altitude information, making it a robust solution.

2.2.3 Ground crew

For the operation of the developed CO\textsubscript{2} sensing helicopter UAV, a ground crew of at least two persons is necessary: A backup pilot, who keeps the helicopter in its field of view and can take over the control of the UAV in case of unexpected problems. The second person is the GCS operator, who is responsible for transmitting the flight path to the helicopter, observing the measurement process on the GCS, monitoring the vital helicopter parameters (battery voltages, task states), and providing information about any detected problems to the backup pilot.

While two persons are suffice for the operation of the UAV, a third person keeping an eye on the periphery and on the area which should be monitored is beneficial. This

\textsuperscript{1}The UAV is currently operated for educational, non-commercial purposes and is flown under the laws applicable for model aircraft in Australia. These laws mandate to keep flights within the visual line-of-sight of the pilot (see CASR Part 101).
Chapter 2. The rotorcraft UAV and its autopilot

Figure 2.13: UML Activity Diagram depicting the implementation of the measurement FlightPattern (page one of two).
2.2. A helicopter UAV for aerial CO$_2$ monitoring

Figure 2.14: UML Activity Diagram depicting the implementation of the measurement FlightPattern (page two of two).

person can warn the operator and the backup pilot about approaching persons or people trying to get into the measurement area. A similar group structure has been discussed by Pratt et al. in [74] in which he discusses the use of a remote controlled helicopter for building inspections after Hurricane Katrina.
2.3 Feasibility study

A surface release experiment was conducted to confirm that the influence of the main rotor onto the atmospheric CO\textsubscript{2} measurements does not prohibit the use of the developed UAV for the monitoring of geological CO\textsubscript{2} storage sites as long as the aerial platform does not have to hover to take a measurement.

The CO\textsubscript{2} concentration which can be measured on the surface in case of a containment breach depends amongst other things on the CO\textsubscript{2} leakage rate from the reservoir as well as over which area the CO\textsubscript{2} is released into the atmosphere (e.g. a point source release or a diffuse release over a large area). To gain a better understanding about leakage rates, naturally occurring CO\textsubscript{2} seeps have been studied. The work presented in [75] analyzes the data of 286 of such seeps in Italy and shows that the studied CO\textsubscript{2} sources most commonly degas between 10-100 t of CO\textsubscript{2} per day and claimed the lives of 19 people in the last 50 years. Dry seeps in diffuse and vent configurations have been found to pose the highest danger to humans. The natural CO\textsubscript{2} seeps have been categorized in low (<1 t/d), medium (1-10 t/d), high (10-100 t/d) and very high (>100 t/d) flux seeps. As outlined in [75], with a suggested retention of at least 99% of the stored CO\textsubscript{2} during a 1000 year period, a leakage rate of 10-100 t/d would be considered acceptable for a storage facility injecting 3.6 Mt per year.

The aim of the developed CO\textsubscript{2} sensing UAV is to detect and localize even small CO\textsubscript{2} leaks. Therefore, a ground level release chamber with a release rate of an equivalent of 100 kg of CO\textsubscript{2} per day was used as emission source during the experiment. This CO\textsubscript{2} source would be categorized as a very low flux seep in the context of natural occurring CO\textsubscript{2} seeps.

![Figure 2.15: The photo shows the developed helicopter UAV while flying over the CO\textsubscript{2} emission source (Photo courtesy of Ben Coughlan).](image)

The experiment was conducted on a sunny and calm day with only sporadic and minor wind gusts. The helicopter was flown remote-controlled repeatedly in an average elevation of about two meters\textsuperscript{2} over the emission source. The VAISALA GMP343 CO\textsubscript{2} sensor was used with its air filter attached and set to a 2 s measurement time. The manufacturer

\textsuperscript{2}The elevation information provided is with respect to the center of the main rotor blades.
recommends not to use the air filter in conjunction with the fastest response rate. However, with the CO$_2$ sensor being fully subject to the downwash of the helicopter main blades, a steady airflow through the measurement chamber can be guaranteed even with the filter still attached. The advantage of keeping the air filter on is that the dust particles dispersed by the helicopter (see Figure 2.15) do not get into the measurement chamber and pollute its surface, which could negatively influence the measurements. The CO$_2$ source was activated a few minutes prior to the experiment to allow a CO$_2$ plume to be formed.

Figure 2.16: Recorded atmospheric CO$_2$ concentration during the experiment.

Figure 2.16 shows the measured CO$_2$ concentration throughout the flight. The experiment can be logically split into four parts: (I) measuring the ambient CO$_2$ concentration with the helicopter standing on the landing pad, (II) spin-up phase, (III) flying phase, and (IV) post-flying phase.

During the first phase, a relatively constant ambient CO$_2$ concentration with an average of 376 ppm has been measured by the VAISALA GMP343. The following spin-up phase did only have a minor effect onto the measured ambient CO$_2$ concentration which dropped to an average of 375 ppm. A major jump in the CO$_2$ concentration can be seen in the graph when the helicopter was flown over the emission source for the first time. Each consecutive flyover resulted in the corresponding CO$_2$ spike in the graph being less significant than its preceding one. This can be traced back to the ongoing dilution of the air-CO$_2$-mixture through the main rotor blades of the helicopter. Variations in wind speed and the altitude of the helicopter during the remote controlled flight also affected the CO$_2$ readings. After landing the UAV, one can see the CO$_2$ concentration recovering to the levels recorded in the second phase (the average CO$_2$ concentration in phase IV is 375 ppm).

The experiment shows that it is possible to detect the atmospheric effects of a CO$_2$ point source leak in the order of 100 kg of CO$_2$ per day with the developed UAV using a NDIR CO$_2$ sensor during flight. Furthermore, the recorded CO$_2$ measurements show the detrimental effects of the main rotor onto the pre-experimental CO$_2$ distribution if the UAV is flown repeatedly over the same area. Even if a single point source leak can be assumed, adaptive measurement techniques which can result in the UAV flying multiple times over the same area should therefore be avoided. Furthermore, the results indicate
that if an operator is interested in verifying a detected jump in CO₂ concentration, a second flyover has to be timely separated in order to allow the atmospheric CO₂ concentration to accumulate again (this approach assumes a constant leakage rate).

After the experiment was conducted, the CO₂ perturbations were modeled (without taking the effects of the UAV into consideration). The prevailing flow rate of the CO₂ emission source in addition to the meteorological conditions which were measured during the experiment via a weather station close by were used to create a model of the CO₂ plume based on the Lagrangian stochastic air dispersion model. The modeling of the CO₂ perturbations was done by Berko and Feitz, who published their results in [76]. A graphical summary can be seen in Figures 2.17 and 2.18.

Figure 2.17: The figure shows a slice through the modeled CO₂ plume at an elevation of 1 m (Image by Berko and Feitz [76], reproduced with permission).

Figure 2.17 depicts a horizontal cut through the modeled CO₂ plume at 1 m elevation. The plot shows that the CO₂ plume is skewed into the predominant wind direction. Additionally, one can see that the CO₂ quickly disperses into the surrounding air, which results in only a minor increase of the CO₂ concentration further downwind. Due to possible background variations from natural effects and industrial sources, small variations in the CO₂ concentration cannot be clearly assigned to a CO₂ leak. It is therefore necessary to fly the UAV along measurement lanes in small distance to each other. For the simulated CO₂ point source leak, measurement lanes approximately 5 m apart in an elevation of 1 m would provide enough CO₂ perturbations (~50 ppm) to clearly identify a CO₂ leak as reason for the increased CO₂ concentration at the measurement site.

Figure 2.18 shows vertical cross sections of the modeled CO₂ plume. It can be seen that the CO₂ plume stays close to the ground and that virtually no increase in CO₂ concentration can be expected in elevations greater than 4 m. That means that the UAV has to stay close to the ground (~1.5 m elevation) to detect small CO₂ leaks.
Figure 2.18: The figure shows four vertical slices through the created CO₂ perturbation model coincident and parallel to the plume centreline (Image by Berko and Feitz [76], reproduced with permission).
If one compares Figure 2.16 with Figure 2.18 part b, one can see that the CO$_2$ perturbation measured by the UAV during its first flyover of the emission source in an elevation of approximately 1.6 m is consistent with the perturbation predicted in the plume model. That shows that the main rotor does not significantly influence the measured CO$_2$ concentration when measurements are taken during flight.

One can summarize that the developed CO$_2$ sensing helicopter UAV can be used to detect CO$_2$ point source leaks which emit CO$_2$ with a flow rate of $\geq$ 100 kg per day. Furthermore, the prevailing CO$_2$ concentration can be accurately estimated by the NDIR CO$_2$ sensor on-board the aerial platform. It is however necessary to measure the CO$_2$ concentration close to the ground in order to detect low flux CO$_2$ point sources. Furthermore, the dispersion of the CO$_2$ into the air around the emission source demands that the distance between the lanes flown by the helicopter UAV has to be kept small in order to ensure that no leaks are being missed. The optimal elevation above ground and distance between neighboring measurement lanes is dependent on the CO$_2$ emission rate as well as on the meteorological conditions during the monitoring flight.

2.4 Summary

The development of a helicopter UAV and its autopilot system was discussed in this chapter. The introduced general purpose autopilot software system allows the user free choice of sensors, helicopter hardware and stabilizer system. In addition, no software protocol for the communication between the GCS and the autopilot software is enforced. The software system can be easily extended with new flight patterns giving the user full control of the behavior of the UAV. Extensions like obstacle avoidance which have to be implemented directly into the control loop can be realized. Sensor redundancy for reliable beyond line-of-sight operations can also be implemented.

The internal structures of the autopilot software can be reused in various helicopter setups. An InputDataManager has to be implemented for each helicopter setup, while an OutputDataManager has to be implemented for each stabilization system. Drivers necessary for communicating with sensors can be reused if multiple helicopter setups use identical sensor models. Furthermore, flight patterns can be reused across UAV configurations without source code changes.

Decoupling the helicopter stabilization and the waypoint-based flight into two separate entities allows COTS stabilizer systems to be used. This step can reduce the maneuverability of the helicopter to non-acrobatic maneuvers depending on the capabilities of the stabilization system. The solution however benefits from robustness against changes in payload, which can be problematic with model-based solutions.

While the autopilot software does not restrict the user to any specific sensor, it assumes a position estimation unit, a stabilizer system as well as a motor controller with governor mode to be present. The helicopter hardware can be freely chosen according to the size and payload restrictions of the task to be solved. The autopilot can support the swashplate setups allowed by the stabilizer system.

The autopilot software grants the user a great deal of flexibility while being reliable, simple to understand and to extend. The key advantage of the autopilot software is that the UAV system can be easily upgraded over time with new sensors and flight patterns to solve tasks which the helicopter was initially not designed for. The advantages come...
with the negative side effect of added weight (multiple individual components instead of a single all-in-one PCB) which can result in a shorter flight time. While the expected weight differences are small, it can restrict this solution to be utilized on small systems.

The developed aerial measurement platform is a prototype system which was developed to test if a helicopter UAV can be used for the atmospheric CO$_2$ monitoring of onshore geological storage sites. The conducted experiment shows that the introduced hardware setup allows one to detect CO$_2$ point source leaks in the order of 100 kg of CO$_2$ per day. Additionally, it was found that the main rotor does not prohibit reasonable measurements of the prevailing CO$_2$ concentration during flight. An area of approximately 750 m$^2$ can be covered in one elevation with a measurement resolution of 2 m by 2 m with the described UAV configuration$^3$. CO$_2$ measurements in low altitude (~1.5 m) along measurement lanes with small distance to each other are however necessary to detect small CO$_2$ point source leaks.

$^3$Approximation based on collected flight data.
Chapter summary and structure

This chapter reviews existing strategies for autonomous helicopter landings on dedicated platforms, and the suitability of these approaches for the UAV-based CO$_2$ monitoring task. Passive acoustic localization methods can provide a solid alternative to the introduced strategies, integrating well with the ecosystem in use while avoiding shortcomings of existing approaches.

A detailed review of state-of-the-art passive TDOA-based acoustic source localization strategies is presented. The subsequent discussion about the approaches elucidates the relationships between them, their advantages as well as their shortcomings.

Thereafter, two new TDOA-based passive acoustic localization algorithms are introduced. The methods can calculate the position of a sound source in $\mathbb{R}^3$ in both the near-field and the far-field of planar microphone arrays by exploiting the geometric properties of the sensor arrays. The techniques work with sensor arrays of at least four microphones. The introduced solutions have a predictable computation time, making them suitable for real-time implementations.

Subsequently, optimal sensor placement strategies found in the literature are reviewed. The idea of uniform angular arrays is extended to maximum base distance uniform angular arrays and further to uniformly Cramér-Rao Lower Bound distributed maximum base distance uniform angular arrays. The latter sensor array geometry is the optimal solution with respect to the UAV localization task to be solved.

The developed algorithms are compared against competitors in their field in both simulations and real-world scenarios. The experiments underline that the newly introduced approaches provide accurate estimates for the position of the developed UAV in its landing area, enabling autonomous landings if the sensor array is embedded into the helicopter landing pad.
3.1 Introduction

3.1.1 Motivation

Automated helicopter landings require a precise position estimation of the airborne vehicle relative to the landing area. The use of a dedicated helipad for the take-off and landing of the helicopter UAV offers a number of advantages for a practical application like the CO₂ monitoring task in which the start and end-point of a mission remains the same: First, the planar surface of the helipad is beneficial for take-off and landing maneuvers, making it unnecessary to deal with uneven ground or rubble hidden in the grass around the deployment area. Concealed obstacles can easily shatter the tail rotor or cause problems if one of the skids is placed on them during a landing maneuver. Furthermore, some stabilization systems require the UAV to be started from level ground in order to correctly initialize their internal level point\(^1\). In addition, a landing pad reduces the amount of dispersed dust during take-off and landing. The planar surface of a helipad also provides staff with a suitable platform to prepare and service the UAV. But most importantly, it is useful for the (mandatory) backup pilot to know exactly where the UAV is supposed to land, allowing him to identify possible problems early on and react quickly in case of such unforeseen circumstances. With the landing position of the UAV known, it is also possible to place necessary tools and accessories close by.

While GPS is usually sufficient for general waypoint-based flight of UAVs, it is not accurate enough for the delicate task of landing the developed rotary-wing UAV on a small helipad of the size of 1 m by 1 m. A landing pad of this size can be transported in small cars while still giving the backup pilot the necessary area to land the UAV on the helipad in moderate winds. The landing skids of the helicopter used for experimentation in this thesis are 0.35 m long and 0.25 m apart. To ensure that the laser range finder mounted on the sensor gimbal delivers the distance from the helicopter to the helipad and not to the ground (in case the landing pad is elevated) and that the landing skids remain fully deployed on the platform after the landing approach, the absolute positioning error in the \(x, y\) plane (parallel to the ground) is not allowed to exceed ±0.25 m if the UAV is positioned above the center of the helipad. The error of both GPS systems available on the helicopter are given by 2.5 m circular error probable (CEP) and therefore do not provide the accuracy necessary for landing the UAV on the helipad.

Table 3.1: Non-GPS based sensors on-board the helicopter UAV providing the elevation of the aerial platform above ground.

<table>
<thead>
<tr>
<th>sensor</th>
<th>resolution</th>
<th>accuracy (ideal conditions)</th>
<th>minimum range</th>
<th>maximum range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maxbotix MB1320 (ultrasonic sensor)</td>
<td>0.01 m</td>
<td>-</td>
<td>0.2 m</td>
<td>7.65 m</td>
</tr>
<tr>
<td>Hokuyo UTM-30LX (laser range finder)</td>
<td>0.001 m</td>
<td>±0.03 m</td>
<td>0.1 m</td>
<td>10 m</td>
</tr>
<tr>
<td></td>
<td>0.001 m</td>
<td>±0.05 m</td>
<td>10 m</td>
<td>30 m</td>
</tr>
</tbody>
</table>

The altitude estimate of a GPS receiver is its most error prone datum and in practice less accurate than the horizontal position estimates [77]. The relative distance of the

\(^1\text{The stabilization system used on the developed UAV (LF-TECHNIK GYROBOT 900) needs to be initialized on level ground.}\)
3.1. Introduction

UAV above ground is however a vital piece of information, especially during landing approaches. Pressure sensors are commonly used to estimate the altitude of UAVs, however their low accuracy (approximately ± 1.0 m, e.g. BOSCH BMP180 DIGITAL PRESSURE SENSOR [78]) makes them unsuitable for automated landings. The developed UAV has two sensors which provide accurate estimates of the distance between the helicopter and the ground (see Table 3.1). The ultrasonic sensor which is deployed between the landing skids of the helicopter (~0.03 m off the ground) will stop providing measurements close to touchdown (this problem cannot be avoided as it is impractical to mount a sensor more than 0.2 m above the landing skids). The laser range finder mounted on the sensor gimbal can provide valid measurements during the whole landing process. Keeping in mind to ideally have each vital quantity measured by at least two sensors, the localization approach used to calculate the position of the helicopter relative to the landing pad should also deliver an accurate elevation measurement.

3.1.2 Related work

Safe and precise landings can be achieved by using GPS-based waypoint navigation combined with additional information from other sensors. A combination of GPS and vision has been used for automated landings of small scale helicopters on dedicated landing pads [79, 80]. A downside of these approaches is that the dimensions of the necessary marker on top of the helipad as well as the intrinsic camera parameters have to be known. In addition, the marker has to be clearly visible in the field of view of the camera and the lighting has to be adequate. These conditions might not be available all the time due to natural occlusions (leaves or debris on the landing pad) or unfavorable lighting during dawn, dusk or foggy weather. In case of the developed CO2 monitoring UAV, a camera is also not a part of the standard sensor set mounted on the helicopter. Due to strict payload restrictions, on-board sensors which are not necessary for mission completion but for landing purposes only should be avoided in favor of lower take-off weight and thus increased flight time.

A possible solution to keep the UAV free of additional payload is to embed all required sensors into the landing pad. Equipping the landing platform with the necessary sensors changes the task to be solved: Instead of the UAV finding the helipad to land, the landing pad now has to estimate the relative position of the UAV and has to provide this estimate to the aerial platform. The necessary hardware for the position estimation is now located on the ground, allowing for a wider range of sensors and computational units to be deployed. Nevertheless, two important points have to be kept in mind: First, dust particles are dispersed into the air during take-off and landing maneuvers of rotorcraft UAVs which can remain on the sensors embedded into the landing pad and negatively influence their readings. In addition, objects protruding from the landing pad should be avoided because they pose a risk for the UAV to get caught in.

Puls and Hein developed a system for autonomous landings of quadcopters which could be integrated into a landing platform [81]. Their three-tiered approach utilizes GPS for general flight, a radio-based position estimation system for the intermediate range, and finally an ultrasonic-based measurement system close to the ground. Unfortunately, their system depends on additional payload which has to be deployed on the UAV: multiple ultrasonic transmitters and a special radio transceiver which allows one to measure the distance between two devices of this type.
Chapter 3. An acoustic localization approach for precise landings on a helipad

For a vision-based approach, a wide-angle lens would be necessary to get a 180° field of view from the camera embedded into the helipad. The sensor would very likely suffer from the previously discussed dust problem in addition to almost unavoidable lighting problems with the camera facing straight up, and likely to include the sun in its visual field. Laser range finder based solutions offer an alternative, but 3D lidars are expensive while using a 2D scanner with an additional motor to enable 3D scans would lead to a low position estimate update rate. In addition, direct sunlight can also negatively influence the laser range finder measurements [82, 83].

Sound-based localization methods pose an interesting alternative with microphones not suffering much from dust particles, and due to their independence of lighting conditions. The sensors are also affordable and compact. Acoustic localization approaches are widely used in underwater environments where, depending on the application area, alternative sensor solutions are rare and sometimes not available at all [84, 85]. Atmospheric acoustic measurements are more prone (compared to underwater acoustics) to a number of disturbances: Wind, temperature, humidity and changes in atmospheric pressure alter the measurements. Additional noise sources which are existent in the application area, as well as reverberations have to be considered as well.

3.2 Setting the stage

3.2.1 Introduction to acoustic localization approaches

Acoustic localization schemes are commonly deployed indoors in class room or meeting environments for speaker position estimation and tracking. In virtual conference rooms this position information can be used to automatically keep the talker centered in a video frame (e.g. [86]). The speaker position can also be utilized to electronically steer the microphone array, which can increase the overall performance of the speech acquisition.

Sound-based localization methods have also been developed for outdoor applications involving aerial vehicles. In [87] a single-sensor method is introduced which can calculate the altitude, speed and engine revolutions of propeller aircraft and helicopters by utilizing passive acoustic sensor systems. The single-sensor method first calculates the instantaneous frequency of the received signal by exploiting the Doppler effect, and then utilizes a least mean squares approach to estimate the flight parameters. If multiple microphones are deployed, additional information like the angular trajectory of a transiting aircraft can be estimated [88].

To avoid any additional payload on the helicopter platform, a microphone array can be embedded into the landing pad which can also accommodate the computational resources necessary to calculate estimates of the helicopter’s position. The results of the calculations can be sent via the GCS to the helicopter. Ideally, this intelligent helipad can utilize the intrinsic sound of the helicopter for the localization of the UAV. Alternatively, the on-board speaker can be dual-used as sound source during take-off and landing next to its main task of providing acoustic feedback to the pilot and ground crew. The latter solution allows the transmitted signal to be chosen while the helicopter noise is dependent on the currently executed flight maneuver.

The transmission time as well as the travel time of the sound from its source to the microphone array is unknown for both of the signal sources. However, a broad spectrum of passive acoustic localization methods has been developed which do generally not
3.2. Setting the stage

rely on either information. These approaches can be divided into three categories: steered beamformer based methods, high-resolution spectral estimation based methods, and TDOA-based methods. The first two categories are considerably more computationally expensive than TDOA-based methods, while providing only minor benefits regarding localization accuracy [89]. For this reason, the remainder of this work focuses on TDOA-based position estimators.

Generally, TDOA-based approaches can be broken down into two steps: First, the TDOA $\tau_{ij}$ of the signal emitted by a source between a pair of spatially separated sensors $(i, j)$ has to be estimated (time delay estimation). This information can then be used in a second step to calculate the origin of the acoustic source (position estimation). Usually the range differences (RDs) based on the product of the available TDOAs $\tau_{ij}$ with the signal speed $c$ are used for the position estimation. The problem of finding the source position of a signal based on RD estimates is denoted multilateration. The problem is common in many fields (e.g. civil and military surveillance, service robotics, wireless sensor networks) and the applications are widespread. The requirements for available algorithms vary widely based on the signals being used (audio or electromagnetic), the environment in which the approach is being deployed, the dimensions of the problem (2D or 3D), the sensor measurement error margins and potentially available additional information (signal runtime, distance from the sensors to the signal source, initial guess of the signal source position).

3.2.2 Data acquisition process

The acoustic signal produced by the source is converted by means of microphones into an analog signal. This continuous signal has to be amplified before it can be converted via an ADC into a digital stream (see Figure 3.1). It is important that the signals of all sensors get digitized synchronously in order to avoid introducing time delays between channels.

![Figure 3.1: Data acquisition process - from the microphones to RD estimates.](image-url)

Based on the working environment and possibly available information about interfering noise signals, the digital stream is filtered to gain a cleaner representation of the signal of the acoustic source. This process is usually done in software with e.g. a series of band-pass filters. The filtered signal can then be used by the time delay estimator. The time
delay estimation process is computationally expensive\(^2\). Depending on the number of TDOAs to be estimated, the computation can be done by dedicated digital signal processors (DSPs), field-programmable gate arrays (FPGAs) or on general purpose computers. The resulting time delay estimates (TDEs) are multiplied by the signal speed \(c\) to obtain RD estimates which form the input for the position estimator.

### 3.2.3 Error sources

After the acoustic signal is emitted by the source, it travels through air to the microphones. The signal speed of sound through air is amongst other things dependent on the current temperature and humidity. These dependencies can vary during a single flight. To simplify or indeed enable any calculations, an ideal medium is generally assumed. The introduced error is non-systematic and depends both on the assumed values of the aforementioned parameters as well as on the weather conditions during the flight.

The signal emitted by the source is commonly not the only acoustic signal present in the environment. Other acoustic signals with similar or identical frequency bands as used by the source can interfere with the desired signal. Such noise is hard to extract and potentially leads to faulty TDEs. Even if no noise signal is present in the environment, reverberations through multipath propagation can be added to the acoustic signal and potentially lead to signal identification problems during the time delay estimation process. However, recently developed time delay estimators can handle certain forms and strengths of reverberations (e.g. [91]).

The microphones themselves are also sources of errors as their transmission curves are always non-linear (due to their mechanical resonance frequencies and other factors). Additional noise-floors are introduced depending on the type of microphone employed. The analog signal produced by the sensors reflects both the acoustic signals present in the system as well as the noise introduced by the sensor. The inexact positioning of the sensors is also reason for concern, leading to systematic errors in the position estimation process. This error is time invariant and therefore often assumed to be easily discoverable and correctable. However, a perfectly positioned sound source in conjunction with an ideal

\(^2\)For an overview of time delay estimation techniques and their complexity, see [90].

![Figure 3.2: Overview of the most important error sources.](image)
medium and the guarantee that no additional systematic errors are present in the system would be necessary to find and correct this problem.

The analog signal produced by the microphones needs to be amplified and digitized. Usually a synchronized bank of ADCs and amplifiers is employed. One can then assume that the delay which is introduced by this step is equal for all signals and therefore without influence onto the time delay estimation process (only relative time delays are used during the calculations). If different amplifiers are used to amplify the signals of all microphones of the sensor array, this might not hold true anymore. In addition, the amplification process adds noise to the analog signal. The imperfect amplification can also lead to different, frequency dependent amplification factors.

The ADCs digitizing the analog signal from the amplifier have to be synchronized in order to avoid adding a time delay to one of the input signals. Imperfect synchronization leads to systematic errors in the TDEs of sensor pairs digitized by out-of-sync ADCs. Each ADC features an unavoidable conversion error which is dependent on the targeted resolution in time (sampling rate) and value (quantization, bit rate) as well as on ADC-architecture characteristics. The higher the sampling rate and bit rate, the lower the resulting systematic error.

The subsequent filtering process can positively influence the time delay estimation process. This works especially well, if information about the properties of the disturbances (reverberations and noise signals) is available. However, false assumptions of these properties can lead to additional problems.

All of the aforementioned errors negatively influence the time delay estimation process. Various estimators are available (e.g. cross-correlation, generalized cross correlation (GCC) [92], adaptive eigenvalue decomposition [91]), and each of them works especially well in a certain environment or with a specific noise signal. Care has to be taken to choose the appropriate time delay estimator for the application environment. Time delay estimators usually assume that the acoustic source is an infinitesimal small and omnidirectional point source. However, real-world sound sources do generally not fulfill these properties which introduces an error to the TDEs or in exceptional cases even breaks the time delay estimation process entirely (no meaningful TDEs can be calculated). Another assumption which is not always true is that the signal source is modeled to be static throughout the time delay estimation process. The magnitude of the resulting error is dependent on the actual speed of the acoustic source as well as on the sampling rate used by the ADCs.

The position estimators base their calculations on the noisy TDEs or their corresponding RDs. Based on the model of the position estimator, additional errors are introduced due to model approximations or linearization steps inside the approaches.

All errors introduced so far are frequently combined to a single error which is assumed to be zero-mean and Gaussian. However, this assumption does not hold true with multiple errors being of systematic nature. The following presented approaches will therefore be analyzed with an error assumed to have both a Gaussian as well as a systematic error component.

### 3.2.4 Classification and quantity of TDOA estimates

For $N$ microphones, there are only $N(N - 1)/2$ distinct TDOA estimates $\tau_{ij}$ if one neglects the cases $i = j$ and uses each TDOA pair $\tau_{ij} = -\tau_{ji}$ only once [93, p. 1054]. The
Chapter 3. An acoustic localization approach for precise landings on a helipad

TDOAs span a \((N - 1)\)-dimensional space, meaning that \(N - 1\) linearly independent (LI) TDOAs can determine all the others in the absence of errors. In the presence of errors, the additional linearly dependent (LD) TDOAs can be used to reduce the overall error of the position estimate.

To simplify further discussions, the three sets \(I_0, I_{LI}\) and \(I\) are introduced. The set \(I_0\) denotes the set of sensor pairs producing all \(N(N - 1)/2\) distinct TDOA estimates, while \(I_{LI}\) is a subset of \(I_0\) which includes only the sensor pairs which provide the \(N - 1\) LI TDOA measurements. The set \(I\) includes all sensor pairs which are used for the position estimation process. A detailed discussion about the pros and cons of using \(I = I_0\) or \(I = I_{LI}\) can be found in Section 3.2.7.

### 3.2.5 The Cramér-Rao lower bound and the geometric dilution of precision

The Cramér-Rao lower bound (CRLB) is used in statistics to evaluate the performance of unbiased estimators. Specifically, the CRLB specifies the lower bound on the variance of the estimate. The CRLB is extensively used in conjunction with TDOA-based position estimators, even though most of the suggested estimators are biased. That can lead to situations where the variance of the position estimates is lower than the CRLB [94]. However, the CRLB is still an informative measure which can be used to determine how well a specific algorithm performs in relation to the theoretical limit.

For a microphone array with \(N\) sensors which is located in a 3D Cartesian coordinate system with its origin set in an arbitrary spatial point, the position of the microphones can be described by the vector

\[
p_{n} = \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix}, \quad n = 1, 2, \ldots, N. \quad (3.1)
\]

Let \(p_s = \begin{bmatrix} x \\ y \\ z \end{bmatrix}\) denote the position of the acoustic source, then the distances from the origin to the source and to the \(n\)-th microphone are

\[
R_s = \|p_s\|_2 = \sqrt{x^2 + y^2 + z^2}, \quad (3.2)
\]
\[
R_n = \|p_n\|_2 = \sqrt{x_n^2 + y_n^2 + z_n^2}. \quad (3.3)
\]

The distance between the acoustic source and the \(n\)-th microphone is denoted by

\[
D_n = \|p_n - p_s\|_2 = \sqrt{(x_n - x)^2 + (y_n - y)^2 + (z_n - z)^2}. \quad (3.4)
\]

With \(\tau_{ij}\) representing the time delay between the microphone pair \((i, j)\), and \(c\) denoting the signal speed (for the UAV localization problem, \(c\) equals the speed of sound), then

\[
d_{ij} = c \cdot \tau_{ij} = D_i - D_j \quad i, j = 1, 2, \ldots, N. \quad (3.5)
\]

The CRLB for the estimated acoustic source position is defined to be the inverse of the Fisher information matrix (FIM) \(F\) [95]. Modeling the errors of the TDEs as uncorrelated
and Gaussian with equal variance \( \sigma^2 \), the CRLB is [96]

\[
F^{-1} = (\sigma^2_\tau (G^T G)^{-1}
= \sigma^2_{RD} (G^T G)^{-1}.
\tag{3.7}
\]

\( \sigma^2_{RD} \) represents the variance of the RDs. In addition,

\[
F^{-1} = \sigma^2_{RD} (G^T G)^{-1}
\tag{3.8}
\]

with

\[
G = \begin{pmatrix} g_{ij} \\ \vdots \end{pmatrix}, \quad (i, j) \in J,
\tag{3.9}
\]

and

\[
g_{ij} = g_i - g_j
\tag{3.10}
\]

and

\[
g_i = \frac{p_i - p_s}{\|p_i - p_s\|_2} = \frac{p_i - p_s}{D_i}.
\tag{3.11}
\]

For a more detailed derivation see [97, p. 48]. For the CRLB to be defined for position estimates in \( \mathbb{R}^3 \), a microphone array of at least four sensors is necessary [98]. One can see in Equation 3.11 that the vector \( g_i \) has unit-length, and points from the signal source position to the sensor \( i \). That means that the vectors \( g_i \) depend on where the acoustic source is located with respect to the sensor array. Therefore, the CRLB is dependent on both the variance \( \sigma^2_{RD} \) of the RD estimates as well as on the geometry of the sensor array. Based on these findings, Yang and Scheuing propose in [99] to use the trace of the CRLB

\[
\Xi = tr(F^{-1})
= \sigma^2_{RD} tr((GG^T)^{-1})
\tag{3.12}
\tag{3.13}
\]

as a measure for the accuracy of the overall position estimate. \( \Xi \) summarizes the lower bound on the variances of the \( x, y \) and \( z \) coordinates of the position estimate into a single value. For the remainder of this thesis \( \Xi(p) \) is further referred to as the CRLB at the position \( p \).

The influence of the sensor array geometry onto the quality of the position estimates can be decoupled and analyzed individually:

\[
\Gamma = \sqrt{tr((GG^T)^{-1})}
\tag{3.14}
\]

Commonly, \( \Gamma \) is referred to as geometric dilution of precision (GDOP) [100].

### 3.2.6 Optimal sensor placement

#### 3.2.6.1 Uniform angular arrays (2D)

In order to gain the maximum localization accuracy, one has to find advantageous sensor array layouts which maximize the FIM \( F \) or minimize the GDOP or the CRLB. Based on the latter criterion, Yang and Scheuing derive in [98] the conditions for CRLB optimum sensor array geometries for \( J = J_0 \). For the 2D case, uniform angular arrays (UAAs)
satisfy these conditions. This class of microphone arrays can be described by

\[ g_i = \begin{bmatrix} \cos \alpha_i \\ \sin \alpha_i \end{bmatrix} \]  

(3.15)

with

\[ \alpha_i = \alpha_0 + \frac{2\pi}{N}(i - 1), \quad i = 1, 2, \ldots, N. \]  

(3.16)

### 3.2.6.2 Platonic solids and spherical code based array geometries (3D)

It is also shown in [98] that array geometries in which the direction vectors \( g_i \) point from the center of a platonic solid towards its vertices, are CRLB optimum arrays in \( \mathbb{R}^3 \). However, there are only five platonic solids (tetrahedron, octahedron, cube, icosahedron, dodecahedron), which restricts the amount of sensors to 4, 6, 8, 12 or 20 microphones respectively. To overcome this problem it is suggested in [96] to use spherical codes instead of platonic solids to describe the sensor array geometries. Despite solving the problem of the limited number of platonic solid based microphone arrays, most of the spherical code array geometries are not CRLB optimum, even though they approach the CRLB.

### 3.2.6.3 Remarks

For \( I = I_L \) the CRLB changes, and UAAs as well as platonic solid based array geometries are not CRLB optimum anymore. The implications of only using the LI TDOA measurements onto the CRLB optimal array geometry is discussed in depth in [99].

### 3.2.7 Implications of \( I \) onto the accuracy of the position estimates

The set \( I_{LI} \) includes all sensor pairs which provide the LI TDOA measurements. These LI TDOAs can be used in the error-free case to determine all other TDOAs of the sensor array. Even though this does not hold true in real-world scenarios anymore in which noise corrupts the TDOA measurements, it is quite common to see implementations of acoustic localization schemes use \( I = I_{LI} \). However, one should not default to this option, because \( I = I_{LI} \) is not the optimal solution for every application.

![Figure 3.3: Sensor array with \( N = 4 \) (left) and \( N = 7 \) (right) used for the analysis. For both arrays, the sensor pairings of \( I_{LI} \) have been connected by lines. The distances between the sensor \( p_1 \) and the remaining sensors is identical for both microphone layouts.](image-url)
Figure 3.4: Differences in CRLB: $\Delta \Xi_1$ (left) and $\Delta \Xi_2$ (right). One can see that using all available TDOAs ($N = 4 \Rightarrow 6$ available TDOAs) instead of only the LI ones ($N = 4 \Rightarrow 3$ LI TDOAs) improves the theoretically possible position accuracy only slightly (left). Increasing the number of microphones to be able to use the same amount of LI TDOAs ($6$ LI TDOAs $\Rightarrow N = 7$) however improves the accuracy significantly with increasing distance (right).

A small case study is used to show the pros and cons of both possibilities: The sensor array on the left of Figure 3.3 with $N = 4$ has altogether six TDOAs, three of them are LI. The microphone array on the right has $N = 7$, which results in overall 21 TDOAs with six of them being LI. One can analyze the difference in CRLB

$$
\Delta \Xi_1 = \Xi(N = 4, \mathcal{I} = \mathcal{I}_{LI}) - \Xi(N = 4, \mathcal{I} = \mathcal{I}_0)
$$

to gain insight about the difference in the possible location accuracy between $\mathcal{I} = \mathcal{I}_{LI}$ and $\mathcal{I} = \mathcal{I}_0$. The results for $\Delta \Xi_1$ are depicted in the left plot of Figure 3.4 for

$$
p_s = [x \ y \ 1]^T \text{m}, \quad x, y \in [-15, 15] \text{m},
$$

$$
\Delta x = \Delta y = 0.25 \text{m}, \quad \sigma_{RD}^2 = 0.001 \text{m}.
$$

$\Delta \Xi_1$ is positive for all $p_s$, which indicates that using all available TDOAs achieves a higher localization accuracy than solely relying on the LI TDOAs of a sensor array. However, if one determines the difference in CRLB

$$
\Delta \Xi_2 = \Xi(N = 4, \mathcal{I} = \mathcal{I}_{LI}) - \Xi(N = 7, \mathcal{I} = \mathcal{I}_{LI}).
$$

one can see (right plot of Figure 3.4) that using the six LI TDOAs of the second sensor array results in a much higher improvement of the localization accuracy than previously achieved with all available six TDOAs (a mixture of three LI and three LD TDOAs) of the microphone array for $N = 4$.

While the result looks like a clear recommendation for $\mathcal{I} = \mathcal{I}_{LI}$, one has to keep in mind that three additional microphones, amplifiers and ADCs are necessary to upgrade the left sensor array to $N = 7$. Depending on the implementation, time delays can be introduced due to synchronization issues: Usually banks of ADCs are available on a single ADC chip, ensuring a highly accurate and reliable synchronization between the individual ADCs. However, if multiple chips have to be used to digitize the sound signals, a synchronization between the ADC chips has to be implemented. Without a synchronization, time delays can be introduced which result in systematic errors in all TDEs which are based on microphones which have been sampled by separate ADC chips.
Chapter 3. An acoustic localization approach for precise landings on a helipad

The costs for the additional hardware (microphones, amplifiers, ADCs) to upgrade the sensor array from \( N = 4 \) to \( N = 7 \) are a one-time expenditure. Nevertheless, one has to be able to create a meaningful sensor array layout with the increased amount of microphones. With larger distance between the sensors, the TDOAs are less prone to noise in the system [90], which favors array geometries with microphones spread out. Optimal sensor placement with the increased amount of sensors might however not be possible if the application restricts the overall size of the sensor array.

To conclude, if one can only calculate a limited number of TDEs due to computational constraints, more sensors and \( J = J_{\text{LI}} \) should be considered first. If however the amount of sensors is restricted due to space or costs considerations, all available TDOA measurements should be used to gain the maximum possible localization accuracy. This of course does only hold true if the localization method in use allows \( J = J_0 \).

### 3.2.8 The consequences of using a reference microphone

Some localization approaches depend upon a reference sensor and therefore require \( J = J_{\text{LI}} \). Assuming the index \( r \) for the reference sensor, only the RDs \( d_{ir} \) with \( i = 1, 2, \ldots, N \) and \( i \neq r \) are then used for the calculation of the position estimate.

To achieve the maximum possible localization accuracy, one has to know which sensor of the given microphone array should be chosen as a reference sensor. This choice directly influences the CRLB, meaning that the theoretical limit of the localization accuracy is dependent on which sensor is chosen as reference microphone. Figure 3.5 shows a microphone array with \( N = 6 \) and highlights the sensor pairs providing the LI TDOAs which are used if either the sensor at position \( p_1 \) (red and black edges) or \( p_2 \) (red and green edges) is used as reference sensor.

\[ p_1 = \begin{bmatrix} 0 & 0 & 0.1 \end{bmatrix}^T \]
\[ p_2 = \begin{bmatrix} b & 0 & 0.0 \end{bmatrix}^T \]
\[ p_3 = \begin{bmatrix} b \cos(\alpha) & b \sin(\alpha) & -0.1 \end{bmatrix}^T \]
\[ p_4 = \begin{bmatrix} b \cos(2\alpha) & b \sin(2\alpha) & 0.2 \end{bmatrix}^T \]
\[ p_5 = \begin{bmatrix} b \cos(3\alpha) & b \sin(3\alpha) & -0.2 \end{bmatrix}^T \]
\[ p_6 = \begin{bmatrix} b \cos(4\alpha) & b \sin(4\alpha) & 0.1 \end{bmatrix}^T \]

with \( \alpha = \frac{2\pi}{5} \).

**Figure 3.5:** Microphone array used to calculate the CRLB plots shown in Figure 3.6. The sensor pairs connected by black lines are used if \( p_1 \) is chosen as reference sensor, while the sensor pairs connected by the green edges provide the LI TDOA measurements for reference sensor \( p_2 \). The TDOA provided by the sensor pair \( (p_1, p_2) \) is used in both cases.

Figure 3.6 shows a cut through the working area at \( z = 1 \) m. One can see that using \( p_2 \) as reference sensor reduces the overall possible localization accuracy at this particular elevation. If one has a reasonable idea about the position of the acoustic source and knows
the influence of picking a certain sensor as reference sensor, it is possible to electronically steer the microphone array to optimize the localization accuracy in the area the source is assumed. However, such a priori information about the source localization is usually not available in real-world scenarios.

It is important to note that the optimal choice of the reference sensor is highly dependent on the application. If one knows e.g. that the source cannot be located in a certain area of the environment due to walls or natural barriers, the choice of a reference sensor leading to an unbalanced CRLB around the sensor array can achieve better results in the desired area but is not the optimal choice to maximize the CRLB of the overall area.

### 3.3 Review of passive TDOA-based position estimators for $\mathbb{R}^3$

There is a rich literature on passive TDOA-based acoustic localization schemes. The following section gives a review of the most representative methods (approaches which are commonly used as reference algorithms) and state-of-the-art methods. The algorithms are introduced with the objective of giving the reader a general idea about the inner workings of each method, their advantages and shortcomings, as well as an overview of how the methods relate to each other. The introduction to each approach is done with respect to acoustic localization, even though the corresponding papers might discuss the performance of the algorithms in other domains. The reader is advised to consult the referenced original papers when an interest in a more detailed derivation of a specific approach, or its usage in other application areas exists.

The methods are classified according to their underlying models. Each model is first explained to the reader, followed by the introduction of the algorithms associated with the model. After the introduction, a comparison of the models and methods follows. An overview of their pros and cons with respect to acoustic localization is given, and their suitability for the helicopter localization task is discussed. A detailed performance analysis can be found in Sections 3.6 (simulation) and 3.7 (experiments).

For the introduction of the methods it is assumed that the time delay estimation step has already been performed, and that noise affected TDOA measurements are available.
3.3.1 Approaches using the hyperbolic model

Based on Equations 3.4 and 3.6 one can formulate

\[ d_{ij} = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} - \sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2} \quad i, j = 1, 2, \ldots, N. \]  

Equation 3.18 shows that the range differences \( d_{ij} \) specified by Equation 3.17 are equal to the known function \( f_{ij}(x, y, z) \) in the error-free case. With only the measured quantities

\[ f_{ij}(x, y, z) = d_{ij} = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} - \sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2} \]  

The error-free case presented in Equation 3.17 shows that each RD \( d_{ij} \) and its corresponding microphone positions \( p_i \) and \( p_j \) define a hyperboloid of two sheets on which the sound source is located (see Figure 3.7, left). The microphone positions are the focal points of the hyperboloid. Three hyperbolic curves based on three TDOAs measured at the same time via different microphone pairs, intersect in the acoustic source position (see Figure 3.7, right). The challenge of TDOA-based localization is the non-linear relationship between the TDOA measurements and the position of the signal source.

3.3.1.1 Foy’s algorithm (Taylor-series)

Iterative solutions have been proposed to estimate the position of the sound source in the presence of measurements errors. A popular solution is the one given by Foy in [101] utilizing the Taylor-series method. Torrieri performed a statistical analysis of the approach in [102].
3.3. Review of passive TDOA-based position estimators for $\mathbb{R}^3$

$d_{ij}$ of the actual range differences $d_{ij}$ available, one can write

$$f_{ij}(x, y, z) = \hat{d}_{ij} - \epsilon_{ij}$$

(3.19)

where $\epsilon_{ij}$ is the measurement error in $\hat{d}_{ij}$. The superscript $\hat{\cdot}$ will be further used to denote a measured or estimated quantity. The non-linear function $f_{ij}(x, y, z)$ can be linearized by a Taylor-series expansion about the position

$$p'_s = [x' \ y' \ z']^T,$$

(3.20)

which represents an initial guess of the actual source position $p_s$. The Taylor-series expansion of $f_{ij}(x, y, z)$ using only the terms of the zeroth and first order is

$$t_{ij}(x, y, z) \simeq f_{ij}(x, y, z) = f_{ij}(x', y', z') + \gamma_{ijx}\delta_x + \gamma_{ijy}\delta_y + \gamma_{ijz}\delta_z$$

(3.21)

with

$$\gamma_{ijx} = \frac{\partial f(x', y', z')}{\partial x} = \frac{x' - x_i}{\alpha} - \frac{x' - x_j}{\beta}$$

$$\gamma_{ijy} = \frac{\partial f(x', y', z')}{\partial y} = \frac{y' - y_i}{\alpha} - \frac{y' - y_j}{\beta}$$

$$\gamma_{ijz} = \frac{\partial f(x', y', z')}{\partial z} = \frac{z' - z_i}{\alpha} - \frac{z' - z_j}{\beta}$$

where

$$\alpha = \sqrt{(x_i - x')^2 + (y_i - y')^2 + (z_i - z')^2}$$

$$\beta = \sqrt{(x_j - x')^2 + (y_j - y')^2 + (z_j - z')^2}$$

and

$$\delta_x = x - x'$$

$$\delta_y = y - y'$$

$$\delta_z = z - z'.$$

If $p'_s$ is in close proximity to the actual source position $p_s$, Equation 3.21 resembles an accurate approximation. Equation 3.19 can then be rewritten as

$$\gamma_{ijx}\delta_x + \gamma_{ijy}\delta_y + \gamma_{ijz}\delta_z = \hat{d}_{ij} - f_{ij}(x', y', z') - \epsilon_{ij}.$$  

(3.23)

Equation 3.23 can be expressed in matrix-vector form as

$$M\delta = d - \epsilon$$

(3.24)
Chapter 3. An acoustic localization approach for precise landings on a helipad

with

\[
M = \begin{bmatrix}
\gamma_{ijx} & \gamma_{ijy} & \gamma_{ijz} \\
\vdots & \vdots & \vdots \\
\end{bmatrix}, \quad \delta = \begin{bmatrix}
\delta_x \\
\delta_y \\
\delta_z \\
\end{bmatrix}, \quad d = \begin{bmatrix}
\hat{d}_{ij} - f_{ij}(x', y', z') \\
\vdots \\
\end{bmatrix}, \quad e = \begin{bmatrix}
e_{ij} \\
\vdots \\
\end{bmatrix},
\]

where \((i, j) \in \mathcal{I}\). Each row of the system of equations is based on the positions of two microphones and their corresponding RD measurement. Equation 3.24 can be solved via the least squares calculation

\[
\delta = (M^T \xi^{-1} M)^{-1} M^T \xi^{-1} d
\]

where \(\xi\) is the covariance matrix. With \(\xi\) dependent on the actual, yet unknown source position \(p_s\), \(\xi\) is usually removed from Equation 3.25 assuming an identity matrix. In other words, the microphones are assumed to suffer equally from the present noise (valid assumption, but usually not true in reality). Using the assumption, Equation 3.25 can be written as

\[
\delta = (M^T M)^{-1} M^T d
\]

where the superscript \((\cdot)^\dagger\) denotes the Moore-Penrose pseudoinverse. One can now iterate over Equation 3.27, replacing \(p_s\) after each iteration with

\[
p'_s \leftarrow p_s + \delta'
\]

where \(p''_s\) stands for \(p'_s\) used in the next iteration, \(p'_s\) is the approximation of \(p_s\) which has been used for the current iteration, and \(\delta'\) represents the result vector of the current iteration. The iterations will have converged when the values of \(\delta\) get close to zero.

Foy’s algorithm provides a very accurate result if the method converges. It is simple to detect if the approach converged \((\delta \rightarrow 0)\), but a satisfactory result is only achieved if the method does not converge to a local minimum. In real-world scenarios it is problematic to find an estimate for \(p_s\) close to its actual position to avoid local minima. Usually, a two step approach is implemented in which a closed-form position estimator is used to provide \(p'_s\) which is then refined by the iterative algorithm. Foy’s algorithm works both for \(\mathcal{J} = \mathcal{J}_0\) and \(\mathcal{J} = \mathcal{J}_{LL}\) for all \(N \geq 4\).

3.3.1.2 A misconception about iterative algorithms

Papers presenting closed-form approaches for the localization problem often present available iterative methods with a note about their unsuitability for real-time implementations. As reasons therefore are usually given:

- The need of a suitable starting point
- Convergence is not guaranteed
- High computational load
While all of these points can be considered as disadvantages of iterative methods, they do not necessarily prohibit a real-time implementation of the approaches. If one considers for example Foy’s algorithm, it is fair to say that the method by itself is not very useful in a practical setup, due to the need of an initial guess close to \( p_s \) to avoid local minima. However, this problem can be overcome by taking the previously introduced two-step approach.

Of course there is still the valid point of the non-guaranteed convergence. For Foy’s method one can check the convergence by monitoring \( \delta \). A more general procedure which is usually used for other iterative methods is to calculate the Euclidean distance between the results of two consecutive runs of the algorithm. The result of this simple calculation gives a good indication whether the algorithm converges or not. A threshold for this distance has to be defined, which can be found via simulations.

If a non-convergence case is detected, the result of the closed-form approach can be used as position estimate for the further calculations. This is not a problem, because the initial position estimate is assumed to be close to the actual source position to avoid local minima. Iterative solutions are generally only used to refine already available position estimates.

A second threshold can be used to determine if the iterative algorithm converges to a local minima. The Euclidean distance between the initial position estimate and the result of the iterative approach can be compared to this upper threshold to detect if the position estimate of the iterative approach converged to a point too far away from the initial estimate. If that is the case, the result of the closed-form algorithm should be used for the further calculations.

Real-time systems have to be fully deterministic. Known worst-case execution times in conjunction with necessary deadlines provide the fundamental means to determine a suitable scheduler for the overall system, allowing strict deadlines to be met. In case of the introduced two-step approach, an upper limit for the execution time of the closed-form algorithm can be determined. The same can be done for the iterative method, if a constant maximum amount of iterations is specified. The high computational load is not a real-time issue as long as a suitable computing platform can be found to calculate the predefined maximum number of iterations times the worst-case execution time per iteration faster than its deadline.

To summarize, iterative approaches are very useful to refine initial position estimates. In case of problems (non-convergence, convergence to a local minima), the initial position estimate can be used for the further calculations. The thresholds which are necessary to discover problem cases can be found via simulations. This is already common practice to get an idea of the expected localization accuracy from the chosen setup. Iterative algorithms are well suited for real-time implementations, even though they require some additional effort.

### 3.3.2 Approaches using the linear model

The linear model was introduced by Schmidt in [103]. He realized that the known positions of three sensors combined with their corresponding RDs result in a plane on which the acoustic source is positioned if no errors are present in the system. Geometrically interpreted, the plane represents the major axis of a general conic (see Figure 3.8) in the Euclidean 3-space.
In contrast to the hyperbolic model where the sensors are the focal points of the resulting hyperboloid, the microphones are here located on a general conic. In addition, Schmidt showed that in $\mathbb{R}^2$ one of the focal points of the general conic is the position of the acoustic source.

It is important to realize that the linear model is not based on any assumptions or linearization steps which introduce errors. The model is solely linear due to the interpretation of the geometric properties of the hyperbolic localization problem. In contrast, the previously introduced approach by Foy linearized the problem function by approximating it with a Taylor-series expansion, which introduces a linearization error into the system (which is then assumed to be negligible).

### 3.3.2.1 Schmidt’s algorithm: Plane intersection (PX)

Schmidt described in [103, 104] the linear model and introduced a solution which he termed location on the conic axis (LOCA). The major axis of the general conic on which three sensors $i$, $j$, and $k$ are positioned, can be described by

$$\alpha_{ijk}x + \beta_{ijk}y + \gamma_{ijk}z = \delta_{ijk} \quad (3.29)$$

with

$$\alpha_{ijk} = x_id_{kj} + x_jd_{ik} + x_kd_{ji} \quad (3.30)$$
$$\beta_{ijk} = y_id_{kj} + y_jd_{ik} + y_kd_{ji} \quad (3.31)$$
$$\gamma_{ijk} = z_id_{kj} + z_jd_{ik} + z_kd_{ji} \quad (3.32)$$

and

$$\delta_{ijk} = \frac{1}{2}(d_{ij}d_{kj}d_{ik} + R_i^2d_{kj} + R_j^2d_{ik} + R_k^2d_{ji}). \quad (3.33)$$

If five sensors are available, three planes defined by Equation 3.29 can be calculated. The intersection point of the position estimation planes is the unknown source location in the
3.3. Review of passive TDOA-based position estimators for $\mathbb{R}^3$

error-free case. Therefore, the algorithm has been termed \textit{plane intersection method} in the literature. With errors present in the system, Equation 3.29 can be expressed in matrix-vector form as

$$M \cdot \hat{p}_s = d - \epsilon$$

(3.34)

with

$$M = \begin{bmatrix} \alpha_{ijk} & \beta_{ijk} & \gamma_{ijk} \\ \vdots & \vdots & \vdots \end{bmatrix}, \quad d = \begin{bmatrix} \delta_{ijk} \\ \vdots \end{bmatrix}, \quad \epsilon = \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \end{bmatrix}, \quad (i,j,k) \in \mathcal{R}.$$  

All sensor triplets $(i,j,k)$ used for the calculation of the position estimate form the set $\mathcal{R}$, while $\mathcal{R}_0$ is defined to be the set of all available sensor triplets of the given sensor array. With $\mathcal{R} = \mathcal{R}_0$, a solution to Equation 3.34 can be found via the least squares calculation

$$\hat{p}_s = (M^T M)^{-1} M^T d$$

(3.35)

$$\hat{p}_s = M^d d$$

(3.36)

In the error-free case

$$\epsilon = d_{kj} + d_{ik} + d_{ji}$$

(3.37)

$$= D_k - D_j + D_i - D_k + D_j - D_i$$

(3.38)

$$= 0.$$  

(3.39)

However, with only the noise affected RD estimates available for the calculation of the position estimate

$$\epsilon = \hat{d}_{kj} + \hat{d}_{ik} + \hat{d}_{ji}$$

(3.40)

$$\|\epsilon\|_2 \geq 0.$$  

(3.41)

To compensate for the measurement error $\epsilon$, Schmidt introduces an error correction step in which each range difference is reduced by $\frac{1}{3}\epsilon$, assuming an equal error for $\hat{d}_{kj}$, $\hat{d}_{ik}$ and $\hat{d}_{ji}$. The TDOA averaging results in an increased performance of the overall algorithm (see [103], Figure 13) by reducing the number of outliers in the resulting position estimates.

If one only wants to use LI TDOAs for the calculation of the position estimate, the parameters for $M$ and $d$ have to be reformulated. Assuming $(1,2)$ as reference sensor pair and

$$d_{kj} = -d_{ik} - d_{ji},$$

(3.42)

Equations 3.30 to 3.33 can be rewritten as

$$\alpha_{12k} = x_1(-d_{1k} - d_{21}) + x_2d_{1k} + x_3d_{21}$$

(3.43)

$$\beta_{12k} = y_1(-d_{1k} - d_{21}) + y_2d_{1k} + y_3d_{21}$$

(3.44)

$$\gamma_{12k} = z_1(-d_{1k} - d_{21}) + z_2d_{1k} + z_3d_{21}$$

(3.45)

69
Chapter 3. An acoustic localization approach for precise landings on a helipad

and

\[ \delta_{12k} = \frac{1}{2}(d_{21}(-d_{1k} - d_{21})d_{1k} + R_2^2(-d_{1k} - d_{21}) + R_2^2 d_{1k} + R_2^2 d_{21}) \]  

(3.46)

with

\[ k = 3, 4, \ldots, N. \]  

(3.47)

The sensor triplets (1, 2, k) form the set \( \mathcal{R}_{LL} \).

Schmidt’s approach has the advantage that the location of the sound source can be determined via a least squares calculation when errors are present in the system. In addition, the algorithm works both with \( \mathcal{R} = \mathcal{R}_0 \) and \( \mathcal{R} = \mathcal{R}_{LL} \) for \( N \geq 5 \). However, the TDOA averaging can only be used with \( \mathcal{R} = \mathcal{R}_0 \). On the negative side, Schmidt’s plane intersection algorithm does not work if all sensors are coplanar. In this case the localization planes defined by Equation 3.29 are perpendicular to the sensor plane and do intersect in a common line, but not in a common point anymore. Mathematically, \( M \) is singular for coplanar sensor arrays.

### 3.3.2.2 Fang’s algorithm

Fang presented in [105] a closed-form solution for the multilateration problem. His approach utilizes three TDOA measurements gathered from four sensors to find a solution in \( \mathbb{R}^3 \). First, the four sensor microphone array with the microphones \( i, j, k, \) and \( l \) is divided into two sets of three sensors \((i, j, k)\) and \((i, j, l)\). This represents \( \mathcal{R} = \mathcal{R}_{LL} \) for \( N = 4 \). Then the first set is translated and rotated into a Cartesian coordinate system with

- one microphone position \( p_{i,\text{fang}} = [0 \ 0 \ 0]^T \) being the origin
- the second microphone position \( p_{j,\text{fang}} = [x_{j,\text{fang}} \ 0 \ 0]^T \) lying on the x axis, and
- the third microphone position being \( p_{k,\text{fang}} = [x_{k,\text{fang}} \ y_{k,\text{fang}} \ 0]^T \).

This coordinate system will be further referred to as \( cs_{\text{fang}} \), and the world coordinate system with the origin chosen at an arbitrary point in space will be further referred to as \( cs_{\text{world}} \). To make it easier to distinguish between the variables in the two coordinate systems, all variables in \( cs_{\text{fang}} \) will have the same basic name they have in \( cs_{\text{world}} \), but with the subscript \( \text{fang} \) attached. If the value of a variable in both coordinate systems is equal, no subscript will be attached. Equation 3.17 for the two microphone pairs \((i, j)\) and \((i, k)\) can be written in \( cs_{\text{fang}} \) as

\[ d_{ij} = D_{i,\text{fang}} - D_{j,\text{fang}} \]  

(3.48)

\[ = \sqrt{x_{f,\text{fang}}^2 + y_{f,\text{fang}}^2 + z_{f,\text{fang}}^2} \]  

\[ = \sqrt{(x_{j,\text{fang}} - x_{f,\text{fang}})^2 + y_{f,\text{fang}}^2 + z_{f,\text{fang}}^2} \]  

(3.49)

\[ d_{ik} = D_{i,\text{fang}} - D_{k,\text{fang}} \]  

(3.50)

\[ = \sqrt{x_{f,\text{fang}}^2 + y_{f,\text{fang}}^2 + z_{f,\text{fang}}^2} \]  

\[ = \sqrt{(x_{k,\text{fang}} - x_{f,\text{fang}})^2 + (y_{k,\text{fang}} - y_{f,\text{fang}})^2 + z_{f,\text{fang}}^2}. \]  

(3.51)
Transposing the term for $D_{i,fang}$ to the left-hand side of the Equations 3.49 and 3.51, squaring and simplifying, one obtains the following equations:

$$\frac{d_{ij}^2 - x_{j,fang}^2 + 2x_{fang}x_{i,fang}}{d_{ij}} = 2\sqrt{x_{fang}^2 + y_{fang}^2 + z_{fang}^2}$$ (3.52)

$$\frac{d_{ik}^2 - k^2 + 2x_{k,fang}x_{fang} + 2y_{k,fang}y_{fang}}{d_{ik}} = 2\sqrt{x_{fang}^2 + y_{fang}^2 + z_{fang}^2}$$ (3.53)

with

$$k = \sqrt{x_{k,fang}^2 + y_{k,fang}^2}.$$

Equating 3.52 and 3.53 and simplifying afterwards, one obtains the following equation:

$$y_{fang} = \alpha \cdot x_{fang} + \beta$$ (3.54)

with

$$\alpha = \frac{1}{y_{k,fang}} \left( \frac{d_{ik}x_{fang}}{d_{ij}} - x_{k,fang} \right)$$ (3.55)

$$\beta = \frac{1}{2y_{k,fang}} \left( k^2 - d_{ik}^2 + d_{ik}d_{ij}(1 - (\frac{x_{fang}}{d_{ij}})^2) \right).$$ (3.56)

Equation 3.54 defines a plane orthogonal to the $x$-$y$ plane in $cs_{fang}$. Even though not discussed in Fang’s paper, this intermediate step has a close relationship to Schmidt’s method. As shown in Appendix A, the planes calculated by Equation 3.54 are actually identical to the ones calculated by Schmidt in Equation 3.29. However, Schmidt’s approach for the intermediate step of calculating position estimation planes can be seen as the general solution to the one given by Fang, because Schmidt’s method does not require any transformation (translation and rotation) prior to the execution of the algorithm. This makes Schmidt’s solution also computationally less demanding.

After calculating the position estimation planes, Fang and Schmidt’s approach take different directions. Fang substitutes Equation 3.54 back into 3.52 and obtains

$$z_{fang}^2 = \gamma x_{fang}^2 + \delta x_{fang} + \eta$$ (3.57)

with

$$\gamma = -(1 - (\frac{x_{fang}}{d_{ij}})^2 + \alpha^2)$$

$$\delta = x_{f,fang}(1 - (\frac{x_{fang}}{d_{ij}})^2) - 2\alpha\beta$$

$$\eta = (\frac{d_{ij}}{2})^2(1 - (\frac{x_{fang}}{d_{ij}})^2)^2 - \beta^2$$

resulting in

$$p_{s,fang} = x_{fang}q + (\alpha x_{fang} + \beta)r \pm (\sqrt{\gamma x_{fang}^2 + \delta x_{fang} + \eta}s)$$ (3.58)
where $q$, $r$, and $s$ are the unit vectors pointing into the direction of the $x$, $y$, and $z$ axis of $cs_{f_{\text{ang}}}$ respectively. Applying the same ideas and equations to the sensor triplet $(i, j, l)$ than one did previously for the sensor triplet $(i, j, k)$, one gets a second equation

$$p_{s, f_{\text{ang}}} = x_{f_{\text{ang}}} q + (\alpha^* x_{f_{\text{ang}}} + \beta^*) r' \pm (\sqrt{\gamma^* x_{f_{\text{ang}}}^2 + \delta^* x_{f_{\text{ang}}} + \eta^*}) s'$$  (3.59)

where the $q$, $r'$, and $s'$ are the unit vectors pointing into the direction of the $x$, $y$, and $z$ axis of $cs_{f_{\text{ang}}}'$ defined by the sensor triplet $(i, j, k)$. The coordinate systems $cs_{f_{\text{ang}}}$ and $cs_{f_{\text{ang}}}'$ share the same $x$ axis. Therefore, the vector $q$ is shared between Equation 3.58 and 3.59.

Equations 3.58 and 3.59 are only dependent on the unknown $x$. Transposing $x_{f_{\text{ang}}} q$ to the left-hand side of the Equations 3.58 and 3.59, equating and multiplying by the unit vector $r'$, one obtains

$$a^* x_{f_{\text{ang}}} + \beta' = (a x_{f_{\text{ang}}} + \beta) (rr') \pm (\sqrt{\gamma^* x_{f_{\text{ang}}}^2 + \delta^* x_{f_{\text{ang}}} + \eta^*}) (sr').$$  (3.60)

Squaring and simplifying, one gets

$$\iota x_{f_{\text{ang}}}^2 + \kappa x_{f_{\text{ang}}} + \nu = 0$$  (3.61)

with

$$\iota = \gamma (sr')^2 - (a^* - a(rr'))^2$$
$$\kappa = \delta (sr')^2 - 2(a^* - a(rr'))(\beta' - \beta(rr'))$$
$$\nu = \eta (sr')^2 - (\beta' - \beta(rr'))^2.$$  

The two possible solutions for $x_{f_{\text{ang}}}$ can be calculated via the quadratic formula. Inserting the values back into Equations 3.54 and 3.57 respectively, one gains the $y$ and $z$ coordinates of the possible values of $x_{f_{\text{ang}}}$. Extraneous solutions which have been introduced during the calculations can be detected by inserting the result candidates back into Equation 3.49. Only one of the two possible solutions will satisfy the equations. For a planar microphone array,

$$\iota = -(a^* a)^2$$
$$\kappa = -2(a^* a)(\beta' \beta)$$
$$\nu = -(\beta' \beta)^2.$$  

This results in the single solution

$$x_{f_{\text{ang}}} = -\frac{\kappa}{2\iota}.$$  (3.62)

In a last step the position estimate gained by Fang’s approach must be transferred back to the coordinate system $cs_{\text{world}}$.

A disadvantage of Fang’s approach is that a coordinate transformation is necessary if the coordinate system $cs_{\text{world}}$ does not coincide with $cs_{f_{\text{ang}}}$. In addition, the algorithm cannot make use of any additional TDOA measurements. On the positive side, Fang’s approach
3.3. Review of passive TDOA-based position estimators for $\mathbb{R}^3$

delivers results for planar microphone arrays and the method can be considered optimum in the maximum likelihood (ML) sense [106].

3.3.2.3 Bucher and Misra’s algorithm

Bucher and Misra introduce in [107] a solution for the hyperbolic localization problem using four TDOAs gathered from four sensors. The derivation of the algorithm is straightforward. Starting off with Equation 3.17, transposing the term for $D_i$ to the left hand side and squaring results in

$$d_{ij}^2 - 2d_{ij}D_i + (x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 = (x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2$$

(3.63)

Expanding the squared terms allows one to eliminate the $x^2$, $y^2$, and $z^2$ terms. Solving for $D_i$ and simplifying produces

$$D_i = \frac{1}{2d_{ij}}(d_{ij}^2 + R_i^2 - R_j^2 + 2x_ix + 2y_jy + 2z_jz - 2x_ix - 2y_jy - 2z_jz),$$

(3.64)

an equation dependent on the two microphone positions $p_i$ and $p_j$ as well as the RD $d_{ij}$ derived from the TDOA $\tau_{ij}$. Similarly, one can produce another equation of this form for the microphone pair $(i, k)$. Equating the corresponding equations for the sensor pairs $(i, j)$ and $(i, k)$ leads to

$$y = \alpha x + \beta z + \gamma$$

(3.65)

with

$$\alpha = \frac{d_{ik}(x_j - x_i) - d_{ij}(x_k - x_i)}{d_{ij}(y_k - y_i) - d_{ik}(y_j - y_i)}$$

(3.66)

$$\beta = \frac{d_{ik}(z_j - z_i) - d_{ij}(z_k - z_i)}{d_{ij}(y_k - y_i) - d_{ik}(y_j - y_i)}$$

(3.67)

$$\gamma = \frac{d_{ik}(d_{ij}^2 + R_i^2 - R_j^2) - d_{ij}(d_{ik}^2 + R_i^2 - R_k^2)}{2(d_{ij}(y_k - y_i) - d_{ik}(y_j - y_i))}. $$

(3.68)

Equation 3.65 represents a plane based on the three sensor positions $p_i$, $p_j$, and $p_k$. As shown in Appendix B, the position estimation plane defined by Equation 3.65 is identical to the plane used by Schmidt (defined in Equation 3.29) for his plane intersection method. Thus, the intermediate step of calculating position estimation planes is shared between Schmidt’s, Fang’s and Bucher and Misra’s algorithms.

Using the Equations 3.17, and 3.63 to 3.65 for the microphone triplet $(k, j, l)$ produces another position estimation plane with the coefficients

$$\alpha' = \frac{d_{kl}(x_j - x_k) - d_{kj}(x_l - x_k)}{d_{kj}(y_l - y_k) - d_{kl}(y_j - y_k)}$$

$$\beta' = \frac{d_{kl}(z_j - z_k) - d_{kj}(z_l - z_k)}{d_{kj}(y_l - y_k) - d_{kl}(y_j - y_k)}$$
and
\[
\gamma' = \frac{d_{kl}(d^2_{kj} + R^2_k - R^2_j) - d_{kj}(d^2_{kl} + R^2_k - R^2_l)}{2(d_{kj}(y_i - y_k) - d_{kl}(y_j - y_k))}.
\]

Equating the two resulting planes yields
\[
x = \delta z + \eta
\]  
(3.69)

with
\[
\delta = \frac{\beta' - \beta}{\alpha - \alpha'} \quad \quad \quad \eta = \frac{\gamma' - \gamma}{\alpha - \alpha'}
\]

Substituting Equation 3.69 back into 3.65 leads to
\[
y = \iota z + \kappa
\]  
(3.70)

with
\[
\iota = \alpha \delta + \beta \quad \quad \quad \kappa = \alpha \eta + \gamma
\]

Substituting Equations 3.69 and 3.70 back into Equation 3.64 for the microphone pair \((i, k)\) and rearranging the equation leads to
\[
\phi z^2 - \chi z + \psi = 0
\]  
(3.71)

with
\[
\rho = d^2_{ik} + R^2_i - R^2_k + 2(x_k - x_i)\eta + 2(y_k - y_i)\kappa
\]
\[
\nu = 2((x_k - x_i)\delta + (y_k - y_i)\iota + (z_k - z_i))
\]

and
\[
\phi = 4d^2_{ik}(\delta^2 + \iota^2 + 1) - \nu^2 \quad \quad \quad \quad \chi = 8d^2_{ik}(\delta(x_i - \eta) + i(y_i - \kappa) + z_i) + 2\nu\rho
\]
\[
\psi = 4d^2_{ik}(x_i^2 - \eta^2 + (y_i - \kappa)^2 + z_i^2) - \rho^2.
\]

The possible \(z\) coordinates can be calculated by solving Equation 3.71 via the quadratic formula. Equations 3.69 and 3.70 give the corresponding \(x\) and \(y\) coordinate.

An advantage of this method is that it does not need any prior transformation of the microphone positions into a new coordinate system. The algorithm also works well with planar sensor arrays. A disadvantage of the algorithm is that it is not possible to accommodate any additional TDOA measurements to reduce the overall positioning error. Furthermore, the method needs an additional LD TDOA measurement next to the three mandatory LI TDOAs to solve the hyperbolic localization problem.
3.3. Review of passive TDOA-based position estimators for $\mathbb{R}^3$

3.3.3 Approaches using the spherical model

The spherical model is based on the assumption of an omnidirectional sound source and an ideal propagation medium. In that case, the sound waves propagate in form of a sphere away from $p_s$. Assuming sensor one as reference microphone, a sphere with the sound source as center and with radius $D_1$ will have the position vector $p_1$ on its surface. For $i = 2, 3, \ldots, N$ the range differences $d_{i1}$ can now be seen as the distances from the sensor positions $p_i$ to the surface of the sphere (see Figure 3.9).

If one now moves the origin of the sphere into the reference sensor, and uses $N - 1$ more spheres with the microphone positions $p_i$ as center and the radius $(D_1 + d_{i1})$, the intersection point of the spheres will be the signal source position in the error-free case.

The general idea of the spherical model is to compensate for a problem intrinsic to the hyperbolic model: Small errors in the eccentricity of one of the hyperboloids used for the position estimation can result in large errors of the position estimate. This does not hold true for the intersection of spheres: Small errors in the TDEs lead only to minor changes in the radii of the spheres.

For the introduction of the following methods, sensor one is always chosen as reference microphone. Mathematically, the basic idea of the spherical model can then be described as follows: Squaring Equation 3.4 and using the relationship specified in Equation 3.6 one can write

$$(x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2 = D_1^2$$  \hspace{1cm} (3.72)

$$(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 = (d_{i1} + D_1)^2, \quad i = 2, 3, \ldots, N.$$  \hspace{1cm} (3.73)

Equation 3.72 and 3.73 describe spheres with radius $D_1$ and $(d_{i1} + D_1)$ around the points $p_1$ and $p_i$ respectively. Expanding Equation 3.72 and solving for $R_i^2$ yields

$$R_i^2 = -R_1^2 + D_1^2 + 2x_1x + 2y_1y + 2z_1z.$$  \hspace{1cm} (3.74)

Substituting Equation 3.74 into 3.73 results in

$$R_i^2 - R_1^2 - d_{i1}^2 - 2D_1d_{i1} = 2x_1x + 2y_1y + 2z_1z - 2x_1x - 2y_1y - 2z_1z$$

$$= 2(x(x_i - x_1) + y(y_i - y_1) + z(z_i - z_1)).$$  \hspace{1cm} (3.75)
Equation 3.75 gives the wrong impression of being linear in $p_s$. However, $D_1$ is also dependent on $p_s$ via Equation 3.4. This is the intrinsic dilemma of the spherical model: Not only the source position is unknown, but also the distance from $p_s$ to the reference microphone. A linearization error is usually introduced by the approaches using the spherical model by assuming no dependency between $p_s$ and $D_1$.

A problem shared by all approaches based on the spherical model described in this thesis is that no redundant information can be used to further improve the position estimates (only $I = I_{LL}$ is possible, not $I = I_0$). This phenomenon can be traced back to the idea of using a reference sensor, and only utilizing range differences of the form $d_{ir}$ (with sensor $r$ being the reference sensor). A possible workaround to this problem is to calculate multiple position estimates with a different reference sensor for each iteration, and then combining them in a final step to a single position estimate. That way all TDOA measurements can be utilized, however the computational load increases significantly (especially with increasing $N$).

### 3.3.3.1 Schau and Robinson’s algorithm: Spherical intersection (SX)

In their approach described in [108], Schau and Robinson move the position of the reference sensor into the origin of the coordinate system in use. That results in $R_1 = 0$ and $R_s = D_1$, which simplifies Equation 3.75 to

$$R^2_i - d^2_{i1} - 2R_s d_{i1} = 2(x_i x + y_i y + z_i z). \tag{3.77}$$

In matrix-vector form this equals

$$\delta - 2R_s d = 2M p_s \tag{3.78}$$

with

$$\delta = \begin{bmatrix} R^2_2 - d^2_{21} \\ R^2_3 - d^2_{31} \\ \vdots \\ R^2_N - d^2_{N1} \end{bmatrix}, \quad d = \begin{bmatrix} d_{21} \\ d_{31} \\ \vdots \\ d_{N1} \end{bmatrix}, \quad M = \begin{bmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ \vdots & \vdots & \vdots \\ x_N & y_N & z_N \end{bmatrix}.$$

There are two unknowns in Equation 3.78: The radius of the sphere $R_s$ on the left-hand side, and the source position $p_s$ on the right-hand side. If one assumes that $R_s$ is known and introduces errors into the system, one can use the least squares calculation

$$e = \delta - 2R_s \hat{d} - 2M \hat{p}_s \tag{3.79}$$

$$\hat{p}_s = \frac{1}{2} M^\dagger (\hat{\delta} - 2R_s \hat{d}) \tag{3.80}$$

to gain $p_s$. Equation 3.2 can be written in matrix-vector form as

$$R_s = (p_s^T p_s)^{-1} \tag{3.81}$$

Substituting 3.80 into 3.81 and expanding the equation yields in

$$\alpha R_s^2 + \beta R_s - \gamma = 0 \tag{3.82}$$

76
where \( R_s \geq 0 \) and
\[
\begin{align*}
\alpha &= 4 - 4 \hat{d}^T (M^\dagger)^T M^\dagger \hat{d} \\
\beta &= 2 \hat{d}^T (M^\dagger)^T M^\dagger \hat{d} + 2 \delta^T (M^\dagger)^T M^\dagger \delta \\
\gamma &= \delta^T (M^\dagger)^T M^\dagger \delta.
\end{align*}
\]
Equation 3.82 can now be solved via the well known quadratic formula, which results in zero to two solutions for \( R_s \). Inserting the available values back into Equation 3.80 results in position estimates for the source location.

Schau and Robinson’s algorithm needs at least four sensors and three TDOAs to provide a solution in \( \mathbb{R}^3 \). The method has the problem that the correct position estimate has to be picked from the available set of possible results. In addition, a coordinate transformation is necessary before and after the algorithm if the origin of the coordinate system does not coincide with the position of the reference sensor. The method is also unsuitable for planar microphone arrays (in that case \( M \) is singular). On the positive side, the matrix \( M \) only changes if the sensor layout is changed. Therefore, under normal circumstances it is only necessary to calculate the computationally expensive matrix inverse once.

### 3.3.3.2 Smith and Abel’s algorithm: Spherical interpolation (SI)

To overcome the disadvantage of the SX method of having to pick the right position estimate after the calculation, Smith and Abel introduced the SI method [109]. The derivation of the approach follows Schau and Robinson’s algorithm up to the least squares calculation
\[
e = \delta - 2 R_s \hat{d} - 2 M \hat{p}_s \tag{3.83}
\]
\[
\hat{p}_s = \frac{1}{2} M^\dagger (\delta - 2 R_s \hat{d}) \tag{3.84}
\]
Smith and Abel now substitute Equation 3.84 back into Equation 3.83, which yields
\[
\begin{align*}
e' &= \delta - 2 R_s \hat{d} - MM^\dagger (\delta - 2 R_s \hat{d}) \\
&= M^\perp (\delta - 2 R_s \hat{d}) \tag{3.85}
\end{align*}
\]
with
\[
M^\perp = (I - MM^\dagger).
\]
The resulting error term \( e' \) is linear in the unknown \( R_s \). \( M^\perp \) is an idempotent matrix and can be understood as orthogonal projection operator to \( M \) (for a detailed discussion about \( M^\perp \) see [110]). Solving Equation 3.86 using a second least squares calculation yields in
\[
R_s = \frac{\hat{d}^T M^\perp \delta}{2 \hat{d}^T M^\perp \hat{d}} \tag{3.87}
\]
The result for \( R_s \) can be used in Equation 3.84 to gain the position estimate \( \hat{p}_s \).
Smith and Abel’s algorithm and Schau and Robinson’s approach have the same advantages and disadvantages with the only difference that the former method only provides a single solution. Note that Huang et al. developed in [111] a mathematically equivalent algorithm called one-step least-squares (OSLS), which is computationally more efficient ($O(N)$) compared to $O(N^3)$). For implementations the OSLS approach should therefore be preferred over Smith and Abel’s solution, especially for larger numbers of microphones. The basic idea of both algorithms is however the same.

### 3.3.3.3 Friedlander’s algorithm

The methods introduced by Schau and Robinson (SX) and by Smith and Abel (SI) solve the localization problem by first estimating $R_s$ and then calculating $p_s$ via a least-squares approach. The algorithm presented in [112] takes a radically different approach: Friedlander eliminates the $R_s$ term completely. The derivation starts with Equation 3.75 which can be rewritten in matrix-vector form as

$$Mp_s = \mu - D_1d$$  \hspace{1cm} (3.88)

with

$$M = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\ \vdots & \vdots & \vdots \\ x_i - x_1 & y_i - y_1 & z_i - z_1 \end{bmatrix}, \quad \mu = \frac{1}{2} \begin{bmatrix} R_2^2 - R_1^2 - d_{21}^2 \\ R_3^2 - R_1^2 - d_{31}^2 \\ \vdots \\ R_i^2 - R_1^2 - d_{i1}^2 \end{bmatrix}, \quad d = \begin{bmatrix} d_{21} \\ d_{31} \\ \vdots \\ d_{i1} \end{bmatrix}.$$ 

Note that Friedlander does not move the reference sensor into the origin, and therefore $D_1 \neq R_s$. To eliminate the $D_1$ term, Friedlander proposes to multiply Equation 3.88 by a matrix $\zeta$ which has the measurement vector $d$ in its null-space, and therefore

$$\zeta d = 0$$  \hspace{1cm} (3.89)

Multiplying Equation 3.88 by $\zeta$ results in

$$\zeta M p_s = \zeta \mu$$  \hspace{1cm} (3.90)

which can be solved by the least squares calculation

$$p_s = (M^T \zeta^T \zeta M)^{-1} M^T \zeta^T \zeta \mu.$$  \hspace{1cm} (3.91)

Friedlander builds the matrix $\zeta$ as follows:

$$\zeta = (I - Z)\varnothing$$  \hspace{1cm} (3.92)
3.3. Review of passive TDOA-based position estimators for $\mathbb{R}^3$

with

$$\mathcal{D} = [\text{diag}(d)]^{-1} = \begin{bmatrix} d_{2,1} & 0 & \cdots & 0 \\
0 & d_{3,1} & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & d_{N,1} & 0 \end{bmatrix}^{-1}$$

$$Z = \begin{bmatrix} 0 & 1 & 0 \\
0 & 1 & \ddots \\
\vdots & \ddots & \ddots \\
1 & \cdots & 0 \end{bmatrix},$$

where $\mathcal{D}$, $I$, and $Z$ have all the size $(N - 1) \times (N - 1)$, and $Z$ is a circular shift matrix. $\zeta$ is a singular matrix of rank $(N - 2)$. Therefore, one has to use $N > 4$ to ensure that $\zeta M$ is non-singular.

Friedlander’s algorithm has the advantage that it produces only one position estimate. In addition, it is not necessary to set the origin of the coordinate system to a specific point to avoid coordinate transformations before and after the execution of the method. On the negative side, at least five sensors are necessary to provide a meaningful solution in $\mathbb{R}^3$. Like the other approaches based on the spherical model, Friedlander’s algorithm is also not suitable for planar microphone arrays.

### 3.3.3.4 Chan and Ho’s algorithm

Chan and Ho propose in [95] a two-staged approach: They first compute initial estimates for $D_1$ and $p_s$. In a second step, the first estimates are improved by compensating for the linearization error introduced in the first step. Chan and Ho estimate in their algorithm $D_1$ and $p_s$ in a single step by introducing the vector

$$\omega = [p_s^T \ D_1]^T. \quad (3.93)$$

Using Equation 3.75, one can formulate

$$e = \hat{\mu} - \hat{M} \omega \quad (3.94)$$

with

$$\hat{\mu} = \frac{1}{2} \begin{bmatrix} d_{21}^2 - R_2^2 + R_1^2 \\
d_{31}^2 - R_3^2 + R_1^2 \\
\vdots \\
d_{N1}^2 - R_N^2 + R_1^2 \end{bmatrix},$$

$$\hat{M} = - \begin{bmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 & \hat{d}_{21} \\
x_3 - x_1 & y_3 - y_1 & z_3 - z_1 & \hat{d}_{31} \\
\vdots & \vdots & \vdots & \vdots \\
x_N - x_1 & y_N - y_1 & z_N - z_1 & \hat{d}_{N1} \end{bmatrix},$$

where $e$ stands for the measurement errors introduced by the measured quantities $\hat{d}_{i1}$. Assuming no dependency between $p_s$ and $D_1$, Equation 3.94 can be solved via the least
squares calculation

\[ \omega' = (\hat{M}^T \hat{M})^{-1} \hat{M}^T \hat{\mu} \]  
\[ = \hat{M}^T \hat{\mu}. \]  

Due to the wrong assumption about the independence of \( p_s \) and \( D_1 \), the estimate

\[ \omega' = [x' \ y' \ z' \ D_1']^T \]  

has a linearization error attached to its elements. This can be mathematically formulated as

\[ x = x' + \epsilon_{x'} \]  
\[ y = y' + \epsilon_{y'} \]  
\[ z = z' + \epsilon_{z'} \]  
\[ D_1 = D_1' + \epsilon_{D_1}. \]

Chan and Ho propose to find a final position estimate which minimizes these errors, while imposing the relationship specified by the squared version of Equation 3.4

\[ D_2^2 = (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 \]  

onto a second least squares calculation. This can be achieved by subtracting \( x_1, y_1, \) and \( z_1 \) from Equations 3.98, 3.99, and 3.100 respectively, and then squaring the resulting equations including Equation 3.101. This leads to following least squares calculation:

\[ e_{\omega'} = \alpha - M \beta \]

with

\[ \alpha = \begin{bmatrix} (x' - x_1)^2 \\ (y' - y_1)^2 \\ (z' - z_1)^2 \\ D_1^2 \end{bmatrix}, \quad M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \beta = \begin{bmatrix} (x - x_1)^2 \\ (y - y_1)^2 \\ (z - z_1)^2 \end{bmatrix}. \]

This problem can be solved via

\[ \beta = M^T \alpha. \]  

The final position estimates can be calculated by

\[ p_s = \pm \sqrt{\beta} + \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}. \]

A major problem of Chan and Ho’s algorithm is that in the Euclidean 3-space Equation 3.105 can result in a set of up to eight possible position estimates from which the right one has to be chosen. On the positive side, there is no obligation to choose the
origin of the coordinate system to be the position of the reference sensor to avoid coordinate transformations before and after the execution of the algorithm. In addition, the approach compensates for the linearization error introduced during the first step of the method. Other advantages are that both $D_1$ and $p_s$ are estimated in a single step, and that only a minimum of four sensors is necessary to find a solution in $\mathbb{R}^3$. However, the method does not support planar microphone arrays.

### 3.3.4 Approaches using the cone model

The cone model simplifies the hyperbolic model by approximating the hyperboloid of two sheets created by the microphone pair $(i, j)$ and its corresponding RD $d_{ij}$ with a double infinite cone with the apex $p_x$ at the midpoint between $p_i$ and $p_j$ (see Figure 3.10, left).

![Figure 3.10: The cone model approximates the hyperboloids of two sheets of the hyperbolic model with a double infinite cone (left) to simplify calculations. However, this introduces errors (right).](image)

The approximation leads to an intrinsic error in the cone model (see Figure 3.10, right). However, the error gets insignificant for large ratios of

$$\frac{\|p_s - p_x\|^2}{\|p_i - p_j\|^2}$$

### 3.3.4.1 The algorithm of Brandstein et al.: Linear intersection (LX)

Brandstein et al. introduce in [113] a solution for the hyperbolic localization problem based on the cone model. The basic idea of the approach is to use subsets of four microphones to create individual small sensor arrays. Each sensor array has the microphones positioned on the midpoints of a rectangle. The sensors of each individual microphone array are therefore coplanar. The baselines of the sensor pairs $(i, j)$ and $(k, l)$ of each microphone array are perpendicular to each other, mutually bisecting, and define the $x$ and $y$ axis of a local Cartesian coordinate system (see Figure 3.11).

With this setup, the cones defined by the two sensor pairs and their corresponding RDs intersect in a bearing line, if only values $z \geq 0$ are allowed. The sound source will be
located in close proximity to the line, if the ratio described in Equation 3.106 is sufficiently large.

![Figure 3.11: An individual sensor array as described by Brandstein et al. and its resulting bearing line.](image)

The direction angles $\alpha$ and $\beta$ of the bearing line, which specify the direction angles relative to the $x$ and $y$ axis respectively, can be calculated via

$$
\phi_{ab} = \cos^{-1}\left(\frac{d_{ab}}{\|p_a - p_b\|_2}\right), \quad (a, b) \in \{(i, j), (k, l)\}.
$$

(3.107)

The missing direction angle $\gamma$ can then be gained via

$$
\cos^2 \alpha + \cos^2 \beta + \cos^2 \gamma = 1, \quad \text{with} \quad 0 \leq \gamma \leq \frac{\pi}{2}.
$$

(3.108)

In terms of the global coordinate system, the bearing line of the $p$-th sensor array can be expressed as

$$
l_p = r_p a_p + m_p, \quad \text{with} \quad p = 1, 2, \ldots, M,
$$

(3.109)

where $M$ is the number of sensor arrays, $m_p$ depicts the origin of the $p$-th sensor array in terms of the global coordinate system, $r_p$ represents the distance of a point on the line from the origin of the $p$-th sensor array, and $a_p$ is the rotated version of the direct cosine vector

$$
a_p' = \begin{bmatrix}
\cos \alpha_p \\
\cos \beta_p \\
\cos \gamma_p
\end{bmatrix}.
$$

(3.110)

Brandstein et al. propose to calculate the closest points of intersection $l_{pq}$ and $l_{qp}$ for each available pair of bearing lines $l_p, l_q$. This can be achieved by first calculating the ranges $r_p$ and $r_q$ for the closest points of intersection, and then using these values in Equation 3.109 to gain the points $l_{pq}$ and $l_{qp}$. The ranges $r_p$ and $r_q$ can be calculated via
the over-constrained equation
\[ r_p a_p - r_q a_q = m_q - m_p - \psi(a_p \times a_q) \] (3.111)

where
\[ \psi = \frac{\| (a_p \times a_q) (m_p - m_q) \|_2}{\| a_p \times a_q \|_2} \]

is depicting the shortest distance between the lines \( l_p \) and \( l_q \). In a last step, the position estimate for the acoustic sound source is gained by calculating the weighted average of all available \( L \) closest points of intersection:
\[ p_s = \frac{\sum_{p=1}^{L} \sum_{q=1, p \neq q}^{L} W_{pq} l_{pq}}{\sum_{p=1}^{L} \sum_{q=1, p \neq q}^{L} W_{pq}}. \] (3.112)

The weights \( W_{pq} \) can be calculated via
\[ W_{pq} = \prod_{n=1}^{2M} P(\tau_{pqn}, \tau_n, \sigma^2_{\tau_n}), \] (3.113)

where
- \( \tau_{pqn} \) is the TDOA value measured by the sensor pair \( n \) if the source position would be at \( l_{pq} \)
- \( \tau_n \) is the TDOA value measured by the microphone pair \( n \)
- \( \sigma^2_{\tau_n} \) is the variance estimate associated with \( \tau_n \)

and
\[ P(\tau_{pqn}, \tau_n, \sigma^2_{\tau_n}) = \frac{1}{\sigma_{\tau_n} \sqrt{2\pi}} e^{-\frac{(\tau_{pqn} - \tau_n)^2}{2\sigma^2_{\tau_n}}}. \]

The weights \( W_{pq} \) can only be calculated if the time delay estimation process delivers an estimate for \( \sigma_{\tau_n} \). If this information is not available, all weights should be set equal. Then \( p_s \) is the mean of all available closest points of intersection.

Brandstein’s method has the advantage that it can produce 3D position estimates based on a 2D sensor array geometry. Another positive point is that the approach only delivers one position estimate. A disadvantage of the method is that at least six sensors have to be available to calculate a position estimate (two sensor arrays with four microphones each, two sensors are shared). In addition, the algorithm can only make use of two TDOAs per sensor array. Brandstein et al. discuss in their paper the advantage that the time delay estimation can be performed locally for each sensor array. While this is true, one still has to keep in mind that if the data acquisition processes of the microphone arrays are not synchronized, the resulting bearing lines will have different timestamps attached to them, which will introduce an additional error based on the magnitude of this time interval.
3.3.5 Summary

The above introduced algorithms can be seen as the most representative and state-of-the-art algorithms for their respective model. However, there are many other algorithms available. Recent developments focus strongly on the spherical model, especially on ways how to refine the initial estimate by reducing the introduced linearization error. Chan and Ho firstly suggested to utilize the quadratic constraint

\[ x^2 + y^2 + z^2 - D_1^2 = 0 \]

to further increase the quality of the position estimate. The spherical model can then be understood as an optimization problem with a quadratic constraint. Such problems can be solved by using Lagrange multipliers. This strategy is used by the linear-correction least-squares method [114], the constrained weighted least squares approach [115], and the separated constrained weighted least squares [94] algorithm. However, to find the Lagrange multiplier these methods rely on iterative numerical methods (e.g. secant method) and cannot be considered closed-form approaches anymore.

The methods based on the spherical model can be considered as reference algorithms due to their high localization accuracy. This can be traced back to the inherent advantage of the model, in which errors in the TDEs have not such a negative impact on the position estimate as if a method using the hyperbolic model would be used. Furthermore, only the positions of two sensors and their respective TDOA is used to form a characteristic equation for the resulting least squares calculation. Approaches based on the linear model on the other hand need the position of three microphones and two TDEs. This increases not only the information which is present in each basic equation, but possibly also the amount of measurement errors. Another advantage of methods using the spherical model is that most of them compensate for the introduced linearization error. All these factors lead to a high localization accuracy. Therefore, newly developed approaches have to be measured against these benchmark methods (see Section 3.6). On the negative side, all introduced algorithms using the spherical model cannot be used with planar microphone arrays due to a singular coefficient matrix. This problem is based on the fact that the coefficient matrices of the approaches are highly dependent on the positions or the difference in positions of the sensors.

A model-independent strategy which can be used to overcome the problem of some approaches of not being able to use additional TDOAs, was introduced in [116] by Abel. The divide and conquer (DAC) based idea calculates ML estimates of subsets of all available TDOA measurements, and then combines these to a single position estimate for the overall system. The approach can be used with any closed-form ML estimator to compute the sub-estimates (e.g. for an implementation with Fang’s algorithm see [106]). The method can achieve optimum performance (unbiased, CRLB variance), yet only in case the Fisher information is sufficiently large (small-error assumption). In order for the approach to work, one has to ensure that each subset does indeed provide exactly one position estimate. That makes it necessary to prefilter the subsets in order to detect cases where no or multiple position estimates are found [106]. This can lead to quite complex implementations and high computational load, especially for an increasing number of microphones.
### 3.3. Review of passive TDOA-based position estimators for $\mathbb{R}^3$

**Table 3.2: Pros and cons of the underlying models.**

<table>
<thead>
<tr>
<th>Model</th>
<th>Advantages (+) and disadvantages (-)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyperbolic</td>
<td>- Small errors in the eccentricity of one of the hyperboloids used for the position estimation can result in large errors for the position estimate.</td>
</tr>
</tbody>
</table>
| Linear  | - The location of three sensors and two of their corresponding RDs are necessary to determine one plane. In comparison, the other models only need the positions of a microphone pair and the corresponding RD to form a characteristic equation.  
+ Natural linearization of the nonlinear problem without the introduction of errors. |
| Spherical | - Planar microphone arrays are not suitable for the model.  
- Not all available TDOA measurements can be used (only $I = I_{LI}$ is possible). A reference sensor has to be chosen.  
- Optimal sensor array geometries for $I = I_{LI}$ are hard to realize.  
+ Small errors in the eccentricity of the spheres does not introduce large errors in the position estimate |
| Cone    | - The cone model approximates the hyperboloids used by the hyperbolic model, which results in increasing localization errors with decreasing distance of the source to the center of the microphone array. |

If a localization algorithm produces more than one position estimate, a strategy to choose the right candidate has to be implemented. Usually, this procedure is not discussed in the localization papers themselves. For approaches using the spherical model which first calculate an initial estimate and then refine this estimate, the strategy can be to select the candidate which has the same sign than the initial estimate. Alternatively a second closed-form algorithm can be employed to gain an initial estimate of the source location. The correct candidate can then be chosen according to the shortest distance to this estimate. However, such a proceeding increases the computational load significantly and is only advisable if the actual localization procedure provides position estimates with extremely high accuracy. Otherwise, the first position estimate could also be simply refined by an iterative approach (e.g. Foy’s algorithm). The strategy of choosing the right position estimate is also dependent on the application: For example in case of a speaker localization task in a meeting environment, the microphone array might be put into a corner of the room and the selection process can be simplified by rejecting all possible solutions which are not in the desired quadrant. Having to implement a procedure to pick the right candidate of the possible position estimates always increases the computational load. That gives localization schemes providing only one solution the edge over approaches delivering multiple solutions.

Planar microphone arrays are suited best for the helicopter localization task. Unfortunately, all approaches based on the spherical model cannot be used with coplanar sensors. While Foy’s algorithm can cope with this condition, it needs an initial position estimate which can then be refined. A two-step approach using either the LX method, Fang’s approach or Bucher and Misra’s algorithm to calculate the initial estimate are thinkable.
Table 3.3: Pros and cons of the presented TDOA-based localization algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Model</th>
<th>Minimum amount of TDOAs ($\text{R}^3$)</th>
<th>Coordinate transformation necessary</th>
<th>Planar sensor arrays</th>
<th>Maximum number of position estimates</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foy (Taylor-series)</td>
<td>hyperbolic</td>
<td>3 LI</td>
<td>No</td>
<td>Yes</td>
<td>1</td>
<td>• Iterative</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Convergence not guaranteed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Needs initial guess</td>
</tr>
<tr>
<td>Schmidt (Plane Intersection, PX)</td>
<td>linear</td>
<td>4 LI</td>
<td>No</td>
<td>No</td>
<td>1</td>
<td>• Correction step (TDOA averaging) for $\hat{\text{r}} = \hat{\text{r}}_0$</td>
</tr>
<tr>
<td>Fang</td>
<td>linear</td>
<td>3 LI</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
<td>• Optimum in the ML sense</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Only for $N = 4$ and 3 LI TDOAs</td>
</tr>
<tr>
<td>Bucher and Misra</td>
<td>linear</td>
<td>3 LI+1 LD</td>
<td>No</td>
<td>Yes</td>
<td>2</td>
<td>• Only for $N = 4$ and 4 TDOAs</td>
</tr>
<tr>
<td>Schau and Robinson</td>
<td>spherical</td>
<td>3 LI</td>
<td>Yes</td>
<td>No</td>
<td>2</td>
<td>• Only possible for $\hat{\text{z}} = \hat{\text{z}}_L$</td>
</tr>
<tr>
<td>Smith and Abel (Spherical Interpolation, SI)</td>
<td>spherical</td>
<td>3 LI</td>
<td>Yes</td>
<td>No</td>
<td>1</td>
<td>• Only possible for $\hat{\text{z}} = \hat{\text{z}}_L$</td>
</tr>
<tr>
<td>Friedlander</td>
<td>spherical</td>
<td>4 LI</td>
<td>No</td>
<td>No</td>
<td>1</td>
<td>• Only possible for $\hat{\text{z}} = \hat{\text{z}}_L$</td>
</tr>
<tr>
<td>Chan and Ho</td>
<td>spherical</td>
<td>3 LI</td>
<td>No</td>
<td>No</td>
<td>8</td>
<td>• High amount of sensors necessary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• No arbitrary amount of microphones possible</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Only two of the three available LI TDOAs are used for each sub-array</td>
</tr>
<tr>
<td>Brandstein et al. (Linear intersection, LX)</td>
<td>cone</td>
<td>4 LI</td>
<td>No</td>
<td>Yes</td>
<td>1</td>
<td>• High amount of sensors necessary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• No arbitrary amount of microphones possible</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Only two of the three available LI TDOAs are used for each sub-array</td>
</tr>
</tbody>
</table>
3.4 New TDOA-based position estimators

However, all of these methods either work only with a limited amount of TDOAs (Fang’s approach, Bucher Misra’s algorithm) or impose a certain sensor array layout (LX method of Brandstein et al.). Fang’s algorithm could be used with the above introduced DAC method, however this would increase the complexity and the computational costs of the overall approach. Solely the LX approach of Brandstein et al. can cope with additional sensors as well as planar microphone arrays. However, the approach does not allow a random number of microphones to be used for the sensor array and restricts the geometry of each sub-array to a rectangle.

For the helicopter localization, one would ideally like to use an algorithm which

- can cope with planar microphone arrays,
- works with an arbitrary number of sensors (equal or more than the minimum of four sensors necessary to produce a position estimate in $\mathbb{R}^3$),
- can be used as an input for iterative methods for further refinement (therefore has to produce position estimates close the the actual position of the acoustic source),
- has a predictable computation time allowing it to be used in real-time systems,
- can make use of both $I = I_{LL}$ and $I = I_0$,
- has low computational demands,
- works well in the near as well as in the far-field of the microphone array,
- and does not introduce unnecessary errors due to model approximations.

3.4 New TDOA-based position estimators

Of all previously introduced methods, the most promising algorithms for the helicopter localization task are based on either the cone model or the linear model. The cone model approximates the nonlinear problem, which leads to errors in case the acoustic source is close to the microphone array. With regard to the UAV localization task in which the microphone array is embedded into the landing pad, this particular problem has the biggest effect when the helicopter is close to touchdown and one wants to have the highest localization accuracy possible.

Figure 3.12: For planar microphone arrays, the intersection of the position estimation planes results in a line perpendicular to the $x$-$y$ plane, if the sensor plane equals the $x$-$y$ plane of the Euclidean coordinate system in use.
All approaches of the linear model employ a method to calculate position estimation planes. The general formulation of this intermediate step has been described by Schmidt. However, his algorithm cannot cope with planar microphone arrays due to a singular coefficient matrix. Examining this problem closer, one can see that for coplanar sensors, the resulting position estimation planes are orthogonal to the sensor plane. They therefore intersect in a line and not in a single point anymore. This fact can be exploited if the $x$-$y$ plane of the Cartesian coordinate system in use coincides with the sensor plane. If that is the case, the resulting line is parallel to the $z$ axis (see Figure 3.12). The $x$ and $y$ coordinates of the acoustic source position are then defined by the line, and the $z$ coordinate has to be gained in a separate step.

### 3.4.1 XY coordinates

#### 3.4.1.1 Method XY-I

The introduced idea can be described mathematically by rewriting Schmidt’s formula of the position estimation planes as follows:

$$\alpha_{ijk}x + \beta_{ijk}y = \delta_{ijk}$$  \hspace{1cm} (3.114)

where

$$\begin{align*}
\alpha_{ijk} &= x_idx_jd_k + x_jd_idx_k + x_kd_ji \\
\beta_{ijk} &= y_idx_jd_k + y_jd_idx_k + y_kd_ji
\end{align*}$$  \hspace{1cm} (3.115) and

$$\delta_{ijk} = \frac{1}{2}(d_{ji}d_{kj}d_{ik} + R_i^2d_{kj} + R_j^2d_{ik} + R_k^2d_{ji})$$  \hspace{1cm} (3.117)

For $\mathcal{K} = \mathcal{K}_0$, these position estimation planes are perpendicular to the sensor plane and can be intersected in a least squares manner by

$$M \cdot p_s' = d$$  \hspace{1cm} (3.118)

with

$$M = \begin{bmatrix} \alpha_{ijk} & \beta_{ijk} \\ \vdots & \vdots \end{bmatrix}, \quad p_s' = \begin{bmatrix} x \\ y \end{bmatrix}, \quad d = \begin{bmatrix} \delta_{ijk} \\ \vdots \end{bmatrix}, \quad (i, j, k) \in \mathcal{K}.$$

There is still a sensor triplet $(i, j, k)$ necessary to form one characteristic equation for the least squares problem. Yet only the $x$ and $y$ coordinate of the acoustic source is being calculated via Equation 3.118. Similar to Schmidt’s approach, Equations 3.115 to 3.117 have to be reformulated to accommodate $\mathcal{K} = \mathcal{K}_{L1}$:

$$\begin{align*}
\alpha_{12k} &= x_1(-d_1k - d_{21}) + x_2d_{1k} + x_kd_{21} \\
\beta_{12k} &= y_1(-d_1k - d_{21}) + y_2d_{1k} + y_kd_{21}
\end{align*}$$  \hspace{1cm} (3.119) and (3.120)
and
\[ \delta_{12k} = \frac{1}{2}(d_{21} - d_{1k} - d_{21})d_{1k} + R_1^2(-d_{1k} - d_{21}) + R_2^2d_{1k} + R_3^2d_{21}. \] (3.121)

### 3.4.1.2 Method XY-II

Instead of using the least squares approach described above, one can intersect the position estimation planes defined by Equation 3.114 directly. With errors present in the system, the intersections of the position estimation planes do not result in one common line anymore. Instead multiple lines are scattered around the proximity of the actual line. The position estimate \( p'_s \) can be calculated by taking the mean value of all intermediate position estimates.

This method can expected to be less accurate than XY-I in which \( p'_s \) is gained via a least squares calculation. On the other hand, one gains access to the intermediate position estimates.

### 3.4.1.3 Comparison of method XY-I and XY-II

A Monte Carlo simulation is used to compare the accuracy of the position estimates calculated by the two methods. The noise present in the RD estimates is assumed to be zero-mean and Gaussian with \( \sigma_{RD} = 0.001 \text{ m} \). However, the maximum noise has been limited to the interval \([-D_{ij}, D_{ij}]\) to reflect the maximum possible errors which can be gained in a real-world time delay estimation process. The same noise values are applied to both approaches equally. The microphone array which is used for the simulations is shown in Figure 3.13.

The root-mean-square error (RMSE) of the calculated position estimates is used as a measure for the accuracy of the position estimates. For an unbiased estimator, the RMSE equals the standard deviation of the position estimates. However, if the estimator is biased, the RMSE incorporates both the bias as well as the bias-corrected variance of the
position estimates. For the remainder of this thesis the RMSE will be used to compare the accuracy of the various position estimators. The bias and variance will only be discussed separately if they exhibit special properties. The RMSE is calculated via

\[
rmse_{xyz} = \sqrt{\frac{1}{P} \sum_{p=1}^{P} (\|\hat{p}_s - p_s\|_2)^2}.
\]  

(3.122)

For \( I = I_{11} \) and \( N = 4 \), both methods produce identical results (there is maximum one position estimation plane intersection available). Figure 3.14 therefore shows the RMSE for both methods with \( I = I_{11} \) and \( N = 6 \). For each possible source location \( p_s = [x \ y \ 1]^T \) m with \( x, y \in [-15, 15] \) m and \( \Delta_x = \Delta_y = 0.25 \) m, 100 trials have been calculated. One can see significant differences in the location accuracy between the two approaches. This can be traced back to the fact that method XY-I is similar to Schmidt’s approach a weighted least squares (see discussion in [103, 104] for Schmidt’s algorithm) while method XY-II equally weights all position estimation plane intersections. This leads to large errors especially when multiple position estimation planes are close to being parallel, leading to intersection points far outside the working area.

![Figure 3.14: Comparison of the results of the methods XY-I (left) and XY-II (right) for the source locations](image)

**3.4.2 Improving method XY-II**

**3.4.2.1 Median**

To improve the final position estimate, the median of all available intermediate position estimates \( p_{si} \) can be used instead of their mean value. This makes the approach more resistant against outliers. This strategy however comes with the price of more computation due to the necessity to sort the results. Nevertheless, an upper time limit can be calculated depending on \( N \) and the sort algorithm used. This change does therefore not affect the real-time capabilities of the approach.

**3.4.2.2 Outlier rejection**

For acoustic-based localization schemes, the range in which a localization is possible is limited by the strength of the source signal in comparison to the noise present in the system, usually referred to as the signal-to-noise ratio (SNR). Due to damping of the acoustic signal with increasing distance of the source from the sensor array, and additional effects of the propagation medium onto the strength of the signal received at the microphones.
3.4. New TDOA-based position estimators

(e.g. damping effects due to humidity and temperature based on the frequency of the signal), the working range of atmospheric acoustic localization approaches is limited. If the radius of the catchment area of the acoustic localization is known or can be estimated, all intersection points with \(x\) and \(y\) coordinates outside of this area can be treated as outliers. However, one should increase the expected radius slightly for the outlier rejection process to allow a good position estimate to be calculated at the end of the possible range for the acoustic localization. This strategy is an advantage of method XY-II over XY-I which does not provide any intermediate results which could be checked for outliers. The procedure will have the most impact on position estimates at the edge of the working area. However, it can only be used if the range of the working area is known or can be estimated.

3.4.2.3 Weighting

The smaller the angle between two intersecting position estimation planes, the higher are the negative effects of measurement errors onto the resulting \(p_{si}\): Small errors in the direction of almost parallel position estimation planes can lead to intersection points far away of the intersection position in the error-free case.

It has been proposed earlier to use the unweighted mean or the median of the coordinates of all intermediate position estimates to gain the final estimate for the position of the acoustic source. However, with the knowledge that the angle between the intersecting planes gives an indication about the quality of each intermediate position estimate, one can use the weighted mean of the available \(p_{si}\) to gain an improved final estimate for the location of the source:

\[
p_{s} = \frac{1}{W} \sum_{p=1}^{P} w_{p} p_{si_{p}}
\]  

(3.123)

with

\[
w_{p} = \arccos \left( \frac{a_{p}a_{q} + \beta_{p}\beta_{q}}{\sqrt{a_{p}^{2} + \beta_{p}^{2}}\sqrt{a_{q}^{2} + \beta_{q}^{2}}} \right)
\]  

(3.124)

and

\[
W = \sum_{p=1}^{P} w_{p}.
\]  

(3.125)

The weighting favors these \(p_{si}\) which result from the intersection of position estimation planes defined by pairs of microphones which have an advantageous layout to each other with respect to the current source position. In case the microphone pairs produce the same position estimation planes, \(w_{p}\) will be zero while in case the microphone pairs define planes which are perpendicular to each other, \(w_{p}\) will reach its maximum \(\pi/2\). For almost identical position estimation planes, even small measurement noise can lead to a large error, while for planes intersecting with an angle around \(\pi/2\), a considerable amount of noise has to be present to produce even moderate errors in the intermediate position estimate.
Chapter 3. An acoustic localization approach for precise landings on a helipad

The introduced weighting is closely related to the CRLB. The coefficients of the matrix \( G \) of the CRLB are dependent on the difference in direction between two sensors towards the acoustic source position. In comparison, the weights represent the difference in direction of two position estimation planes defined by two pairs of three microphones which point to the current source position. A big advantage is that the weighting can be done online without any prior knowledge about the approximate position of the source.

3.4.2.4 Comparison

A Monte Carlo simulation using the same sensor array and parameters than previously used in Section 3.4.1.3 is used to show the effects of the improvements onto the final position estimates. The results can be seen in Figure 3.15. The left plot shows the localization accuracy for using the median value of the coordinates of all \( p_{si} \) instead of their mean value. One can see that the RMSE is significantly lower than the one of the original XY-II method. That means that outliers have a big negative influence onto the quality of the position estimates of the original method. However, the effect of these outliers onto the median value are minor which leads to an overall reduced RMSE.

Figure 3.15: RMSE of method XY-II using the median value instead of the mean value of all coordinates of the intermediate intersection points (left), with outlier rejection with range \( \sqrt{20^2 + 20^2} \) m (center) and with weighting (right). The source locations are \( p_S = [x' y' 1]^T \) m with \( x', y' \in [-15, 15] \) m and \( \Delta_x = \Delta_y = 0.25 \) m with \( \sigma_{RD} = 0.001 \) m and \( \mathcal{I} = \mathcal{I}_{LI} \).

The plot in the center of Figure 3.15 shows the improvements gained by rejecting all intermediate position estimates which are located outside the expected area. In the simulation, all position estimates which are further away than \( \sqrt{20^2 + 20^2} \) m from the origin of the sensor array have been rejected. The value has been chosen to still allow good position estimates to be calculated in the corners of the plots. Again, a much lower overall RMSE than the original method can be seen. This is backing the explanation that a small amount of outliers has a big negative influence onto the original method. Comparing the results of the median approach and the outlier rejection, one can see that the median version of the localization routine produces a more uniform RMSE with high errors at increased distance from the center of the sensor array. In contrast, the outlier rejection shows high errors along the extended baselines of the microphone array. This means that along these lines the localization procedure produces outliers which have an error which is not significant enough for the outlier rejection to filter them out. However, these high RMSE position estimates are still outliers, which is why the median approach filters these out quite well.

The left plot of Figure 3.15 depicts the improvements gained by using weighting. The quality of the position estimates exceeds the ones gained from the median and outlier rejection approaches. The results show that the outliers are indeed tied to the intersec-
3.4. New TDOA-based position estimators

...tion of almost parallel position estimation planes. While the outlier rejection removes the intermediate position estimates from the further calculation, the weighting makes their influence onto the overall position estimate insignificant. The weighting has the advantage that even smaller outliers close to the center of the working area are successfully voted down and do not have any major influence onto the overall position estimate.

Figure 3.16: The difference in RMSE between method XY-I and method XY-II median and outlier rejection (left), XY-I and XY-II outlier rejection and weighting (center), and XY-I and XY-II outlier rejection and weighting with \( I = I_0 \) (right). Note the different scale used for the right plot. The range for the outlier rejection has been set to \( \sqrt{20^2 + 20^2} \) m. The source locations are \( p_s = [x \ y \ 1]^T \ m \) with \( x, y \in [-15, 15] \ m \) and \( \Delta x = \Delta y = 0.25 \ m \).

Figure 3.16 shows the performance of method XY-II with variations of the improvements in comparison to the results of method XY-I. Blue areas show where method XY-I is superior over method XY-II with its respective improvement variation, red areas show where it is the other way around. For \( J = J_{II} \) (left and center) method XY-II is no match for method XY-I even with weighting and outlier rejection (center). This changes for \( J = J_0 \): The results are almost identical, however method XY-II with weighting and outlier rejection performs better with increased distance and around the corners of the working area. Towards the center, method XY-I has a slight advantage which gets negligible if the source position gets close to the center of the sensor array.

3.4.3 Z coordinate

3.4.3.1 Method Z-I

The z coordinate for \( p_s' \) has to be calculated in a second step. This can be achieved by utilizing Equation 3.57 of Fang’s method. However, one has to reformulate the coefficients due to Fang’s stricter assumptions about the coordinate system in use (one point equals the origin, the second point is located on the x axis). First, the x coordinate of \( p_s \) gained from the sensor triplet \((i, j, k)\) is needed in \( cs_{fang} \):

\[
x_{fang} = x \cos(\phi) - y \sin(\phi) \quad \text{where} \quad \phi = \text{atan2}(y_j, x_j).
\]

(3.126)

Then the coefficients \( a, \beta, \gamma, \delta \) and \( \eta \) of Fang’s approach can be calculated with the parameters

\[
x_{i,fang} = \|p_j - p_i\|_2 \quad \text{(3.127)}
\]

\[
x_{k,fang} = \frac{(p_j - p_i) \cdot (p_k - p_i)}{\|p_j - p_i\|_2} \quad \text{(3.128)}
\]

\[
y_{k,fang} = \sqrt{\|p_k - p_i\|_2^2 - x_{k,fang}^2} \quad \text{(3.129)}
\]
Chapter 3. An acoustic localization approach for precise landings on a helipad

With Fang also assuming the sensor plane being the $x$-$y$ plane of the Cartesian coordinate system in use, a rotation and translation of the resulting value for the $z$ coordinate is not necessary. A $z$ coordinate can be calculated from each sensor triplet $(i, j, k)$ forming a position estimation plane. The median of all intermediate $z$ values gives the estimate for the $z$ coordinate of the acoustic source.

### 3.4.3.2 Method Z-II

![Figure 3.17: Alternative approach to gain the $z$ coordinate.](image)

An alternative approach to calculate the $z$ value can be seen in Figure 3.17. Using $p'_s$ and the Pythagoras theorem, one can formulate

$$z^2 + a^2 = D_i^2$$  \hspace{1cm} (3.130)
$$z^2 + b^2 = (D_i + d_{ji})^2.$$  \hspace{1cm} (3.131)

Solving Equations 3.130 and 3.131 for $z^2$, equalizing the resulting equations and using $d_{ji} = -d_{ij}$ yields

$$D_i = \frac{b^2 - a^2 - d_{ij}^2}{-2d_{ij}}.$$  \hspace{1cm} (3.132)

Substituting $D_i$ into Equation 3.130 results in

$$z = \left[ \left( \frac{b^2 - a^2 - d_{ij}^2}{-2d_{ij}} \right)^2 - a^2 \right]^{\frac{1}{2}} \quad \text{with} \quad z \geq 0.$$  \hspace{1cm} (3.133)

This can be done three times for each sensor triplet $(i, j, k)$ (once for each pair $(i, j)$ $(i, k)$, $(j, k)$) forming a position estimation plane. The median of all intermediate $z$ values gives the estimate for the $z$ coordinate of the acoustic source.

### 3.4.3.3 Comparison of method Z-I and Z-II

Both method Z-I and Z-II cannot deliver a result in case $d_{ij} = 0$ due to a division by zero. However, a good sensor placement can ensure that at least one candidate for the $z$ coordinate is available at all times.
Figure 3.18 shows the results of a Monte Carlo simulation targeted to show the difference in the localization accuracy between the approaches Z-I and Z-II. The same sensor array and parameters as introduced in Section 3.4.1.3 have been used for the trial. The z coordinate estimates for both approaches have been calculated based on the result $p'_s$ provided by method XY-I with $I = I_0$.

One can see that both methods provide a uniform error distribution with increasing error towards the boundaries of the working area (see left and center plot of Figure 3.18). The difference in RMSE shows that overall method Z-II outperforms method Z-I (red areas of right plot in Figure 3.18). However method Z-I has a slight advantage in the corners of the working area. However, approach Z-II should be generally preferred over method Z-I.

The worse performance of Z-I stems from the fact that two RDs and three sensor positions are necessary to calculate a single z candidate. Therefore, the errors of two RDs influence the result. Method Z-II on the other hand allows to calculate a z coordinate estimate with just a pair of sensor positions and its corresponding RD. That leads to an overall larger number of z candidates and an outlier in one of the RDs only affects a single intermediate estimate. In comparison, a RD outlier negatively influences multiple z candidates for approach Z-I, which in addition has a lower number of z estimates. The negative effect of RD outliers onto the final z estimate are therefore smaller with method Z-II.

### 3.4.4 A matter of trust

Based on the weighting of method XY-II, a measure of trust can be introduced for this method: Comparing two position estimates to each other, the estimate with a larger $W$ (sum of all individual weights, see Equation 3.125) will be generally more reliable because it is based on the intersection of position estimation planes with a larger angle to each other, which again reduces the negative influence of noise. A low $W$ in turn means that the position estimate is based on almost identical position estimation planes and is therefore very susceptible to noise and outliers. Therefore $W$ can be used as a trust value.

The results of a 100-trial Monte Carlo simulation are used to show the suitability of the introduced trust value for making a decision if the position estimate is accurate enough to be used for the localization task at hand. The two sensor array geometries shown in Figure 3.19 in conjunction with method XY-II and Z-II as well as $I = I_0$ are used to calculate estimates of the source positions $p_s = [x \ y \ 1]^T$ with $x, y \in [-15, 15]$.
Figure 3.19: The microphone layouts used to show the characteristics of the introduced trust value: The square layout (left) and the 120° layout (right). The distances from the sensors to the origin of the sensor array are equal for both layouts. The sensor \( p_1 \) of the 120° layout coincides with the origin of its sensor array.

Figure 3.20: CRLB for the two sensor array layouts: Rectangle layout (left), 120° layout (right). The source locations are \( p_s = [x \ y \ 1]^T \) m with \( x, y \in [-15,15] \) m and \( \Delta x = \Delta y = 0.25 \) m and \( \sigma_{RD} = 0.001 \) m.

Figure 3.21: Trust values for position estimates for the rectangle layout (left) and the 120° layout (right) using method XY-II. The source locations are \( p_s = [x \ y \ 1]^T \) m with \( x, y \in [-15,15] \) m and \( \Delta x = \Delta y = 0.25 \) m. The two sensor arrays have a vastly different lower bound on the variance of the position estimates at specific points of the working area. This can be seen in the CRLB plots of Figure 3.20.

\( \sigma_{RD} = 0.001 \) m and \( \Delta x = \Delta y = 0.25 \) m. The two sensor arrays have a vastly different lower bound on the variance of the position estimates at specific points of the working area. This can be seen in the CRLB plots of Figure 3.20.

Figure 3.21 shows the trust values of method XY-II for the rectangle layout as well as for the 120° layout. One can see that the trust value closely resembles the same distribution than the inverse of the CRLB for both microphone layouts, meaning that a low trust value is attached to position estimates which have a high CRLB. The user can therefore decide by means of the trust value if he wants to use a position estimate or reject it as outlier. However, the amount of trust necessary for an application to take a position estimate into account is dependent on the application itself.
3.4. New TDOA-based position estimators

Similar to the trust value of method XY-II, a trust value can be calculated for method XY-I: The matrix \((M^T M)\) is singular in the error-free case for constellations of the sensor positions to the acoustic source which prohibit the calculation of any result. However, with errors present in the system (errors in the RDs as well as numerical errors in the calculations) a result can be calculated. In that case, the matrix \((M^T M)\) can be inverted but is ill-conditioned and features a very high condition number. One can therefore use the inverse of the condition number as trust value for method XY-I. Figure 3.22 shows the trust values gained by using this strategy for both sensor array geometries. Note that the trust value of method XY-I is unit-less while the trust value of approach XY-II has the unit degrees or radians. Based on that, the limit for the trust value specified by an application will be different between XY-I and XY-II.

3.4.5 Case study: 120° layout

In this section the performance of the introduced localization approaches is tested with respect to the helicopter localization task. The prior introduced 120° sensor array is used: \(p_1\) coincides with the origin of the coordinate system in use, and \(D_{1j} = 0.5\) m with \(j = 2, 3, 4\).

Following settings are used for the Monte Carlo simulation: \(\sigma_\tau = 1\) tick, \(f_s = 192\) kHz and therefore \(\sigma_{RD} \approx 1.79\) mm. A systematic error is assumed in the TDEs with a uniform distribution over the interval \([-0.1, 0.1]\) ticks. The positioning error of the microphones is modeled with \(\sigma_{xy} = 0.5\) mm and \(\sigma_z = 1\) mm respectively. Both method XY-I and XY-II are used in combination with method Z-II. In addition, both methods make use of Schmidt’s correction step and utilize \(I = I_0\). The approach XY-II is used with weighting and outlier rejection. The values used for \(\sigma_{xy}, \sigma_z\) and \(D_{1j}\) are the ones expected from the real-world experimental setup, while a systematic error was assumed to be small but present. The plotted values are not filtered / no flight path has been assumed.

To gain a representative number of position estimates which cover the whole working area, the acoustic source is assumed to be at \(p_s = [r_{xy} \cos \alpha \quad r_{xy} \sin \alpha \quad R_s \sin \beta]^T\) with \(\alpha \in [0; 360]^\circ\), \(\Delta \alpha = 10\) degrees and \(\beta \in [5, 10, 30, 60, 80, 85]\) degrees as well as \(r_{xy} = R_s \cos \beta\). The origin of the sensor array is also the origin of the coordinate system in use. Five iterations have been used to produce the graphs in Figure 3.23. For each iteration the faulty microphone positions as well as the systematic errors for each RD have been kept constant.
As stated in Section 3.1.1, the localization error of the position estimator is not allowed to exceed \( \pm 0.25 \) m over the helipad for the localization approach to be accurate enough to precisely land the UAV on the landing platform. Figure 3.23 shows the expected error of the introduced localization algorithms with increasing distance of the signal source towards the center of the sensor array. It can be seen that the position estimates for both approaches stay well inside the specified error margin when the UAV is located close or above the landing pad.

Even though the fourth quartile of the \( z \) estimates approach the \( 0.2 \) m mark in the interval \( R_s \in [0.1, 2.0] \) m, the median of the estimates is close \( (\leq 0.03 \) m) to the actual elevation of the UAV. Filtering the \( z \) estimates will therefore lead to an improved result. The raw \( z \) estimates of both introduced acoustic localization methods already outperform the altitude estimations of GPS and the accuracy provided by pressure sensors.

A significant advantage of both introduced approaches is that the \( z \) values are by definition always positive. Therefore, the worst case scenario is that the position estimates underestimate the distance of the UAV above ground, assuming the helicopter has already landed while it is indeed still aloft. The common landing maneuver executed with a remote controlled helicopter is to fly above the landing position and then slowly decrease the collective pitch which descents the UAV onto the helipad. Before turning off the motor, the collective pitch is negative, pushing the helicopter lightly onto the landing pad. The relatively small errors which can be expected in the altitude estimates can be compensated for by keeping the motor on for \( \Delta t_{land} \) seconds after one assumes that the
UAV landed, and before turning off the motor. That allows the helicopter to further lose altitude in case the distance above the sensor array has been slightly underestimated. Furthermore, the altitude estimates gained during this time period can be averaged to gain an accurate distance of the UAV to the landing pad allowing one to ensure that the helicopter has indeed landed.

On the opposite side, an overestimation of the helicopter altitude would lead to the assumption that the UAV is still aloft, while it has indeed already landed. With the small errors to be expected and the descend of the helicopter being performed over the center of the helipad, this would solely lead to a premature landing.

Figure 3.24 shows the errors which one can assume to encounter over the distance interval $R_s \in [0.5, 15.0]$ m. Both introduced methods perform very similar across the whole working area. The errors which can be expected in real-world applications based on the executed Monte Carlo simulations allow both methods to be used for the UAV localization task.

For distances further away from the landing pad, one can argue over the benefits of sending the position estimates to the helicopter (e.g. at a distance $\geq 5$ m where the fourth quartile exceeds an error of 1 m in the x-y direction). In this case, it is more valuable to guide the UAV towards the center of the helipad by solely providing it with the direction it has to take to reach the landing platform. The angle from the center of the microphone array towards the acoustic source is very accurate even at larger distances (see Figure 3.25).
Chapter 3. An acoustic localization approach for precise landings on a helipad

3.4.6 Summary

The introduced methods have been developed to solve the TDOA-based passive acoustic localization problem with planar sensor arrays. Compared to their direct competitors (Fang’s algorithm, Bucher and Misra’s approach, LX method of Brandstein et al.) the approaches feature increased flexibility:

- Both methods work with $J = J_{LI}$ and $J = J_0$.
- The approaches can make use of Schmidt’s correction step for $J = J_0$.
- Sensor arrays with $N \geq 4$ can be used.
- The planar sensor array is not limited to any fixed geometry.
- The approaches do not rely on an approximation of the underlying model.
- A trust value can be calculated online, allowing an application to choose whether to use the position estimate for further calculations or treat it as an outlier.
- Only the minimum amount of three LI TDOAs is necessary to find a position estimate in $\mathbb{R}^3$.
- The algorithms have a predictable computation time making them suitable for real-time systems.
- Both approaches deliver strictly one unambiguous result.

In addition to these advantages, both approaches produce in the conducted simulations position estimates accurate enough to guide the CO$_2$ monitoring UAV to the landing area and to allow an autonomous landing solely based on the delivered positions. However, both approaches also have limitations:

- The methods only work with planar microphone arrays.
- The outlier rejection of method XY-II is only possible if the radius of the working area is known or can be approximated.
- Both Z-I and Z-II do not provide a solution for $d_{ij} = 0$.
- A coordinate transformation is necessary if the sensor plane does not coincide with the $x$-$y$ plane of the coordinate system in use.

Figure 3.25: The direction error is small if the signal source is not located directly above the landing pad.
3.5 Optimal array geometry for the UAV localization task

3.5.1 Maximum base distance uniform angular arrays

UAAs (2D) and platonic solid based array geometries (3D) are CRLB optimum for $I = I_0$ (see Section 3.2.6.1). However, Equation 3.15 as well as the direction vectors $g_i$ defined by using platonic solid based array geometries do solely provide a direction where the microphones have to be positioned, but no distance. This can be traced back to the fact that the CRLB is also not dependent on the distances $D_i$. However, it has been shown in [117] that the distances between the expected source position and the sensors matters. Indeed, the distances should be made as large as possible if the expected location of the acoustic source is in the center of an UAA.

For acoustic localization, this finding goes well with the fact that noise has less effects onto the TDOA measurement $\tau_{ij}$ with an increased distance $D_{ij}$ between the receiving sensor pair $(i, j)$. The distance $D_{ij}$ is commonly referred to as base distance. The upper end of $D_{ij}$ is constrained by the time delay estimation process: To unambiguously determine the TDOA of a sound signal arriving at a pair of microphones, it is necessary that the wavelength $\lambda$ of the frequency $f$ of the acoustic signal has to be at least twice the distance between the sensors. This can be formulated mathematically as follows:

$$D_{ij} = \|p_i - p_j\|_2 \leq \frac{1}{2} \frac{\lambda}{\lambda} = \frac{c}{2f} \quad (3.134)$$

$$D_{ij,\text{max}} = \frac{c}{2f} \quad (3.135)$$

With the additional constraint of maximizing $D_{ij}$, a subset of UAAs can be defined which will be further referred to as maximum base distance uniform angular arrays (MUAAs). If all sensors are pushed away from the center of the sensor array as far as the time delay estimation process allows, all microphones will be located on a circle with the radius $r$ around the origin of the array. The sensor locations are then specified by

$$p_i = r g_i. \quad (3.136)$$

The maximum radius $r_{\text{max}}$ is constrained by the distance $D_{ij}$ between the two microphones $i$ and $j$ furthest away from each other. Assuming $I = I_0$, one can use Equations 3.15, 3.134 and 3.136 to gain $r_{\text{max}}$:

$$D_{ij,\text{max}} = \|r_{\text{max}} p_i - r_{\text{max}} p_j\|_2$$

$$= \sqrt{(r_{\text{max}} \cos \alpha_i - r_{\text{max}} \cos \alpha_j)^2 + (r_{\text{max}} \sin \alpha_i - r_{\text{max}} \sin \alpha_j)^2} \quad (3.138)$$

$$= \sqrt{2r_{\text{max}}^2 (1 - \cos \alpha_i \cos \alpha_j - \sin \alpha_i \sin \alpha_j)} \quad (3.139)$$

Using the product-to-sum formulas for sine and cosine, one can simplify Equation 3.139 to

$$D_{ij,\text{max}} = \sqrt{2r_{\text{max}}^2 (1 - \cos(\alpha_i - \alpha_j))}. \quad (3.140)$$

Squaring and using Equation 3.16 twice (with $i = 1$ and $j$) allows one to solve for $r_{\text{max}}$. 101
yielding

\[ r_{\text{max}} = \sqrt{\frac{D_{ij,\text{max}}^2}{2(1 - \cos(\frac{2\pi(1-j)}{N}))}} \]  

(3.141)

with

\[ j = \begin{cases} 
\frac{N}{2} + 1 & \text{when } N \text{ is even} \\
\frac{N + 1}{2} & \text{when } N \text{ is odd.} 
\end{cases} \]

This allows one to calculate not only the direction vectors of the microphones but also the sensor locations with Equations 3.135, 3.136 and 3.141.

3.5.2 Uniformly Cramér-Rao lower bound distributed maximum base distance uniform angular arrays

In Section 3.4.4, the square layout as well as the 120° layout have been introduced and used for the discussion of the trust values of method XY-I and XY-II (see Figure 3.19). The square layout is a MUAA for \( N = 4 \), and the 120° layout can be seen as a MUAA for \( N = 3 \) with an additional sensor in the center of the microphone array. Both sensor arrays have the same amount of sensors, and the microphones are located the same distance away from the origin. The CRLB plots for \( J = J_0 \) however show a significantly different distribution (see Figure 3.20): While the CRLB for the 120° layout is evenly distributed (higher \( \Xi \) with increasing \( R_s \)), the CRLB plot for the square layout shows large \( \Xi \) values along the \( x \) and \( y \) axis, and is at some points even not defined. Based on these CRLB distributions, one can see that CRLB optimum sensor arrays are not necessarily application optimum array geometries: The 120° layout is very beneficial for the UAV localization task, allowing accurate position estimates to be calculated independent of the direction from which the helicopter approaches. In contrast, the square layout poses challenges if the UAV approaches the sensor array along either the \( x \) or the \( y \) axis.

The CRLB is undefined if the FIM is ill-conditioned or rank-deficient. It is possible that the CRLB is not defined even though an unbiased estimator and the CRLB actually exist. To overcome this problem, it has been suggested to use the pseudo-inverse of the FIM or to use the reciprocal of the corresponding diagonal elements of the FIM [118]. Alternative formulations of the CRLB which do not require to solve any inversion have also been proposed [118]. However, as noted in [119] there is no unbiased estimator with finite variance if the FIM is singular, except in very unusual scenarios. It will be shown that the CRLB for the square microphone layout is undefined along the \( x \) and \( y \) axis because the theoretical limit on the variance is indeed infinity.

Figure 3.26 shows the condition number of the matrix \((G^T G)\) depending on the position of the acoustic source. One can see that the matrix for the square microphone layout is ill-conditioned (high condition numbers) if the acoustic source is located on either the \( x \) or \( y \) axis. This problem is not present for the 120° layout. A geometric explanation therefore can be given using Schmidt’s position estimation planes: Assuming the acoustic source somewhere on the \( x \) plane, the distances between the source to the microphone pairs \((1,2)\) and \((3,4)\) are equal \((D_1 = D_2\) and \(D_3 = D_4\)). This follows that \(d_{12} = 0\) and
3.5. Optimal array geometry for the UAV localization task

Figure 3.26: The condition numbers of \((G^T G)\) from the calculation of the CRLB for the square microphone layout (left) and the 120° layout (right). Note the problem cases \(x = 0\) and \(y = 0\) for which the condition numbers of \((G^T G)\) \(\to\) \(\infty\) and the inverse \((G^T G)^{-1}\) is undefined. The source locations shown are \(p_s = [x \ y \ 1]^T m\) with \(x, y \in [-15, 15] m\) and \(\Delta_x = \Delta_y = 0.25 m\).

\[d_{34} = 0.\] The distances from the sensors to the origin of the microphone array are equal, therefore \(R_1 = R_2 = R_3 = R_4 = r\). For the microphone triplet \((1, 2, 3)\), Equations 3.114 to 3.117 are as follows:

\[
\begin{align*}
\alpha_{123} &= x_1 d_{32} + x_2 d_{13} \quad (3.142) \\
\beta_{123} &= y_1 d_{32} + y_2 d_{13} \quad (3.143) \\
\delta_{123} &= \frac{r^2}{2} (d_{32} + d_{13}) \quad (3.144)
\end{align*}
\]

with

\[d_{32} + d_{13} = 0 \quad (3.145)\]

which leads to

\[
\begin{align*}
\alpha_{123} &= x_1 d_{32} - x_2 d_{32} = d_{32}(x_1 - x_2) \\
\beta_{123} &= y_1 d_{32} - y_2 d_{32} = d_{32}(y_1 - y_2) \\
\delta_{123} &= \frac{r^2}{2} (d_{32} - d_{32}) = 0.
\end{align*}
\]

The MUAA sensor distribution leads to \(x_1 = x_2, y_1 = -y_2\) and therefore

\[
\alpha_{123} = 0. \quad (3.149)
\]

Using Equation 3.114,

\[
\begin{align*}
\alpha_{123} x + \beta_{123} y &= \delta_{123} \quad (3.150) \\
0 x + \beta_{123} y &= 0 \quad (3.151)
\end{align*}
\]

\[y = 0. \quad (3.152)\]

Similarly, one can calculate the position estimation planes for the microphone triplets \((1, 2, 4)\) \((3, 4, 1)\) and \((3, 4, 2)\). The result will be the same position estimation plane \(y = 0\) (The same steps can be used to show that all position estimation planes are \(x = 0\) if the source position is located on the \(y\) plane). With all position estimation planes being equal, no intersection and therefore no position estimate can be calculated. Therefore, the CRLB
is indeed infinity for the square microphone layout if the source is either located on the \(x\) or \(y\) plane.

The CRLB only shows the error-free case. With errors present in the system (RD errors as well as numerical errors during the calculations), the position estimation planes will not be equal anymore and will actually intersect. A position estimate can then be calculated, but will have a high error attached to it.

Examining the problem of the unevenly distributed CRLB for MUAAs closely, one can see that non-uniform distributions always occur for sensor arrays with an even number of microphones. However, with an increasing number of sensors the problem gets less and less noticeable. This can be again explained geometrically using Schmidt’s position estimation planes: For an even number of sensors and all microphones being located on a circle around the origin to maximize the base distance, the sensors form the vertices of an axis-symmetric (\(x\) and \(y\) axis) figure. If the source is located on either the \(x\) or \(y\) plane, there will therefore be always multiple sensor triplets resulting in the same position estimation plane (either \(x = 0\) or \(y = 0\)). With more microphones, an increasing number of position estimation planes can be calculated for which the problem cases \(d_{ij} = 0\) do not coincide with the \(x\) or \(y\) plane. These planes provide additional information about the system, allowing an intersection of position estimation planes to take place and a position estimate to be calculated (The case \(N = 4\) is therefore the worst-case scenario, because all position estimation planes are the same and no additional information is available to compensate for the problem cases). However, even though a position fix can be gained, the effective number of planes (number of non-identical planes) used for the calculation will be low, leading to a solution which is prone to measurement noise and outliers.

A simple solution for sensor arrays with an even amount of sensors to gain a uniformly distributed CRLB is to put one sensor into the center of the sensor array (e.g. 120° layout). This breaks the axis-symmetry and leads to non-identical position estimation planes. The sensor positions can then be described by

\[
p_i = \begin{cases}
    p_{off} + \left[ r \cos \alpha_i \quad r \sin \alpha_i \quad 0 \right]^T & \text{when } N \text{ is even and } i = 1, 2, \ldots, N - 1 \\
    p_{off} + \left[ \frac{1}{N-1} \sum_{q=1}^{N-1} x_q \quad \frac{1}{N-1} \sum_{q=1}^{N-1} y_q \quad 0 \right]^T & \text{when } N \text{ is even and } i = N \\
    p_{off} + \left[ r \cos \alpha_i \quad r \sin \alpha_i \quad 0 \right]^T & \text{when } N \text{ is odd}
\end{cases}
\]

with

\[
\alpha_i = \alpha_0 + (i - 1) \frac{2\pi}{N} \\
p_{off} = \begin{bmatrix} x_{off} & y_{off} & 0 \end{bmatrix}^T.
\]

\(p_{off}\) is the offset of the sensor array with respect to the origin of the Euclidean coordinate system in use and \(\alpha_0\) is the angle from \(p_{off}\) to the first sensor. The resulting sensor arrays will always be at least partially MUAAs. The advantages of the MUAA class of microphone arrays is therefore still applying to this slightly modified sensor array class, which will be further referred to as uniformly Cramér-Rao lower bound distributed maximum base distance uniform angular arrays (UMUAAs). The newly introduced class of sensor arrays maximizes the base distance to gain the optimum possible results from the time
3.6. Performance analysis: Simulation

3.6.1 Direct competitors

In Section 3.4.5 it has been shown via Monte Carlo simulations using error parameters set to values which are expected in real-world experiments that both approaches XY-I and XY-II in conjunction with Z-II can be used for the localization of the CO$_2$ sensing UAV. With other methods available which can also perform this task, a direct comparison has to be done to determine which of the approaches provides the best localization accuracy for the localization task, and should therefore be used in real-world deployments.

Direct competitors to the introduced approaches are Fang’s approach, Bucher and Misra’s algorithm and the LX method introduced by Brandstein et al., which can all be used with planar microphone arrays. While the competitors to the introduced approaches feature less flexibility than the introduced methods (as has been previously discussed in Section 3.4.6), the focus of this section is solely on the localization accuracy.

Table 3.4: Approaches and their respective settings used for the direct comparison of the localization approaches.

<table>
<thead>
<tr>
<th>No</th>
<th>Method</th>
<th>Settings</th>
<th>RDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>XY-I</td>
<td>Z-II median</td>
<td>same as approach to be compared with</td>
</tr>
<tr>
<td></td>
<td></td>
<td>weighting,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>outlier rejection</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$(r = \sqrt{20^2 + 20^2} , \text{m})$,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Schmidt’s correction step,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Z-II median</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>XY-II</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$I = I_0$</td>
</tr>
<tr>
<td>3</td>
<td>Fang’s algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$I = {(i, j), (i, k), (i, l)}$</td>
</tr>
<tr>
<td>4</td>
<td>Bucher and Misra’s algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$I = {(i, j), (i, k), (k, l), (k, j)}$</td>
</tr>
<tr>
<td>5</td>
<td>Method of Brandstein et al.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$I = {(i, j), (k, l), (l, m), (j, n)}$</td>
</tr>
</tbody>
</table>

All of the direct competitors of the introduced methods require different RDs (see Table 3.4), the LX approach in addition a specific sensor array geometry. A comparison between two algorithms is only meaningful if not only the same amount of RDs is used but also exactly the same RDs. In addition to that, the sensor array has to be the same for both approaches as well as the noise put onto the RDs and the positions of the microphones. Therefore, a direct comparison between all approaches is not worthwhile. However, with the flexibility of the introduced approaches it is possible to compare each of the competitors to the introduced methods.

Method XY-I and XY-II are identical if only three LI TDOAs are used due to the fact that there is only one intersection possible between the two resulting position estimation
planes. Therefore, neither the weighting nor the median improvement does affect the position estimates of method XY-II. In addition, the outlier rejection would remove the only available position estimate if it is outside the extended expected working area. Method XY-I generally yields better position estimates for $\mathcal{J} = \mathcal{J}_{L1}$ (see Section 3.4.2.4) and is used in conjunction with Z-II for the direct comparison. The set $\mathcal{J}$ changes for XY-I with every test to the set $\mathcal{J}$ used by its competitor. Furthermore, method XY-II with $\mathcal{J} = \mathcal{J}_0$ (incl. performance increasing settings, see Table 3.4) is used to show the best possible localization accuracy which can be achieved by the introduced approaches if all measures are taken.

Again, Monte Carlo simulations are used for the direct comparison of the approaches. The settings previously described in Section 3.4.5 reflect good approximations of the error values to be expected in later real-world experiments, and are therefore re-used to show the expected performance of the approaches for the localization task at hand:

- $\sigma_T = 1$ tick, $f_s = 192$ kHz and therefore $\sigma_{RD} \approx 1.79$ mm.
- A systematic error is assumed in the TDEs with uniform distribution over the interval $[-0.1, 0.1]$ ticks.
- The positioning error of the microphones is modeled with $\sigma_{xy} = 0.5$ mm and $\sigma_z = 0.001$ m.

If an approach delivers multiple position estimates, the best candidate is chosen for the evaluation. For the comparison with Fang’s approach and Bucher and Misra’s method, the $120^\circ$ layout (UMUAA for $N = 4$ with $r = 0.5$ m) is used. For the algorithm introduced by Brandstein et al. the sensor layout depicted in Figure 3.27 is used.

![Microphone array]  
Figure 3.27: Microphone array used for the direct comparison with the LX approach.

The data of 15 iterations over the whole working area are used to create the result plots. Following settings are used:

$$p_s = \begin{bmatrix} r_{xy} \cos \alpha \\ r_{xy} \sin \alpha \\ R_s \sin \beta \end{bmatrix}^T$$

$\alpha \in [0; 360[^\circ$  
$\Delta \alpha = 10$ degrees  
$\beta \in [5, 10, 30, 60, 80, 85]$ degrees  
$r_{xy} = R_s \cos \beta$
3.6. Performance analysis: Simulation

Figure 3.28: Box-and-whisker plot depicting the positioning error in $x$-$y$ direction (top) and elevation (bottom) of the approaches with the sound source in close proximity to the center of the sensor array: XY-I (black), Fang (blue), XY-II (red). The black horizontal line depicts the bound of ±0.25 m.

Figure 3.29: Box-and-whisker plot showing the positioning error for larger distances from the acoustic source to the center of the microphone array: XY-I (black), Fang (blue), XY-II (red). The red line depicts $f(x)=x$.

Figure 3.30: Number of results for each of the position estimators: XY-I (black), Fang (blue), XY-II (red). The gray horizontal line shows the maximum possible amount of results.
Chapter 3. An acoustic localization approach for precise landings on a helipad

Figure 3.3.1: Box-and-whisker plot depicting the positioning error in x-y direction (top) and elevation (bottom) of the approaches with the sound source in close proximity to the center of the sensor array: XY-I (black), Bucher and Misra (brown), XY-II (red). The black horizontal line depicts the bound of $\pm 0.25$ m.

Figure 3.3.2: Box-and-whisker plot showing the positioning error for larger distances from the acoustic source to the center of the microphone array: XY-I (black), Bucher and Misra (brown), XY-II (red). The red line depicts $f(x)=x$.

Figure 3.3.3: Number of results for each of the position estimators: XY-I (black), Bucher and Misra (brown), XY-II (red). The gray horizontal line shows the maximum possible amount of results.
3.6. Performance analysis: Simulation

**Figure 3.34:** Box-and-whisker plot depicting the positioning error in x-y direction (top) and elevation (bottom) of the approaches with the sound source in close proximity to the center of the sensor array: XY-I (black), Brandstein et al. (pink), XY-II (red). The black horizontal line depicts the bound of ±0.25 m.

**Figure 3.35:** Box-and-whisker plot showing the positioning error for larger distances from the acoustic source to the center of the microphone array: XY-I (black), Brandstein et al. (pink), XY-II (red). The red line depicts f(x)=x.

**Figure 3.36:** Number of results for each of the position estimators: XY-I (black), Brandstein et al. (pink), XY-II (red). The gray horizontal line shows the maximum possible amount of results.
In addition, the origin of the sensor array coincides with the origin of the coordinate system. Furthermore, for each iteration the faulty microphone positions as well as the systematic errors for each RD have been kept constant.

One can see in Figures 3.28 and 3.29 that Fang’s approach has identical performance with the introduced method XY-I\(^3\). This makes sense, because both methods use exactly the same RDs and both approaches are not based on an approximated model nor do they linearize the hyperbolic localization problem. Both algorithms therefore also deliver the same amount of results (see Figure 3.30).

Method XY-II has a better localization accuracy than Fang’s method close to the origin of the sensor array and towards the border of the working area (≥ 10 m). The latter increase in position accuracy can be traced back to the outlier rejection process: Intermediate position estimates are rejected which are located outside the working area. Therefore the localization accuracy of the final position estimate increases, which however also leads to a slight decrease in the number of position estimates which can be calculated towards the end of the working area (all intermediate \(p'_s\) are rejected).

The calculated \(z\) values of method XY-II close to the center of the sensor array have a higher error attached to them than the \(z\) coordinates gained by Fang’s algorithm and approach XY-I. This seems strange, especially because method XY-I uses the same method (Z-II) to calculate the \(z\) value and starts its calculation with a worse estimate for the \(x\) and \(y\) coordinate. The result can be explained as follows: Small errors in the RDs and positions of the microphones lead to errors in the eccentricity of the hyperboloids. The resulting errors are most prominent close to the focal points of the hyperboloids. With all sensors being equally distributed around the sensor array origin, this results in larger errors around the origin. Assuming a Gaussian error, the errors should on average cancel themselves out. However, with every RD being subject to a small systematic error, this does not hold true anymore. With \(J = J_0\) more systematic errors are present than for \(J = J_{LI}\).

Method XY-I shows superior localization performance to Bucher and Misra’s algorithm (see Figures 3.31 and 3.32). In addition, the number of estimates which can be calculated at each distance are significantly lower for Bucher and Misra’s method (see Figure 3.33). Even though the algorithm performs worse than the introduced approaches, it would still be possible to use the method for the UAV localization task.

\[\begin{align*}
\end{align*}\]

\(\text{Figure 3.37: CRLB for the sensor layout depicted in Figure 3.27 for } p_s = [x \ y \ 1]^T \text{ m with } \Delta x = \Delta y = 0.25 \text{ m and } x, y \in [-15, 15] \text{ m as well as } \sigma_{RD} = 0.001 \text{ m. Only the sensor pairs used for the comparison of the localization approaches have been utilized for the CRLB calculation (see Table 3.4).}
\]

\(^3\)XY-I / XY-II are further used to refer to method XY-I / XY-II with the settings specified in Table 3.4.
For the comparison of the introduced approaches with the method developed by Brandstein et al. a different sensor array layout has to be used (see Figure 3.27). The sensor array is not a UMUAA anymore and does not provide a uniform CRLB distribution (see Figure 3.37). Therefore, the results of this test and the plots gained from the previous comparisons will have a different basic characteristic due to the distinct underlying CRLB distributions.

Figure 3.34 shows the problem from which the LX method suffers due to the approximation of its underlying model: Close to the sensor array the approximation errors reach their maximum, which results in position estimates with low localization accuracy. In contrast, for distances further away from the origin (≥ 4 m), the LX method performs identical to or outperforms method XY-I (see Figure 3.35). However, the amount of results delivered by the LX approach are far less than the amount of position estimates provided by method XY-I. The algorithm XY-II using all available TDOAs still outperforms the LX method.

Focusing solely on the amount of position estimates available for each range, method XY-I shows a significant improvement above the methods of Bucher and Misra as well as the approach described by Brandstein et al. (see Figure 3.36). In addition, method XY-I results in position estimates with higher localization accuracy than Bucher and Misras algorithm, and provides identical performance to Fang’s method for \( J = J_{11} \) and \( N = 4 \). With respect to the helicopter localization task, method XY-I also outperforms the LX method, which yields its lowest localization accuracy close to the center of the sensor array. This is however the region where the most accurate position estimates are necessary.

In summary, method XY-I with \( J \neq J_0 \) is suited best for the UAV localization task of all the reviewed approaches. Only method XY-I and XY-II are flexible enough to be used with \( J = J_0 \). With this setting the overall localization accuracy reaches its maximum. With both methods performing very similar under these circumstances (see Section 3.4.5), either one of the methods can be used to solve the localization task. It has to be noted that the trust value has not been utilized during the comparison because similar confidence values are not available for the other approaches.

3.6.2 General comparison

With the approaches based on the spherical model being the most commonly used methods due to their high localization accuracy, a performance comparison of these algorithms with the introduced methods would be very interesting. However, for a fair comparison between multiple approaches one has to make sure that both the microphone array as well as the noise affected RDs used for the calculation of the position estimate are the same for all methods. Furthermore, all additional parameters like the systematic error and the error of the microphone positions have to be identical as well. Only if the sensor arrays achieve identical CRLB as well as all input parameters are equal for all approaches, the comparison can be deemed as fair. While these rules can be (and have been) applied to the previous analysis, they cannot be applied for the comparison of the introduced methods with the approaches based on the spherical model: The algorithms based on the spherical model cannot cope with planar microphone arrays, and both method XY-I and XY-II have been specifically developed for this purpose and cannot be used with non-planar microphone array geometries. To solve this dilemma and still allow a reasonably fair comparison between the algorithms, two similar microphone arrays will be
employed. Figure 3.38 shows the sensor array layouts. The left plot shows the UMUAA for \( N = 6 \) and \( r = 0.5 \text{ m} \) which is used for methods XY-I and XY-II. The layout shown in the right plot is based on the former, however the sensors are moved off the \( x-y \) plane. The distance \( D_{ij} \) with \( j = 2, 3, \ldots, 6 \) of this layout has been adjusted to gain a very similar CRLB distribution for both sensor arrays over the whole working area.

![Sensor array layouts](image)

*Figure 3.38:* Sensor arrays used for the performance analysis: Microphone array used with method XY-I and XY-II (left), sensor array used with the approaches based on the spherical model (right).

The microphone positions for the array used with the approaches based on the spherical model are as follows:

\[
\begin{align*}
\mathbf{p}_1 & = [0.0 \ 0.0 \ 0.0]^T \\
\mathbf{p}_2 & = [0.51 \ 0.0 \ 0.0]^T \\
\mathbf{p}_3 & = [0.15 \ 0.47 \ -0.1]^T \\
\mathbf{p}_4 & = [-0.38 \ 0.27 \ 0.2]^T \\
\mathbf{p}_5 & = [-0.38 \ -0.27 \ -0.2]^T \\
\mathbf{p}_6 & = [0.15 \ -0.47 \ 0.1]^T
\end{align*}
\]

With both arrays sharing the same basic array geometry, they also provide a similar distribution of the CRLB. The magnitude of the CRLB for different ranges from the origin of the microphone array to the signal source is also much the same for both sensor arrays (see Figure 3.39). To compensate for the remaining deviations between the two CRLB distributions, the difference between the trace of the bias compensated sample variance and the CRLB is used to compare the approaches. Mathematically this can be described by

\[
\bar{p}_s = \sum_{i=1}^{I} \hat{p}_{s_i}
\]

\[
\Delta\sigma^2(p_s) = tr\left(\frac{1}{I-1} \sum_{i=1}^{I} (\hat{p}_{s_i} - \bar{p}_s)^2\right) - \Xi(p_s)
\]

where \( \bar{p}_s \) is the mean of the estimates with the acoustic source being located at \( p_s \), and \( \hat{p}_{s_i} \) is the position estimate of the \( i \)-th iteration of \( I \) iterations calculated at the current source position. In addition, the bias of the estimates is evaluated:

\[
\text{bias} = \left\| p_s - \bar{p}_s \right\|_2
\]

In other words, the methods will be compared based on how close their estimates approach the CRLB and on the magnitude of their bias.
3.6. Performance analysis: Simulation

Figure 3.39: CRLB for $\beta = 3_{1,1}$ for the UMUAA (black) and the sensor array used for the approaches based on the spherical model (red).

Figure 3.40: Box-and-whisker plot of the difference (bias compensated sample variance - $\Xi$): Method XY-I (black), XY-II (red), Schmidt (blue), Friedlander (brown), Smith and Abel (violet), Chan and Ho (orange).

Figure 3.41: Box-and-whisker plot showing the bias of the approaches: Method XY-I (black), XY-II (red), Schmidt (blue), Friedlander (brown), Smith and Abel (violet), Chan and Ho (orange).
Chapter 3. An acoustic localization approach for precise landings on a helipad

A 100-trial Monte Carlo simulation with the acoustic source being assumed at the same positions as previously in Section 3.6.1 and $\sigma_r = 1$ tick with $f_s = 192$ kHz and therefore $\sigma_{RD} \approx 1.79 \text{ mm}$ is used for the performance evaluation. This time however, no errors in the positions of the microphones and no systematic error in the RDs is used. This is necessary to allow a meaningful comparison of the bias of the approaches.

One can see in Figure 3.40 that the estimates of all methods closely approach the CRLB, if $p_s$ is close to the sensor array. For $R_s \leq 5 \text{ m}$, the deviation of the results from the CRLB is insignificant (see scale of top plot of Figure 3.40). For distances further away from the microphone array, one can see that the method of Chan and Ho as well as the algorithm of Smith and Abel perform best of all the approaches. Method XY-I performs generally better than XY-II and can compete with the other approaches, especially towards the far end of the working area. Method XY-I then outperforms Schmidt’s and Friedlander’s approaches.

A significant difference in the performance of the algorithms can be seen in the bias (see Figure 3.41). Both introduced approaches have a higher bias variation than the other methods, if $p_s$ is close to the sensor array. The median bias value of methods XY-I and XY-II stay however close to the median bias values of the other approaches. Generally, the method of Chan and Ho as well as the algorithm of Smith and Abel outperform the other approaches by featuring both small bias and estimates which closely resemble the CRLB.

3.7 Performance analysis: Experiments

3.7.1 Sensor array

A landing pad (see Figure 3.42) was constructed and four microphones (NADY CONDENSER MEASUREMENT MICROPHONE CM-100) were embedded into the helipad based on the UMUAA layout for $N = 4$ and $r = 0.53 \text{ m}$. A microphone amplifier (YAMAHA Microphone Amplifier MLA7) and an ADC box (EDIROL FIREWIRE AUDIOCAPTURE FA-101) in conjunction with a notebook were used to record the four audio channels simultaneously with a sampling frequency of 192 kHz.

![Figure 3.42: The landing pad with the embedded sensor array which was used for the experiments.](image)
3.7. Performance analysis: Experiments

3.7.2  Experiment I - Indoor, hand guided sound source

3.7.2.1  Experimental setup

Based on the simulations, one can expect a high localization accuracy from the introduced approaches close to the landing pad. A ground truth system which provides at least the same localization accuracy as the methods to be tested is necessary for a meaningful real-world performance evaluation. The first experiment was therefore conducted indoors with a hand held sound source, which allowed the use of a VICON system (T-SERIES T10) to provide ground truth measurements. The VICON setup which was available for the experiment can calculate positions of specially marked objects in its workspace with a localization error of less than 1 mm.

![Image of the setup](image)

**Figure 3.43:** The author conducting the experiment. A camera of the VICON setup can be seen in the top section of the left picture. The necessary markers for the optical system were mounted on the speaker board and the landing pad (right).

A 160 Hz square wave was chosen as sound signal which can be accurately and efficiently implemented with a PWM output of a microcontroller. An ATMEGA32 microcontroller was used for this task. The landing pad as well as the acoustic source (PC speaker) were equipped with VICON markers (see Figure 3.43 right), which are necessary for the vision-based localization system to work. Both the microcontroller and the speaker were battery powered to allow the sound source to be freely moved through space.

3.7.2.2  Time delay estimation

The recorded sound signals were bandpass filtered to gain the odd harmonics\(^4\) of the square waves of the sound signal. Due to a large amount of low frequency noise present, the base frequency of 160 Hz was also filtered out and not used for the time delay estimation. Air-conditioning noise, common office noise and reverberations due to multipath propagation were the main error sources during the experiment.

---

\(^4\)The duty cycle of a square wave is 50% and therefore only the odd harmonics \((n \cdot 160\text{ Hz with } n = 1, 3, 5, \ldots)\) are present.
Figure 3.44: Time delay estimates: unfiltered (red), filtered (green). From top to bottom: $d_{12}$, $d_{13}$, $d_{14}$, $d_{23}$, $d_{24}$, and $d_{34}$. 
3.7. Performance analysis: Experiments

The GCC with phase transform weighting (also known as GCC-PHAT) was used to extract the time delays between the filtered signals. GCC-PHAT provides good TDEs in both indoor and outdoor environments and is computationally efficient [90] [93, p. 1047]. The results of the time delay estimation can be seen in Figure 3.44 (red lines). The TDE curves are very smooth but feature some significant outliers which can be traced back to the presence of additional noise sources, whose frequencies overlap with the frequency bands occupied by the sound signal. This makes a filtering of the TDEs necessary.

3.7.2.3 TDE filtering

A parabola is fitted through the TDEs of the last $t_{\text{filter}}$ seconds. If the sum of the residuals are below a specified threshold $\epsilon_{\text{res}}$, the gained parabola is assumed to resemble the TDE curve of the last $t_{\text{filter}}$ seconds sufficiently close to start the filtering process. If however the sum of the residuals exceeds $\epsilon_{\text{res}}$, the TDE curve of the last $t_{\text{filter}}$ seconds has already outliers in it, preventing the initialization of the filtering process until a $t_{\text{filter}}$ seconds interval of outlier-free TDEs is available.

If the TDE $\tau_0$ of the current time step is significantly different ($\geq \Delta_\tau$) than its predicted value $\tau_p$ calculated by evaluating the parabola at the current time step, it is assumed that $\tau_0$ is an outlier. In this case, $\tau_p$ is used to replace $\tau_0$. To ensure that the filtered TDEs follow closely the actual time delays, not more than 40% of the TDEs is allowed to be replaced.

The success of the introduced filtering method is based on the assumption that the TDEs do not change significantly over a short period of time as long as the sound source does not execute erratic maneuvers. This holds true for both the hand guided sound source in this experiment as well as for the flight path of the UAV in the following experiment.

$t_{\text{filter}}$ should be chosen to be longer as interfering noise sources are assumed to be present at each occurrence. An overestimation of $t_{\text{filter}}$ can break the filtering due to the problem that the TDE graph can no longer be approximated with a parabola.

The result of the filtering process for

$$t_{\text{filter}} = 3\, \text{s}$$
$$\epsilon_{\text{res}} = 1000 \, \text{ticks}$$
$$\Delta_\tau = 100 \, \text{ticks}$$

is shown in Figure 3.44 (green lines). One can see for $d_{14}$ and $d_{34}$ that the initialization of the method is achieved only after the initial outliers move out of the TDE set bounded by $t_{\text{filter}}$. The first outliers are therefore not filtered out.

The advantage of the filter is that it can be used online, and the parameter set for the filter can be determined after a test run in the application environment. The parameters can then be kept constant. In addition, the filter does not depend on the position of the sound source nor the knowledge of the motion model of the acoustic source.

3.7.2.4 Localization results

The result of method XY-II in conjunction with Z-II and Schmidt’s correction step can be seen in Figure 3.45. The coordinate system of the ViCON setup was used as reference frame for the creation of the plots. The z axis points into the ground, which results in
negative \( z \) values. Due to the confined space of the room, the maximum possible distance between the sensor array and the sound source is limited. The measurements shown in Figure 3.45 are the raw position estimates provided by the method and have not been filtered.

![Figure 3.45](image)

One can see that the \( x \) and \( y \) coordinates calculated by the introduced method closely follow the ground truth measurements. The high perturbations in the interval \([10, 15]\) s can be traced back to the larger amount of errors in the TDEs, even after the filtering process. Generally, the \( z \) values deviate more from the ground truth measurements than the \( x \) and \( y \) coordinates. However, they still closely resemble the reference elevations.

<table>
<thead>
<tr>
<th>Table 3.5: Localization errors of the approaches.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>1st quartile</td>
</tr>
<tr>
<td>median</td>
</tr>
<tr>
<td>3rd quartile</td>
</tr>
</tbody>
</table>

Figure 3.46 shows a comparison between the results gained by the introduced methods
3.7. Performance analysis: Experiments

![Figure 3.46: Results gained by the introduced methods and their direct competitors: Box-and-whisker plot of the Euclidean distance between the position estimates gained by the methods and the corresponding localization information provided by the VICON system.](image)

and their direct competitors. One can see that all methods perform very similar. The detailed results can be found in Table 3.5. The position estimates calculated by Bucher and Misra’s algorithm have the highest overall error, followed by method XY-I and XY-II. Fang’s approach performs best for the data gained from the experiment. This can be traced back to the TDEs: Fang’s method solely uses the TDEs $d_{12}$, $d_{13}$, and $d_{14}$. These are the TDEs with the least amount of errors (see Figure 3.44). Bucher and Misra’s approach uses the TDE set \{$d_{12}$, $d_{13}$, $d_{32}$, $d_{34}$\} which has a larger amount of errors than the TDE set used by Fang, which results in a worse starting position for Bucher and Misra’s algorithm. This also holds true for methods XY-I and XY-II which both use $J = J_0$. However, taking all available TDEs into account and additionally utilizing Schmidt’s correction step allows methods XY-I and XY-II to provide position estimates with accuracies close to Fang’s approach which only uses the least faulty TDEs. In summary, $J = J_0$ is only beneficial if the additional TDEs can be used to reduce the overall error in the system, but do not introduce additional errors at time steps at which the LI TDEs already have problems. The dilemma is that it is not possible in real-world setups to know how the TDEs will behave beforehand.

3.7.3 Experiment II - Outdoor, intrinsic sound

3.7.3.1 Experimental setup

A second experiment was conducted outdoors with the aim to localize the helicopter UAV based on its main rotor noise. During the experiment, the aerial platform was flown remote controlled in close proximity to the landing pad to simulate landing approaches. A car battery was used in conjunction with an inverter to power the sensor array (ADCs, amplifiers, microphones). Figure 3.47 shows the experimental setup used for the outdoor acoustic experiment.

Based on the results of Experiment I, one can expect the introduced acoustic localization approach to provide an accurate position estimate of the helicopter UAV close to its landing platform. A ground truth localization system for the outdoor experiment featuring at least the same localization accuracy as the developed acoustic approaches was not available. The GPS position of the UAV has been recorded. However, the position information cannot be used to estimate the error of the acoustic localization approach due to its low accuracy.
3.7.3.2 Time delay estimation

The main rotor speed during the flight varied in the interval of [2010, 2020] rpm (extracted from the motor controller log). The most prominent frequency produced by the main rotor is double the frequency of the main rotor shaft (due to the two main blades). The sound recordings were bandpass-filtered to extract the main rotor base frequency (~67 Hz) and its first three harmonics. The filtered signal was then used as input for the GCC-PHAT method to extract the TDEs.

Figure 3.50 shows the extracted TDEs (red lines) of the whole flight. In comparison to the results of Experiment I (see Figure 3.44) one can see that the TDEs are generally more noisy and feature three intervals with high outliers. Following error sources can be made responsible for the noisy TDEs:

- Wind across the microphones consists of mostly low frequencies with high amplitudes. These acoustic signals can overpower the main rotor sound signal which is located in the same frequency band.

Figure 3.48: Frequency spectrum of the unfiltered sound signal with only a small amount of low frequency noise present. The base frequency of the helicopter UAV (~67 Hz) as well as the first three harmonics can be clearly identified.
3.7. Performance analysis: Experiments

Figure 3.49: Frequency spectrum of the unfiltered sound signal with a high amount of low frequency noise present. One can see that the base frequency of the helicopter UAV as well as the first three harmonics are overpowered by the low frequency noise.

Figures 3.48 and 3.49 illustrate the problem: Figure 3.48 shows a part (interval [0, 300] Hz) of the normalized frequency spectrum of one channel of the unfiltered sound recordings at a time where only a small amount of low frequency noise is present. A clearly defined peak can be seen for the base frequency as well as for the first three harmonics. Figure 3.49 shows a problem case in which the base frequency as well as the first three harmonics have been overpowered by low frequency noise. The bandpass-filtered signal therefore does not represent the helicopter main rotor sound signal anymore which leads to faulty TDEs.

This problem gets exaggerated during landing approaches when the helicopter UAV acts as an additional wind source due to the air moved by its main and tail rotor. The three time intervals in Figure 3.50 which feature high outliers can be traced back to strong winds blowing across the microphones during UAV landing approaches. Unfortunately, the problem reaches its maximum when the helicopter is close to touchdown and when the highest position accuracy is necessary. A workaround to this problem is to use an artificial sound source on-board the helicopter during landing maneuvers and to localize the UAV based on this artificial sound source. The on-board speaker could be dual-used for this task. The artificial sound signal should not be located in the lower frequency spectrum and should not coincide with frequencies of other helicopter parts as well as payload actuators and sensors. The artificial sound signal can be chosen to maximize the accuracy of the TDEs.

- For the time delay estimation process, the sound source is assumed to be a point source. In reality however, the sound is produced all the way along the main rotor blades, each of them having a length of 0.73 m.
- Due to the low frequency of the main rotor noise, the peaks in the cross-correlogram of the GCC-PHAT method are wide. This results in TDEs with lower resolution.

The previously introduced TDE filtering process was used again to reduce the amount of outliers. Following parameter set was used:

\[
\begin{align*}
    t_{\text{filter}} &= 2 \text{s} \\
    \epsilon_{\text{res}} &= 1E5 \text{ticks} \\
    \Delta_\tau &= 500 \text{ticks}
\end{align*}
\]
Figure 3.50: Time delay estimates: unfiltered (red), filtered (green). From top to bottom: $d_{12}$, $d_{13}$, $d_{14}$, $d_{23}$, $d_{24}$, and $d_{34}$. 

122
While the filtering process reduces the amount of outliers, it does not cater for the general noise which affects the TDEs. Therefore, an additional filtering step was implemented: The weighted average result of the last five time delay estimations is used as input for the acoustic localization approach:

\[ \tau_{\text{result}} = \frac{\tau_{n-4} + 2\tau_{n-3} + 3\tau_{n-2} + 4\tau_{n-1} + 5\tau_n}{15} \]

Both filters can be used online and their parameters can be determined after a test run in the application environment. The resulting TDEs can be seen in Figure 3.50 (green lines).

### 3.7.3.3 Localization results

Figure 3.51 shows the unfiltered results of the position estimation of method XY-II with Z-II and Schmidt’s correction step (green data points). One can see that the \( x \) and \( y \) position of the helicopter UAV could be determined for all parts of the flight where good TDEs were available (see Figure 3.50). The generally noisy TDEs however prevented the calculation of the \( z \) coordinate of the helicopter UAV for most of the flight. Furthermore, during the three time intervals with a high amount of TDE outliers, no accurate position estimate could be calculated.

![Localization results](image)

*Figure 3.51: Results of method XY-II with Z-II and Schmidt’s correction step (green data points) in comparison to the position estimates calculated with Fang’s approach (red data points).*
Chapter 3. An acoustic localization approach for precise landings on a helipad

In Experiment I, Fang’s method was found to estimate the position of the sound source accurately. The localization approach is therefore used as a benchmark in this experiment. One can see in Figure 3.51 that the position estimations provided by Fang’s method (red data points) coincide with the position estimates of the approach XY-II for all sections of the flight where good TDEs could be extracted. However, for most of the flight Fang’s method does not provide any position estimates. Method XY-II can still provide a position estimation for the helicopter in 2D.

3.7.3.4 Summary

The experiment shows that it is generally possible to localize the helicopter UAV based on its intrinsic sound signature with the introduced acoustic localization approaches when the microphone array is embedded into the landing platform and quality TDEs are available. Localizing the helicopter based on its main rotor noise however has its limitations:

- During the landing approach the aerial platform acts as an additional wind source due to the air moved by its main and tail rotor. The acoustic signal of the wind can overpower the sound signal of the helicopter in the sound recordings which leads to faulty TDEs.
- The generally noisy TDEs (due to the point source assumption and the wide peaks in the cross-correlogram due to the low base frequency of the main rotor) do not allow the introduced acoustic approaches (nor Fang’s method and Bucher & Misra’s method) to calculate the z coordinate of the helicopter UAV.

The limitations are caused by either faulty or low quality TDEs. To gain better TDEs, an acoustic signal which is not located in the low frequency band should be used for the localization instead of the main rotor sound signal. The on-board speaker could be used to send out an artificial acoustic signal during landing approaches. This would allow one to extract more accurate TDEs from the recorded sound signals and to gain improved position estimates for the helicopter UAV.

3.8 Conclusions

This chapter reviewed existing TDOA-based acoustic localization approaches based on their respective models. Furthermore, two new methods for localizing an acoustic source in the near- and far-field of planar microphone arrays were introduced. The developed approaches use noise affected TDEs gained from planar sensor arrays with four or more sensors to estimate the position of the acoustic source. The methods feature more flexibility than other approaches (e.g. work with both $J = J_0$ and $J = J_{LL}$, sensor array geometry is not fixed, correction step). Furthermore, their predictable computation time makes them suitable for real-time implementations.

The position estimation accuracy of the discussed acoustic approaches is dependent on the sensor array geometry. Optimal sensor array geometries discussed in the literature were reviewed. It was argued that CRLB optimum sensor array geometries do not necessarily yield the maximum benefit for a certain application. Subsequently, two new classes of sensor array geometries (MUAAAs, UMUAAs) were introduced which are beneficial for localizing the developed helicopter UAV in its landing area. Furthermore, a trust value
was introduced for the approaches which allows the user to decide whether the position estimate should be used for further calculations or not.

Simulations showed that the developed approaches feature equal or improved localization accuracy than their direct competitors. Experiments were conducted to show the usefulness of the approaches in real-world setups. A first experiment conducted indoors showed a median localization error of $\leq 53$ mm if the sound source is kept within a radius of 3 m from the center of the sensor array. Furthermore, an outdoor experiment was used to show that it is generally possible to localize a helicopter UAV based on its intrinsic sound while it is flying close to the sensor array and accurate TDEs are available. It is however recommended to use an artificial sound source instead of the main rotor noise to be able to extract quality TDEs which allows the position of the helicopter to be determined with increased reliability and accuracy. Broadband signals specifically tuned to form sharp peaks in the cross-correlogram of the time delay estimator should have a positive effect on the quality of the position estimates. Furthermore, the use of multiple artificial sound sources on-board the UAV could be used to determine not only the position of the helicopter but also its attitude.

Generally, the presented strategies enable autonomous landing systems for a wider range of small scale helicopters for which additional instrumentation (and accompanying on-board processing) with e.g. laser range finders or cameras is not a feasible or preferred option. The discussed approaches can also be utilized in other areas, e.g. for speaker localization and tracking.
Modular UI framework

Chapter summary and structure

A monitoring suite for the observation of an onshore geological CO₂ storage site consists of a mixture of sensors, sensor networks and possibly mobile robots to detect containment breaches. To allow the operator a comprehensive interpretation of the measurement data gained by all individual devices and to make full use of the mobile robots included in the monitoring suite (e.g. the developed helicopter UAV), an application providing tools for the simultaneous management of multiple sensor and robotics systems in addition to measurement data analysis tools is necessary.

At a commercial onshore geological CO₂ storage site, one can expect to find monitoring devices and mobile robots from a range of vendors. Furthermore, the hardware configurations of the mobile robots as well as the software frameworks deployed on them can be expected to vary. Supporting the management of such a heterogeneous environment from a single application necessitates the development of a new software framework which can not only cope with such an environment but can also accommodate ongoing changes.

To tackle this problem, dedicated UI frameworks are introduced and their advantages and requirements are discussed. An architecture describing the structure of such software systems in a programming language and graphical user interface (GUI) toolkit independent way is then explained. The architecture aims to minimize source code development by splitting the software framework into reusable modules. Furthermore, the suitability of the architecture is shown by using its reference implementation in three robotics scenarios targeting various classes of sensors and robots driven by different software frameworks. The GCS for the developed helicopter UAV was implemented using the reference implementation and is one of the examples.
4.1 Introduction

The introduced UAV for the aerial monitoring of CO₂ concentrations over onshore geological storage sites forms one part of a comprehensive monitoring suite. To gain the maximum amount of information out of the collected measurement data of all individual monitoring techniques, a unifying data analysis tool is necessary. Ideally, such a software can not only connect to and gain real-time updates¹ from multiple sensors, sensor networks, and mobile robots simultaneously, but can also be used to manage the mobile robots (path changes, robot behavior changes, monitoring of internal robot state) which are a part of the monitoring suite. This would allow the operator to mobilize robots based on findings gained from the latest measurement data while measurements collected by the robots could be directly incorporated into the ongoing analysis. Robots are increasingly used to complement existing monitoring solutions, especially environmental systems (see [120] and citations therein). The need for a comprehensive tool suite for sensor and robot management, as well as for measurement data analysis is therefore not restricted to the task at hand.

4.2 Problem analysis

Traditionally, the management of robots as well as the accompanying measurement data analysis is done via UIs provided by the software system used to implement the control algorithms and the behavior of the robot. A robotics framework is commonly used for this task. Robotics frameworks are software systems which implement robotics specific tools, functionality, drivers and algorithms. They offer the robot application developer a modular² architecture, allow the re-use of implemented and tested device drivers and popular algorithms and take the burden of developing an effective data flow between modules of one robot and modules of multiple robots across a network from the application developer. The system can implement its own middleware or it is built upon an existing middleware solution. A robotics framework abstracts the hardware, allowing the developer to implement the robot behavior based on sensor data gained by a device of a specific device class (e.g. laser range finder) without having to know the actual type of sensor (e.g. Sick LMS200) used on the robot. This also allows behavioral modules to be used across robot hardware configurations and for different robotics applications without any source code changes. Due to the possibility to save a considerable amount of time by using a robotics framework, their popularity (also in commercially available robots, e.g. [121]) is therefore not surprising.

A variety of robotics frameworks is available today [122]. Some systems are open-source (e.g. SmartSoft [123], Player [124], Carmen [125], Robot Operating System (ROS) [126]), while others are of commercial nature (e.g. Microsoft Robotics Developer Studio (MSRDS) [127]). New robotics frameworks are still being developed to cope with shortcomings of existing systems. This diversity is fortunate and allows robotics application developers to choose the system which copes best with their specific needs.

¹If permitted by the monitoring technique.
²Due to different naming conventions in various robotics frameworks, the notion of a module is used in this thesis to describe the fundamental, encapsulated unit which forms the basis for code reuse inside a robotics framework in a platform and framework independent manner.
4.2. Problem analysis

The encapsulation of the robotics framework modules is the key to their reuse in various robotics scenarios across different hardware platforms and robot configurations. To also allow the reuse of the robotics framework UIs, each UI is ideally coupled to only a single robotics framework module. This results in desktops being cluttered with an increasing amount of windows (see e.g. Figure 4.1) in which the number of windows increases with the amount of framework modules being used (and with the complexity of the implemented scenario). The missing integration of the individual UIs into a single solution makes the management of complex robot systems unintuitive and cumbersome, and requires the end-user to have an understanding about the control systems implemented on the robots. While the one-window one-module strategy has its limitations, it allows the reuse of robotics framework UIs across various robotics scenarios.

Requirements can change throughout a project and at some point robotics application developers might wish to change from one robotics framework to another or even use multiple robotics frameworks simultaneously (e.g. in a heterogeneous group of robots). To increase the code reuse between robotics frameworks, Makarenko et al. [129] propose to implement drivers and algorithms in libraries. This enforces a clear separation between robotics framework specific APIs and the implemented logic. The libraries can then be used in various robotics frameworks, overcoming the problem of the Software Lock-In (also [129]). If one complies with this guideline, merely the glue code between the robotics framework and the library itself has to be rewritten. After this integration step both robotics frameworks profit from bug fixes and extensions to the library.

Unfortunately, the reuse of robotics framework UIs is limited to applications using robots which run the same robotics framework. That means that for example a PLAYER UI cannot be simply used with a CARMEN module and vice versa. Therefore, multiple implementations of the same UIs are available for various robotics frameworks, multiplying the overall implementation effort (see e.g. Figure 4.1). With the increasing complexity of the robot behavior, UIs grow more and more complex, and their implementation and maintenance times rise. The creation of realistic 3D environments can be seen as the tip of the iceberg where the implementation often takes a considerable amount of time and also requires a fair bit of programming expertise. Up to now, such UIs are often implemented from scratch over and over again. This time consuming and error prone practice would not be necessary if one could use an implementation of such a UI (e.g. map viewer, waypoint manager, measurement viewer) across robotics frameworks without any source code changes.

![Figure 4.1: Screenshots of a PLAYER/STAGE/GAZEBO setup [128] (left) and a MSRDS setup (right) featuring UIs provided by the robotics frameworks and their respective simulations.](image-url)
To avoid UI redevelopments, research groups usually utilize only a single robotics framework on all of their robots. In a commercial setup however, robots might be purchased from different vendors for distinctive tasks, which means that not necessarily all robots are implemented using the same underlying robotics framework. This leads to the problem that a system operator either has to learn, operate and maintain multiple robot management tools next to each other or has to pay for a custom developed solution, leading to increased overall costs (development and maintenance expenses).

There are several problems which restrict a simple re-use of UIs between robotics frameworks. The main challenge is that an implementation of the protocol of each of the robotics frameworks to be supported has to be available inside the UI implementation. Without that, the essential exchange of information between the UI and the robotics frameworks would not be possible at all. To enable a UI to be shared among robotics frameworks it is also necessary that the data structures used inside the UI implementations are based on the common set of information shared between the robotics frameworks specific data types used to describe a sensor output or action. Additional information provided by a subset of the supported robotics frameworks has to be handled as optional information on top of the basic data structures. Only if that is the case, the re-use of UIs between multiple robotics frameworks can be achieved. The overhead which is tied to the implementation of both of these necessary features into an existing UI can be extensive depending on the number of robotics frameworks to be supported and the overall size of the UI itself.

In special cases, software bridges can enable the re-use of UIs between two robotics frameworks [129]. Nevertheless, such a solution is not feasible anymore as soon as more robotics frameworks are taken into consideration. Software bridges are very hard to maintain and sometimes are simply not possible due to various and at times incompatible approaches of the robotics frameworks. Such differences can come in the form of programming languages used to implement the frameworks, opposing middleware systems, and contradicting design ideas and architectures. Software bridges are therefore not a feasible option for the development of a general solution.

The key to allow high level UI reuse based on modules instead of source code snippets in a robotics framework independent manner is to separate all parts responsible for the graphical representation and user interaction away from the robotics frameworks themselves and to move them into a dedicated UI framework. That clearly draws a line between modules responsible for the robot behavior / control (model) and modules for the visualization (view). Nevertheless, the design of the UI framework also has to maintain the separation of concerns inside the UI framework (separation of connections to robotics frameworks and graphical representations) to allow the usage of UIs across robotics frameworks.

### 4.3 Dedicated UI frameworks

A robotics framework abstracts the hardware and allows the application developer to create behavioral modules based on device classes which are independent of the sensor or actuator model and make. The information inputs for a dedicated UI framework are data streams from both hardware abstraction modules as well as behavioral modules of different robotics frameworks. The task of a UI framework is to further abstract the device
4.3. Dedicated UI frameworks

classes of the individual robotics frameworks into a unifying device class interface. This enables the usage of UI implementations independent of the robotics frameworks used for the control of the robot(s).

4.3.1 Advantages

The introduction of dedicated UI frameworks has a number of advantages:

- **Easier comparisons:** Robots running different robotics frameworks can be monitored using the same UIs. That simplifies direct comparisons between robotics frameworks for robots executing the same task.

- **Robotics framework selection:** Available UI implementations of a particular robotics framework are not a factor to consider anymore when choosing a robotics framework for a specific task. Solely the advantages and disadvantages of a robotics framework regarding the control of the robot are relevant.

- **Write once, use often:** Instead of implementing a particular UI for each framework, each UI has only to be implemented once and can be used by multiple robotics frameworks. Users profit from faster availability of new UIs.

- **Diversity:** Users can choose between UI frameworks the same way as they choose their favorite window manager in a Linux environment. They can choose the most robust implementation or a UI framework which is best suited for their specific needs (e.g. a version targeting tablet computers, a solution designed specifically for end-users or robotics application developers). That also increases the competition between UI frameworks which can result in higher code quality.

- **Higher quality:** The amount of people interested in a specific UI implementation might be higher than it is at the moment. Problems should be exploited quicker which can result in faster bug-fixing leading to increased code quality and overall robustness.

- **Developer support:** The enforced separation of model and view in distinct frameworks frees the robotics behavior developer from any entanglement with the UI.

- **Opportunities for members of other communities:** The approach enables UI designers and developers to directly participate in robotics projects. That might result in more intuitive and usable UIs than currently available.

- **Simulation:** With a dedicated UI framework, the implemented UIs can be used both for simulated and real world scenarios. For the user it is only necessary to be familiar with one UI for both cases. In addition to that, simulators can use the UI framework for visualization purposes as well.

- **Central and distributed robot monitoring:** It is possible to either monitor a robot from a single machine or from a number of machines. If the user decides to monitor the robot in a distributed fashion, one UI framework instance would be running on each monitoring node with the same or different UIs.

- **Improved handling:** Each UI framework hosts its UIs as a single solution. That makes repeating rearrangements of UIs on the desktop obsolete and improves the handling of complex robot scenarios.
Chapter 4. Modular UI framework

- *Parallel development:* UIs and robotics framework modules can be implemented by different groups independently of each other at the same time.

### 4.3.2 Requirements

Dedicated UI frameworks will only get adopted if they can be integrated into existing solutions with minimal effort while providing the benefits outlined above. Therefore, following requirements have to be met.

#### 4.3.2.1 Compatible with existing robotics frameworks

The proposed UI framework has to be able to work together with existing robotics frameworks without the necessity to change any source code in the robotics framework modules. The UI framework solely has to provide an alternative graphical representation to the existing robotics framework specific UI implementations.

#### 4.3.2.2 Side-effect free

The UI framework has to be interference free regarding the behavior of the robot. That means e.g. the only effect of closing the UI framework during a monitoring process of a robot is that the user loses the ability to interact with the robot. The robot behavior itself is not allowed to change. That implies that the UI framework must not alter the behavior of the robotics system itself by any means except through commands given by the user. Therefore, the software framework deployed on the robot has to provide an interference free read access (e.g. via double or triple buffered structures protected against concurrent write access or dual-ported memory).

#### 4.3.2.3 Modular, reusable and extendable

The UI framework has to provide graphical representations of sensors and actuators as UI building blocks. One should be able to combine these building blocks to create graphical representations of robots with various configurations. Only if one of the building blocks is not available yet, it should be necessary for the developer to write source code. It has to be possible to use the UI building blocks in generic 2D and 3D representations of an application environment.

The proposed UI framework has to support the integration of new UIs and interfaces to robotics frameworks. It has to be possible to design and implement UIs for specific tasks without any knowledge about the robotics framework which will be used to control the robot.

#### 4.3.2.4 Support a broad range of systems

In order to cope with a comprehensive monitoring suite, the following classes of systems have to be supported by the UI framework:

- **Class I: Robotics framework driven mobile robots**
  The first class of robotics systems which has to be covered is robots which use a network-based robotics framework to access sensors and actuators, and to implement the platform behavior. For the purpose of this thesis we subdivide robotics
Figure 4.2: The systems which have to be supported by UI frameworks can be broadly divided into three classes: Robotics framework driven mobile robots (Class I), mobile robots which do not use a robotics framework (Class II), and direct sensing and control systems (Class III). Different communication strategies implemented in robotics frameworks make a further distinction necessary (Class I, Category I and II).
frameworks based on their communication strategy into two categories: The first category consists of systems like the PLAYER framework, which allows access to multiple robotics framework modules via a single communication link to the framework core. Systems of the second category (e.g. MSRDS) access each robotics framework module directly after retrieving its position in the network from a central registry (see Figure 4.2).

- **Class II: Mobile robots which do not use a robotics framework**
  The second class of robotics systems consists of mobile robots which do not utilize a robotics framework to fulfill their tasks. Members of this class are not exclusively, but very likely robots using embedded systems for their computation and might be in addition to that real-time systems. The differences between the systems of Class I and Class II are that the robots of the latter class are not necessarily connected to a network, and that their software framework most likely does not make all features of the connected sensors and actuators available to its clients. It is also not possible to directly access the devices of the system through the software framework deployed on the robot. The introduced CO$_2$ sensing UAV and its autopilot system is a representative of this class.

- **Class III: Direct sensing and control**
  Under specific circumstances (further discussed in section 4.4.1), sensors and actuators can be directly connected to the UI framework. This enables the user to directly control actuators and to gain measurements directly from sensors or sensor networks without the need for an intermediate software system of Layer II (see Figure 4.2, right).

The UI framework has to be able to manage individual robots, groups of robots (both homogeneous and heterogeneous), as well as multiple systems belonging to different classes at the same time in order to support comprehensive monitoring suites (e.g. a CCS monitoring solution consisting of stationary sensors, sensor arrays, and mobile robots like the developed UAV). Due to the possibly diverse software environment present, the UI framework has to be able to cope with the diversity of robotics frameworks in use.

For the remainder of this work, the term *backend* will be used as an umbrella term for the software system deployed on a robot (Layer II in Figure 4.2) without distinguishing between systems belonging to the first and second class.

### 4.3.3 Summary

Dedicated UI frameworks have to deal with the complexity of concurrent activities originating from the multitude of sensors and actuators of one or multiple robots which again can be controlled by various backends. UI frameworks are not allowed to enforce a specific robot hardware architecture nor a specific software architecture for the backend of the robot. In addition to that, UI frameworks have to cope with multiple middleware systems and communication protocols, as well as with the distributed nature of robotics applications itself, while still being easily extendable and simple to use.
4.4 The ROBOTUI architecture

Instead of solely introducing an implementation of an UI framework, emphasis is given to the description of an architecture. An implementation of this architecture will result in a clearly structured UI framework allowing the user to take full advantage of the benefits outlined in Section 4.3.1. The operating system, GUI toolkit and programming language independent nature of the architecture allows interested parties to implement their own UI framework based on their favorite combination of these. An implementation of the ROBOTUI architecture used for the monitoring of the developed UAV as well as for the analysis of the measured atmospheric CO\textsubscript{2} concentrations is discussed in Section 4.5.

4.4.1 Separation of concerns

The UI framework forms an additional layer on top of the backend deployed on a robot (see Figure 4.2). While the backend communicates with the hardware and implements the robot behavior, the UI framework is exclusively used for user interaction and the graphical representation of the internal and external state of the robot. The decoupling satisfies the separation of concerns and enables the parallel development and maintenance of the backend and UIs by independent groups.

The UI framework acts as an optional client to the backend. It subscribes to sensor data and delegates user triggered commands to the backend. The communication is always initiated by the UI framework. Due to the fact that the presence of a UI framework cannot be assumed during the runtime of the robot, the robotics behavior developer is enforced to implement the backend in a way that the internal state of the robot is always valid even though no UI framework might be available or the user does not respond in a specified amount of time. This solution therefore supports the developer indirectly by requiring him to think about these cases during the implementation of the robotics framework modules.

As long as the data provided by a hardware component is not used to trigger automatic changes in the behavior of a device managed by the UI framework, the system can be directly connected to the UI framework (see Figure 4.2, Class III). If however changes in the behavior of a system are intended based on the information provided, the system cannot be managed by the UI framework directly anymore (the separation of concerns would be violated, see also 4.3.2.2). In such cases the responsibility of the device management falls to a backend system.

4.4.2 The component structure

To monitor heterogeneous groups of robots efficiently, it is advantageous to be able to group UIs and to run multiple of these UI collections simultaneously. One can take the cooperation between a ground-based robot and an UAV as an example: While one group of UIs shows the position of both robots in a 3D visualization of the application environment and lets the user plan group related targets, a second UI collection can implement the GCS for the UAV while yet another group of UIs lets the user interact with the ground-based robot. The latter two UI collections provide detailed information about the settings and internal state of each robot which are not necessary for planning group related targets. If all information would be shown in a single UI collection, the user would
be overwhelmed by the amount of information provided. However, by putting the UIs into task-related groups, the complexity of the overall UI decreases and the user experience increases.

The ROBOTUI architecture specifies that the UI framework core provides the application window(s) and hosts UI collections. UIs and UI collections are represented internally as follows (see also Figure 4.3):

- **User interface components (UICs)** are encapsulated UI implementations. They can be implemented as classical graphical UIs but can also use other means to interact with the user, e.g. audio signals or speech.

- **Robotics user interface components (RUICs)** bundle and host a collection of UICs targeting a robotics related task.

A static, physical robot configuration is created by establishing parent-child relationships between individual devices. Each device is represented by an instance of the class DeviceEntity inside the UI framework. Figure 4.4 shows an example.

For devices of a robot driven by a robotics framework of Class I Category II, each DeviceEntity interacts through a robotics framework interface component (RFIC) with its corresponding module of the backend, which in turn interacts with either a physical sensor or actuator. Behavioral modules of the robotics framework which allow user inputs to change the behavior of the robot are seen as special devices by the UI framework, and are also represented by an instance of the class DeviceEntity (e.g. planner module in Figure 4.4).

RFIC instances run asynchronously and concurrently with each other as well as with the UI framework core, allowing a different update rate for each component instance. RFICs implement the communication protocol required by the backend. In addition, RFICs are responsible for the transformation of the robotics framework specific data structures and messages into UI framework specific data structures whenever such a
4.4. The ROBOTUI architecture

A DeviceEntity is responsible to manage all graphical representations (via a GraphicalRepresentationRegistry) of the device it represents (see Figure 4.3). It therefore builds a bridge between the RFIC responsible for the communication with the backend (model) and the graphical representations of the device (view) inside the UI framework. Graphical representations of devices are optional, e.g. a DeviceEntity representing a behavioral module may not have a graphical representation.

With multiple UIs presenting data of the same robot simultaneously it is beneficial to share the robot configuration between them, which then also ensures that only one connection to each robotics framework module is established. In addition, with only a single source of truth available throughout the UI framework, updates and changes are reflected throughout the UI framework consistently. The ROBOTUI architecture puts the UI framework core in charge of managing the robot configurations.

4.4.3 The registry

To extend the UI framework with new UIs as well as with support for additional backends is one of the core requirements of the UI framework. Depending on the combination of programming language and GUI toolkit chosen by the user for the implementation of the UI framework, a plug-in mechanism might be already available which can be utilized. Therefore, the ROBOTUI architecture does not impose a specific pattern for the implementation of the plug-in concept. A precondition for the extension of the UI framework via plug-ins is however that all extensions (e.g. graphical representations, UICs, RUICs, and RFICs) can be encapsulated into modules.

Ideally, the UI framework can be extended throughout its entire life cycle. To be able to access and instantiate all available extensions during runtime, a central registry is used. The management of the registry is task of the UI framework core. How the registry is filled with information is dependent on the implementation of the plug-in mechanism and the features provided by the programming language used for the implementation. The following sections will provide information about what data has to be put into the registry to allow the instantiation of needed UI framework components during runtime.
4.4.4 Creating RobotConfigurations and RUICs

A UI collection termed Home Screen is shown to the user after the start of a ROBOTUI implementation. The RUIC features at least two UICs, the Global Device Manager and the Scenario Manager.

The Global Device Manager provides access to all RobotConfigurations which have been defined in the UI framework. The UIC allows the user to define new or to remove outdated RobotConfigurations and to alter the settings of existing ones. A Robot-Configuration is necessary for each physical and simulated robot or device which should be managed with the UI framework. For each DeviceEntity of a newly created Robot-Configuration, the user has to select the backend used, and provide the necessary information on how to connect to the corresponding backend module of the DeviceEntity. This information is used to configure the RFIC of the DeviceEntity.

In order to instantiate the right RFIC for a DeviceEntity during runtime, it must be possible for the UI framework to retrieve the class information of a RFIC implementation from the registry using the identifier pair (deviceClassId, backendId). The user defines a DeviceEntity by selecting a DeviceType (e.g. SICK LMS200) and a backend (e.g. MSRDS). A RFIC however implements the data exchange with a module of a backend of a specific DeviceClass (e.g. laser range finder). Therefore the UI framework has to be able to retrieve the DeviceClass of the DeviceType selected by the user from the registry. To instantiate a RFIC based on the user’s selection, each DeviceType has to be registered with its corresponding DeviceClass, and each RFIC with its backend and DeviceClass in the registry. The information put into the registry is also available in the instantiated objects during runtime. This is reflected in the UML class diagram shown in Figure 4.5.

---

3The user has to select a DeviceType and not a DeviceClass in order to allow the UI framework to show the correct graphical representation of the device in its UIs.
4.4. The ROBOTUI architecture

The *Scenario Manager* allows the user to instantiate RUICs. RUIC implementations are made known to the registry with a name and a description which allows the user to select the right RUIC for a given task from a list of all available RUICs. With the exception of the *Home Screen*, all RUICs are used with a subset of the available *RobotConfigurations* defined in the *Global Device Manager*. The *RobotConfigurations* used with each RUIC are selected via the *Local Device Manager*. This UIC is available in each RUIC besides the *Home Screen*. A UIC can only access the *RobotConfigurations* used by the RUIC it belongs to.

4.4.5 UIC types and graphical representations of devices

The information provided by a *DeviceEntity* (through its RFIC) can be shown in multiple UICs simultaneously. Figure 4.6 shows an example: The UIC shown in the upper left part of the screenshot features a waypoint editor presenting the flight plan of a UAV. The UIC located on the upper right side views the same information in a 3D visualization of the application environment.

![Figure 4.6: A screenshot of the reference implementation discussed in Section 4.5 shows an implementation of the introduced ROBOTUI architecture: The application window provided by the UI framework core shows one RUIC (framed red) which consists of four UICs (framed black). The UIC Local Device Manager is part of every RUIC and allows one to select the *RobotConfigurations* which are used with the current RUIC. In this example, a single *RobotConfiguration* consisting of two *DeviceEntities* (helicopter, CO₂ sensor) is used.](image)

Three types of UICs are available. The waypoint editor is a representative of the first type, an information-centered UIC. In contrast, the 3D visualization is a graphics-centered UIC which represents the second type of UICs. The third UIC type includes all UICs which do not provide any graphical manifestation, but use other means to interact with the user (e.g. audio).
As soon as a DeviceEntity is selected by the user in the Local Device Manager, the latest information of its corresponding module in the backend is presented by all information-centered UICs of the current RUIC which support the type of data structure provided by the RFIC of the DeviceEntity. Usually, a form is used to implement an information-centered UIC.

Graphics-centered UICs show a collection of GraphicalRepresentations which visualize the DeviceEntities of the RobotConfigurations used inside the RUIC. Each graphics-centered UIC can be implemented using different libraries or frameworks, and therefore expect a GraphicalRepresentation of a specific type. This type is further referred to as ImplementationTechnology. A UIC registers itself with the ImplementationTechnology it expects in the registry. Graphical representations make themselves known to the registry with the DeviceType they visualize as well as with the ImplementationTechnology in which they are implemented.

One aim of robotics frameworks is to abstract the hardware and to allow behavioral modules to exchange data with sensors and actuators of the robot via device class specific interfaces. The application developer can therefore write algorithms dependent on device classes and does not need to know the specific type of sensor used on the robot at runtime. This allows behavioral modules to be used across robot configurations. Graphics-centered UICs of the UI framework show graphical representations of the sensors, actuators and robot platforms in use. Therefore, the device type has to be known during runtime in order to show the appropriate graphical representation of the device. This is the reason why the user defines a DeviceEntity by selecting a combination of Backend and DeviceType (and not DeviceClass) when creating RobotConfigurations in the Global Device Manager.

A GraphicalRepresentation is implemented with a specific ImplementationTechnology and represents a specific DeviceType. Each GraphicalRepresentation is instantiated via the central registry by a graphics-centered UIC using the identifier pair (DeviceType, ImplementationTechnology). The GraphicalRepresentation is then added to the GraphicalRepresentationRegistry of the DeviceEntity, from which it can be retrieved via the identifier pair (ruicId, uicId). A GraphicalRepresentation is only instantiated by the graphics-centered UIC if no instance is available in the GraphicalRepresentationRegistry for the combination (ruicId, uicId). A graphics-centered UIC can allow the user to configure its GraphicalRepresentations (e.g. change color). The configuration only applies to this UIC.

New information gained by the RFIC of a DeviceEntity is forwarded to all GraphicalRepresentations in the GraphicalRepresentationRegistry, allowing them to update their internal state. This also enables GraphicalRepresentations which depend not only on the current measurement but on a range of values (e.g. a graph) to keep their internal state up-to-date, even though they might not be shown at the moment (belong to an inactive RUIC).

The presentation of the current measurement inside GraphicalRepresentations of devices belonging to the same DeviceClass are usually very similar. One can take the visualization of a laser scan gained by two laser range finders of different types as an example: The source code used to create the visualization of the measurements (e.g. a series of rays with different length dependent on the distance measurements or a surface formed by the distance measurements) is identical, however the visualization of the housings of
the devices are different. To speed up the implementation of a GraphicalRepresentation for a new DeviceType of an existing DeviceClass, UI entities are used. An UI entity is an encapsulated building block which can be used by various GraphicalRepresentations or graphics-centered UICs implemented using the same ImplementationTechnology. They implement e.g. measurement visualizations for a DeviceClass or building blocks for UICs (e.g. an implementation of an image-based map or a coordinate system visualization). UI entities are not listed in the registry.

![Diagram showing the ROBOTUI architecture with UI toolkit element, UI entity, Graphical Representation, UIC, and RUIC.]  

Figure 4.7: Increasing granularity of code reuse based on encapsulated modules in the ROBOTUI architecture

Inside a ROBOTUI implementation, UI entities, GraphicalRepresentations, UICs, and RUICs form the building blocks for UI related source code reuse on a modular level. The building blocks are robotics domain specific. This is in contrast to GUI toolkit elements, whose granularity is very fine to allow them to be reused in applications of various domains (see Figure 4.7). The high-level code reuse inside ROBOTUI implementations based on encapsulated modules simplifies the development of new UIs for the robotics domain:

- **New UI collection:**
  A new RUIC can be created by bundling multiple UICs. While implementation dependent, it is possible that this is done without the need to write any source code.

- **New UI:**
  - Information-centered UIC:
    The UI can be build with an UI editor (if available for the chosen programming language and GUI toolkit in use).
  - Graphics-centered UIC:
    A graphics-centered UI implementation solely needs to provide support for GraphicalRepresentations of a specific ImplementationTechnology. Only a small amount of graphical components might have to be provided by the UIC itself (e.g. a map). This can be supported by the use of existing UI entities for the ImplementationTechnology of the UIC.

- **New graphical representation of a device:**
  New implementations can be based on existing UI entities for the ImplementationTechnology used, reducing the overall development effort.

In case a DeviceEntity has no GraphicalRepresentation for a specific ImplementationTechnology, its data can still be presented in information-centered UICs, but it will not be visualized in a graphics-centered UIC using this ImplementationTechnology. It is however not necessary for all DeviceEntities of a RobotConfiguration to provide a GraphicalRepresentation in order to view parts of a RobotConfiguration in a graphics-centered UIC.

---

4 The GUI toolkit used for the reference implementation allows UI collections to be defined via Extensible Markup Language (XML) files.
### 4.4.6 Serialization

The state of a ROBOT UI instance consists next to the RobotConfigurations defined via the Global Device Manager and the RUICs instantiated through the Scenario Manager of settings specific to individual UICs and GraphicalRepresentations which have been changed from their default values through user interaction. A problem regarding the serialization of the current UI framework state pose UICs and GraphicalRepresentations based on GUI toolkits and libraries which do not support the serialization of their components. To be able to save user defined changes (e.g. color changes) to such GraphicalRepresentations, each DeviceEntity provides a GraphicalRepresentationSettings-Registry. The structure of the registry is similar to the GraphicalRepresentationRegistry itself and allows each UIC to save and retrieve the current state of a GraphicalRepresentation using the identifier pair (ruicId, uicId). UIC specific settings which should be kept between UI framework sessions (e.g. view angle, zoom settings) can be stored by each UIC in a storage depot provided by its corresponding RUIC.

### 4.4.7 Coordinate systems, positioning of graphical representations of devices

The position is a key element in the state of mobile robots. Depending on the sensors deployed on the robot, it is likely that either a 2D position (x, y, heading), a 3D position (x, y, z, roll, pitch, yaw) or a GPS position (latitude, longitude, altitude, roll, pitch, yaw) will be provided by the sensors of the robot platform. The root DeviceEntity of a RobotConfiguration representing a mobile robot is the robot platform itself. It can therefore be assumed that the root DeviceEntity provides the current position of the RobotConfiguration through its RFIC. Furthermore, the root DeviceEntity defines the position type used by the RobotConfiguration. If the root DeviceEntity of a RobotConfiguration has not a position providing RFIC, the robot can assumed to be stationary.

Not all DeviceEntities receive data from their backend module including the measurement position. If for example the RobotConfiguration introduced in Figure 4.4 would be used in conjunction with P LAYER, the RFICs of the sonar array and the laser range finder would only provide distance measurements without the position at which they were taken. A GraphicalRepresentation of a DeviceEntity might however need to know the measurement position in order to show the visualization of the measurements at the right position inside a graphics-centered UIC. To overcome this problem, the parent-child relationship between the DeviceEntities of a RobotConfiguration can be used in conjunction with the relative offsets of the devices to each other, to show the delivered measurements at the positions of the devices derived from the position of the root DeviceEntity of the RobotConfiguration.

To provide the position information of each DeviceEntity of a RobotConfiguration to its corresponding GraphicalRepresentations depending on the position type used by the root DeviceEntity of the RobotConfiguration, PositionHandlers (see Figure 4.8) are introduced. Each position providing RFIC lists the type of the PositionHandler it uses (e.g. PositionHandler2d) in the registry. During the definition of a RobotConfiguration via the Global Device Manager, an instance of the PositionHandler registered for the root DeviceEntity is created for each of the DeviceEntities of the RobotConfiguration. With the PositionHandler being placed inside the DeviceEntity, both the RFIC as well as the GraphicalRepresentations of the DeviceEntity can access the
4.4. The ROBOT UI architecture

AbstractPositionHandler

- initialPosition : T
- currentPosition : T
- offsetToParent : U

+ T getInitialPosition()
+ setInitialPosition(Position : T)
+ T getCurrentPosition
+ setCurrentPosition(Position : T)
+ U getOffsetToParent
+ setOffsetToParent(Position : U)

Figure 4.8: The AbstractPositionHandler defines the interface for all PositionHandlers and implements basic functionality. Two template parameters are necessary to support GPS positions: While the position of the robot is a GPS position (e.g. World Geodetic System 1984 (WGS84) datum, type T), the offset of a device to its parent device will be provided in meters and radians (type U).

PositionHandler. The runtime instantiation is possible because each PositionHandler encapsulates the whole position management of a DeviceEntity into a single unit.

After the RobotConfiguration has been created, the user has to specify the initial position of the root DeviceEntity as well as the offset of each child DeviceEntity to its parent DeviceEntity. By making use of the parent-child relationship between the devices of a RobotConfiguration, the position of child devices can be set automatically by using the position of their parent DeviceEntity in conjunction with the relative offset to their parent device.

UI frameworks have to cope with a multitude of coordinate systems used by the supported backends as well as by the ImplementationTechnologies used for the implementation of the UICs. It should however not be necessary for the user to know specifics about these coordinate systems in order to operate the UI framework. The ROBOT UI architecture solves this problem by using three dedicated coordinate systems inside the UI Framework: A 2D coordinate system, a 3D coordinate system, and a GPS coordinate system. The coordinate systems specify the handling of all positions (including attitude and rotations) and can differ between UI framework implementations. All positions inside the UI framework presented to the user must be specified in either one of the three coordinate systems. This can be achieved by enforcing that all positions inside all PositionHandlers are specified in one of these coordinate systems.

The definition of UI framework internal coordinate systems allows the user to set the initial positions and offset positions of the DeviceEntities of a RobotConfiguration without any knowledge about the coordinate system used in the backend. In addition, with PositionHandlers having to provide the position in UI framework internal coordinates, coordinate transformations can be provided by the developers of the components knowing the specifics of each coordinate system: In case of a position providing RFIC implementation, the developer knows about the coordinate system used by the backend and can implement the coordinate transformation from backend coordinates to UI framework internal coordinates. In case of the implementation of a UIC or GraphicalRepresentation, the developer knows about the coordinate system of the device.

\footnote{Otherwise DeviceEntities of different type (e.g. DeviceEntity2d, DeviceEntity3d, DeviceEntityGPS) would be necessary.}
ImplementationTechnology used, and can implement the coordinate transformation from UI framework internal coordinates to UI coordinates. The N:M necessary coordinate transformations (from N backend coordinate systems to M coordinate systems used by the ImplementationTechnologies used for the implementation of UICs and GraphicalRepresentations) are split into N:1 and 1:M coordinate transformations.

A GraphicalRepresentation positions itself within a graphics-centered UIC. To do so, it expects position information of a specific type. Therefore, each GraphicalRepresentation lists the PositionHandler it expects its DeviceEntity to have in the registry. This is also the case for graphics-centered UICs. Graphics-centered UICs can determine based on the information from the registry which GraphicalRepresentation should be instantiated if the same ImplementationTechnology has been used to implement GraphicalRepresentations of a DeviceEntity with a different number of dimensions (e.g. a 2D and 3D visualization of a device has been implemented using the same ImplementationTechnology).

The introduced relationships between the modules of the ROBOTUI architecture ensure that RobotConfigurations are presented in graphics-centered UICs which provide a visualization of the application environment with the same number of dimensions than provided by the PositionHandler of their root DeviceEntity. A robot simulated in a 3D environment can however be monitored in a 2D graphics-centered UIC by using a RFIC which takes the 3D position provided by the backend and transforms it into a 2D position. The RFIC would list a PositionHandler2d in the registry.

If a position is provided with the measurement data by the backend module, GraphicalRepresentations should favor this position over the information provided by the PositionHandler of the DeviceEntity. The RFIC delivering the measurements has to implement the coordinate transformation from backend coordinates to UI framework internal coordinates and list the PositionHandler it uses in the registry.

The integration of a PositionHandler as a central element of a DeviceEntity raises the question how this move influences stationary robots which are also supported by the ROBOTUI architecture. Non-mobile devices which are not monitored in 2D or 3D visualizations of the application environment do also not provide any GraphicalRepresentations which are dependent on the position. For such stationary deployed devices, no PositionHandler will be used. If however a stationary device should be monitored within graphics-centered UICs of the UI framework, a PositionHandler providing the fixed position of the device can be used. A PositionHandler for the stationary device can be defined by the user via the Global Device Manager.

4.4.8 Communication between UI framework and backend

For robotics frameworks of Class I Category II, one DeviceEntity in the UI framework represents exactly one device on a physical or simulated robot. In addition, the data of the device is gained through a single RFIC from a single backend module. These 1:1 relationships allow the user to reproduce a robot in the real world or in a simulation in the UI framework by establishing parent-child relationships between individual DeviceEntities. The parent-child relationships in turn enable the automatic combination of the GraphicalRepresentations of the individual DeviceEntities of a RobotConfiguration to form a visualization of the RobotConfiguration itself, which can then be viewed inside graphics-centered UICs.
For robotics frameworks of Class I Category I, one connection to the robotics framework is enough to interact with multiple devices. If one would make use of this single-connection feature, a RFIC would have to be implemented for each RobotConfiguration. The RFIC would then provide data from all devices of the robot and would be represented by a single GraphicalRepresentation visualizing the whole RobotConfiguration. Such a course of action would be in stark contrast to the core ideas of the ROBOTUI architecture, enforcing source code development where it is not absolutely necessary. Therefore, ROBOTUI does not take advantage of the single-connection capability of robotics frameworks of Class I Category I. Instead, a dedicated communication channel between each DeviceEntity (through its RFIC) and its corresponding backend module (via the robotics framework core) is established (see Figure 4.9).

The same process to create RobotConfigurations can now be used by all backends of Class I. However, the RFIC configuration routines are different: For robotics frameworks of the Category II the user only has to provide the uniform resource locator (URL) of the robotics framework’s central registry to the UI framework. The user can then select the corresponding robotics framework module for each DeviceEntity from a list of possible modules retrieved from the central registry of the robotics framework. The missing robotics framework module registry for robotics frameworks of the Category I makes it necessary for the user to configure the RFIC with all the information necessary to connect to its corresponding robotics framework module (e.g. PLAYER: host, port, deviceld).

Figure 4.9: Neglecting the single-connection capability of robotics frameworks of Class I Category I (e.g. PLAYER) allows the creation of robot configurations without the need to write any source code.

Figure 4.10: A CompositeRFIC enables the UI framework to support backends of Class II.
Chapter 4. Modular UI framework

The solution provided for backends of Class I, Category I cannot be used to accommodate backends of Class II. The software framework on board of the robot does not allow one to connect to each component individually but only offers a single communication link for the whole system. To cope with this problem, the concept of a \texttt{CompositeRFIC} is introduced: A \texttt{CompositeRFIC} uses the dedicated communication link to exchange information with the backend while offering a RFIC for each inaccessible device on the robot to the UI framework layers III.B and III.C (see Figure 4.10). This enables the user to apply the same process to define \texttt{RobotConfigurations} than for robots controlled by a backend of Class I, and allows the same strategy to be used by the UI framework to create a \texttt{GraphicalRepresentation} of the robot during runtime.

The only task of each \texttt{CompositeRFIC} is to split the information gained through the dedicated communication channel to the provided RFICs and vice versa. To support a robot of Class II, the developer has to implement the \texttt{CompositeRFIC} and its RFICs. A minor overhead is also necessary for the UI Framework users who have to first instantiate and configure the \texttt{CompositeRFIC} with the connection details to the backend, but can then use the already introduced method to create \texttt{RobotConfigurations} as usual.

Theoretically, the solution introduced for robots controlled by a backend of Class II can also be applied to robots driven by a robotics framework of Class I, Category I. Nevertheless, the developer would have to create a \texttt{CompositeRFIC} for each \texttt{RobotConfiguration} - a disadvantage which by far outweighs the benefits of a single connection to the robotics framework core.

It should be noted that the introduced solution allows the user to create \texttt{RobotConfigurations} with each physical or simulated device being powered by a different backend.

4.4.9 Device interfaces and data structures

To ensure the maximum reusability of RFICs, UICs, and \texttt{GraphicalRepresentations} as well as to enable their transparent exchange, the following requirements must be met:

1. Internal states must stay encapsulated inside the components themselves and must never be passed on to other components.

2. Data structures used to exchange information between the components should be generic and use standardized units (SI units) whenever possible. The amount of data structures should be kept as small as possible. Only when the data structure provided by a RFIC is identical, or an extended version of the data structure expected by an UIC or \texttt{GraphicalRepresentation} (and vice versa), a direct connection between the components can be setup.

3. A component has to clearly define its interface. Only if the interfaces of two components is identical, or the exchanging component provides an extended interface of the one provided by the component to be exchanged, a transparent swap of the components is possible.

The first two rules cannot be enforced by programming languages or frameworks, but are still essential to maximize the reusability of the components.

The problem of clearly defining component interfaces is well known in robotics frameworks that are based on modular architectures. Different ideas are implemented in these to cope with the problem: To be able to transparently exchange modules that solve the
same task in different ways, PLAYER uses interfaces (generic specifications of devices) [130]. The data structures used to exchange information with the various interface implementations (these data structures are further referred to as communication objects) are specified in the interface definition. Thus, the interface creator specifies the communication objects, shifting the responsibility away from the user. Unfortunately, due to the generic nature of the interface, features and functions unique to a specific interface implementation are ignored.

SMARTSOFT has a different idea to cope with the problem: Each component describes its own interface and specifies the communication objects to be used. Predefined communication objects are available and used extensively by the components provided by the framework. Due to the object oriented nature of the data structures, it is possible to derive and extend existing communication objects without losing the generic compatibility. Unfortunately, users still have the possibility to create contradicting communication objects (same information content but different type). Such action prevents the transparent exchange of otherwise exchangeable components.

MSRDS services describe their interfaces using contracts. Other services can implement the same contract if they wish to be transparently exchangeable with other implementations of the contract. It is possible to extend existing contracts if additional functionality is provided by a service but the compatibility to the original contract should be kept. Microsoft provides a set of generic service contracts one can implement and extend. Users can also define their own, possibly contradicting contracts for their services.

While PLAYER and SMARTSOFT use programming language and middleware specific means to describe the interfaces of their components, MSRDS introduces a programming language independent interface description. That is necessary to enable the creation of MSRDS services in all available .NET languages. The ROBOTUI architecture does not specify an interface description language for ROBOTUI components on a programming language independent basis. Most programming languages provide means to clearly describe a component interface. An exchange of components of different ROBOTUI implementations is unlikely due to various incompatible GUI toolkits based on different programming languages. The introduction of a mandatory interface description language for all ROBOTUI implementations does therefore not gain any advantages.

4.5 Reference implementation

For the selection of the programming language and GUI toolkit used for the reference implementation, high emphasis was placed on finding a combination of tools which provided most of the features necessary to implement the ROBOTUI architecture. In addition, the way how the components of the architecture (RFICs, UICs, RUICs) can be encapsulated played a significant role in the selection process.

The decision was made to base the reference implementation of the ROBOTUI architecture on the Eclipse Rich Client Platform (ERCP) [131] which can be extended and customized using Java. The decision was made based on following reasons:

- ERCP is based on Equinox, which is the reference implementation for the Open Service Gateway initiative (OSGi) [132] framework specification. The standardized OSGi module runtime supports the encapsulation of RFICs, UICs and RUICs on multiple levels: Each ROBOTUI component can be implemented as an individual
OSGi bundle that is represented by an individual implementation project. In addition, each OSGi bundle is packed as a .jar file on the file system. The component structure of the ROBOTUI architecture is therefore recognizable throughout the whole development and deployment process.

- The OSGi framework supports the developer during the implementation process of the UI framework components: It defines a modularization model for Java (Module Layer), introducing rules which enable the developer to select which contents can be seen by other bundles. A life cycle API for bundles is also provided (Live Cycle and Service Layer). In addition to that, dependencies between bundles can be described formally and version information can be attached to bundles.

- Multi-platform support due to ERCP and Java.

- Available OSGi modules can be incorporated into the reference implementation.

- The Eclipse integrated development environment (IDE) offers strong support for the development of ERCP projects through a set of plugins designed to simplify the management of OSGi and ERCP projects. The Eclipse IDE is widespread and available on many platforms. In addition, a GUI builder plugin supporting the Standard Widget Toolkit (SWT) which forms the basis of ERCP, is available for the Eclipse IDE.

- Libraries for the implementation of 2D and 3D representations of application environments and visualizations of devices are available for Java (e.g. Draw2d, Lightweight Java Game Library (LWJGL), Nasa World Wind). The libraries can be integrated into ERCP which enables the implementation of UI entities, Graphical Representations and UICs using different Implementation Technologies, which is supported by the ROBOTUI architecture.

- ERCP provides a strong plugin model, allowing the reference implementation to be easily extended with new components, Graphical Representations, and UI entities. In addition, the plugin model allows developers to attach their copyright to individual plugins. No license model is enforced by the UI framework core, enabling developers to pick a suitable license for their plugins.

- Even though most robotics frameworks are implemented in C or C++, the most commonly used robotics frameworks either provide Java support (PLAYER, ROS, CARMEN) or are easy to interface with Java (MSRDS).

- ERCP uses a registry to manage extension points. The registry gains information from specific XML files residing inside the plugin projects. This registry can also be used as the ROBOTUI registry, allowing the existing tools for the ERCP registry to be used by the reference implementation as well.

Due to its dependency on ERCP, the reference implementation is called ROBOTUI ERCP.

### 4.6 Example scenarios

In this section, three scenarios are introduced which show the ROBOTUI reference implementation being used for diverse robotic monitoring tasks. Each example is highlighting the challenges of a particular class of robots (as introduced in Section 4.3.2.4) onto the
UI framework. Special emphasis is given Example II (Section 4.6.2) which represents the GCS for the developed helicopter UAV.

4.6.1 Monitoring of two simulated robots using two backends of Class I

4.6.1.1 Description

This scenario is designed to show that it is possible to monitor multiple robots simultaneously in the UI framework even though they are driven by different robotics frameworks. While monitoring the robots, the user is unaware that the robots are controlled by different backends.

The two robotics frameworks which are used in this scenario are PLAYER (Class I, Category I) and MSRDS (Class I, Category II). MSRDS is a commercial product and forms a nice contrast to PLAYER, an open-source robotics framework. Both MSRDS and PLAYER are available since several years and are well recognized in the robotics community. The two robotics frameworks are used in conjunction with their respective simulations (MSRDS’s built-in 3D simulation and PLAYER’s 2D simulation STAGE\(^6\)), each of them controlling one robot. MSRDS is run on Windows, while PLAYER and STAGE is run on Linux.

The robot controlled by PLAYER is a ROOMBA 500 SERIES equipped with a HOKUYO URG-04LX laser range finder. The second robot controlled by MSRDS is a PIONEER P3DX including 16 sonar sensors and a SICK LMS200 laser range finder on top of the robot base. Both robots are simulated in the same environment, however by different simulations.

4.6.1.2 Framework internals

ROBOTUI ERCP makes extensive use of abstract classes and Java Generics in the implementation of the UI framework core (ERCP plugin org.robotui.core) which implements the ROBOTUI component structure. That simplifies the implementation of new components by forcing the developer to implement certain methods while basic functionality is provided by the implemented methods of the superclasses.

For the described scenario, three device classes had to be introduced: laser range finder, sonar array, and position velocity 2d. The RFIC interface of a device class (e.g. class AbstractLaserRangeFinder) and the device class specific communication object (e.g. class CommLaser) are defined together in an ERCP plugin (e.g. org.robotui.-spec.device.laserrangefinder). Furthermore, each device class is made known to the registry from its defining plugin.

Five RFICs had to be implemented for the scenario (see Figure 4.11). Each of the RFICs is encapsulated in an individual ERCP plugin (org.robooui.rfic.backend.deviceclass). PLAYER provides a Java library to communicate with the robotics framework core which simplifies the RFIC implementations for this backend considerably. Libraries are also packed into ERCP plugins (org.robotui.library.libraryname).

The Backends (MSRDS, Player) are defined and made known to the registry in independent ERCP plugins as well. These plugins (org.robotui.spec.backend) also include classes which are shared between all RFICs of one Backend, e.g. the RFIC configuration wizard for the specified Backend (see Figure 4.12).

\(^6\)A 3D simulation for PLAYER is available (GAZEBO [133]) which is however not being used in this work.
Figure 4.11: The use of Java Generics and abstract classes allows basic functionality to be provided to all subclasses while enforcing the developer to implement certain methods. ROBOTUI ERCP defines the basic UI framework structure in the plugin org.robotui.core. A new device class, its interface and communication object are defined together in a plugin (org.robotui.spec.device.deviceclass). The implementations of RFICs for a specified combination of device class and backend form separate plugins (org.robotui.rfic.backend.deviceclass). For the introduced scenario, three device classes had to be specified and five RFICs had to be implemented.

Figure 4.12: Configuration wizards for MSRDS (left) and PLAYER (right). The configuration wizards for all RFICs of one Backend are identical and are therefore packed together with the backend registry specification into an ERCP plugin (org.robotui.spec.backend).

Splitting the component definitions and their implementations into separate packages allows new components to be added without the need to customize or extend any existing ERCP plugins. In addition to that, the naming convention used by ROBOTUI ERCP for the ERCP plugins gives the user an overview of already specified devices and their available implementations. By sticking to this convention, it is also unlikely that the user gets caught in circular dependencies of ERCP plugins.

Figure 4.13 shows the RUIC implemented for the monitoring of robots executing tasks in a 2D environment. The UI collection uses six UICs (Local Device Manager, Measurement
4.6. Example scenarios

Figure 4.13: Screenshot of ROBOTUI ERCP monitoring two simulated robots controlled by different robotics frameworks simultaneously in a single graphics-centered UI. The fact that the robots use different backends is not evident for the user.

Viewer, Error Log, RobotUI Log, Robot Monitoring, SWTXYGraph). The Error Log is an ERCP plugin native to Eclipse which is reused by ROBOTUI ERCP to give information about unexpected errors in plugin implementations. The RobotUI Log shows notifications and messages from the UI framework core and other ROBOTUI ERCP plugins. The Measurement Viewer is an information-centered UIC and shows the latest measurement of the currently selected device in the Local Device Manager. As described before, the Local Device Manager is used to add or remove robot configurations to the list of devices which should be monitored with the RUIC. It is available in every RUIC except the Home Screen RUIC. Both the Robot Monitoring and the SWTXYGraph UIC are graphics-centered UICs. This section focuses on the Robot Monitoring UIC, while the SWTXYGraph UIC will be covered in a later example.

The Robot Monitoring UIC features an image-based map, a coordinate system as well as a scale. The latter two can be shown or hidden via buttons on the UIC toolbar (the scale is hidden in the screenshot). The three components are UI entities which provide basic features of the UIC. The other elements are GraphicalRepresentations of the devices managed by the Local Device Manager. UICs listen to changes in the list of managed robot configurations and update their information accordingly. This means for graphics-centered UICs that new GraphicalRepresentations have to be instantiated via the registry based on the information provided by a RobotConfiguration added to the list, or the removal of GraphicalRepresentations if a RobotConfiguration is deleted from the list. Figure 4.14 gives an overview of the GraphicalRepresentations which had to be implemented for the scenario.
Figure 4.14: Five graphical representations had to be implemented for the scenario. The graphical representations of the laser range finders use both a UI entity (LaserMeasurements) which displays the current laser scan. The implementation technology of the graphical representations and the UIC is Draw2d.

Figure 4.15: The Home Screen RUIC. This is the place where the user can create robot configurations and instantiate UI collections. Device settings (communication settings of the RFIC, initial position, offset to parent) can be edited in the RUIC as well.
Figure 4.15 shows the Home Screen RUIC. This RUIC belongs to the UI framework core and is shown right after the start of ROBOTUI ERCP. It features the Global Device Manager as well as the Scenario Manager. One can see that one RUIC has been created with the Scenario manager (robotics framework test) which can be seen in Figure 4.13. While new robot configurations can be created via the Global Device Manager, the Properties UIC (a native ERCP plugin) can be used to re-configure the RFICs as well as the position parameters (either initial position or the position offset to the parent device) of the devices. The position information is managed by the PositionHandlers of the DeviceEntities (here: PositionHandler2d).

The scenario shows that it is possible to monitor and manage multiple robots with various configurations driven by different robotics frameworks together in a single UI, without the user having to know any specifics about the backends deployed on the robots. In addition, the example shows the steps necessary to extend the UI framework with new backend connections as well as new robotics related UICs.

4.6.2 Implementation of the GCS for the developed CO$_2$ sensing UAV

4.6.2.1 Description

In this scenario, the UI framework reference implementation is used to create the GCS for the developed UAV. The GCS is comprised of UICs to specify the path to be flown, to change the parameters used by the autopilot software (e.g. PID gains), to visualize the UAV position and CO$_2$ concentration in a 3D representation of the application environment, and to monitor the battery voltages of the aerial robot.

The scenario shows that devices of Class II can be incorporated into the UI framework with only little overhead compared to the devices of Class I. In addition, the use of an audio UIC telling the GCS operator about flight pattern changes shows that non-visual alternatives to interact with the user can also be implemented with ROBOTUI ERCP.

4.6.2.2 Framework internals

The developed UAV offers the user a single communication link through which its state can be monitored, autopilot settings and parameters can be changed, and commands can be sent to the helicopter. The sensors and actuators of the aerial platform cannot be directly accessed by the user through the communication link. The developed UAV is therefore a representative of a robot of Class II.

In order to integrate the robot into the UI framework, a CompositeRFIC is necessary. A CompositeRFIC manages the communication between the UI framework and the robot while exposing RFICs representing the devices deployed on the robot towards the UI framework layers III.B and III.C.

To implement a CompositeRFIC, the developer has to first decide which devices of the platform he wants to make known to the UI framework and the end user. One can for example expose the laser range finder deployed on the UAV in order to show the device itself as well as its measurements in a graphics-centered UIC. However, only a single measurement of the laser range finder is accessible through the communication link which coincides with the elevation of the UAV. One might therefore define the helicopter as a single device which provides a position (latitude, longitude, elevation above
ground, yaw, pitch, roll), ignoring the number and types of sensors which are responsible to provide this information.

In ROBOT UI ERCP, a new device class has been introduced for helicopter platforms based on the implemented autopilot software system, which will be further referred to as Autonomous Helicopter Framework (AHF). Each device type of the introduced device class (AHF helicopter) represents a helicopter which provides a GPS position, the relative elevation of the UAV above ground, the platform’s velocities, and is able to receive waypoints and autopilot parameter adjustments from the user. In addition, information about the currently executed FlightPattern and information about the battery voltages are provided. All these features have been combined into a single device class because they are all necessary for the basic operation and monitoring of a helicopter UAV using the introduced autopilot system.

The communication object (CommAHFHelicopterState) of the device class AHF helicopter only relies on a single GPS position and elevation. For the CO\textsubscript{2} sensing UAV, the basic communication object has therefore been extended (CommTRex700EState) to also include the GPS position and elevation gained by the backup sensors. In addition, details about which sensor is currently used for the position estimation, as well as information about the task states of the autopilot are part of the extended communication object. To expose the CO\textsubscript{2} sensor to the upper layers of the UI framework, the CompositeRFIC defined for the device class AHF helicopter has been derived and extended for the CO\textsubscript{2} sensing UAV. Figure 4.16 shows the CompositeRFIC class hierarchy while Figure 4.17 shows the Global Device Manager with the fully configured RobotConfiguration.

Figures 4.18 and 4.19 show screenshots of the UI collection used for the monitoring of the developed helicopter UAV. From the ten UICs used in the RUIC, two information-centered UICs (AHF Waypoints and AHF Helicopter Parameters) have been specifically developed to plan the flight path and to adjust the UAV parameters of an AHF controlled helicopter. One information-centered UIC has been specifically implemented for the communication object CommTRex700EState (TRex700E Overview), providing an overview of the key elements\textsuperscript{7} of the helicopter state. Helicopter configurations using the same or an extended version of the communication object can reuse the UIC.

The World Wind Viewer UIC is a graphics-centered UIC and presents GraphicalRepresentations of the ImplementationTechnology Nasa World Wind. The GraphicalRepresentation of the CO\textsubscript{2} sensing UAV consists of multiple layers: A layer showing the coordinate system used to define relative waypoints, a layer showing the planned flight path, a layer showing the path flown by the UAV, a layer showing the UAV at its current position\textsuperscript{8}, and a layer showing the CO\textsubscript{2} measurements. Each layer can be shown or hidden by the user. The World Wind Viewer is a general purpose UIC.

One of the UICs is not shown in the screenshots. The plugin org.robotui.uic.marytts provides text to speech support via the library OPENMARY. The UIC can be used to give verbal feedback to the user. In the case of the GCS, the UIC is used to notify the operator about flight pattern changes.

\textsuperscript{7}The UI provides an overview of the positions delivered by the primary and secondary sensors, indicates which of the sensors is currently used by the autopilot, and gives details about the flight pattern in use as well as which step is currently executed in the autopilot if the measurement flight pattern is executed.

\textsuperscript{8}A line with an arrow indicating the forward direction of the UAV is used instead of a 3D model of the helicopter. This simple representation has been found more suitable during the conducted experiments than the use of a generic 3D model of a helicopter.
4.6. Example scenarios

Figure 4.16: The diagram shows the class hierarchy used in ROBOTUI ERCP to support helicopter UAVs using the introduced autopilot system. The implementation provides basic support common for all helicopter UAVs controlled by AHF through the components implemented in the plugin org.robotui.spec.crfic.ahf. Different robot configurations with various payloads can build on the provided components and communications objects. This has been done for the CO₂ sensing helicopter UAV (plugin org.robotui.crfic.anu.trex700e).

Figure 4.17: To monitor the developed CO₂ sensing helicopter UAV controlled by AHF, the user has to first instantiate its CompositeRFIC via the Global Device Manager and configure its communication settings (see Properties UIC). The RFICs provided by the CompositeRFIC are shown in the Global Device Manager as children of the CompositeRFIC. After this step, the user can create robot configurations as usual.
Figure 4.18: GCS screenshot presenting the (AHF) Waypoints UIC as well as the World Wind Viewer UIC.

Figure 4.19: GCS screenshot showing the TRex700E Overview UIC as well as the (AHF) Helicopter Parameters UIC.
4.6. Example scenarios

The autopilot system delivers each CO$_2$ measurement with its corresponding measurement position. The matching of the CO$_2$ concentration with the current position of the UAV inside the autopilot results in a better measurement position estimate than it would be the case if the position and CO$_2$ measurement values are matched together in the UI framework. Due to the fact that a position is delivered with the CO$_2$ measurement, the PositionHandler of the DeviceEntity is not being used by the GraphicalRepresentations of the device.

4.6.3 Temperature monitoring

4.6.3.1 Description

In this scenario, the reference implementation of the ROBOTUI architecture is used to visualize the measurements delivered by a temperature sensor array monitoring a chemical reaction. The sensor measurements are used by the operator to check the progress of the reaction. Based on the presented data, the user might influence the reaction via manual inputs to the experimental apparatus. The measurements are not used to automatically trigger changes influencing the chemical process. The sensor array is stationary and a member of the devices forming Class III.

4.6.3.2 Framework internals

Members of Class III form a special group of devices inside the UI framework. The RFICs for these devices do not communicate with a module of a backend but implement the device protocol itself. In this example, the values of all temperature sensors of the array are read by the RFIC via a virtual COM port interface from a single MCU board, which gains the temperature measurements periodically from the K-type thermocouples of the array. The RFIC belongs to the newly created device class temperature sensor array which uses the communication object CommTemperatureArray.

The progress of the chemical reaction is monitored using a graphics-centered UIC which views GraphicalRepresentations using the ImplementationTechnology SWT-XY-Graph (see Figure 4.20). The GraphicalRepresentations viewed by the UIC are all based on the library SWT-XY-GRAPH which provides graph drawing support for SWT applications. The UIC itself only provides a container to view the GraphicalRepresentations in addition to a drop-down list enabling the user to select an available graph. Next to the SWTXYGraph UIC, the information-centered Measurement Viewer UIC is used to present the user the latest measurements in a table.

The SWTXYGraph UIC has also been used in the first two scenarios to present graphs showing the velocities of the mobile robots, the voltage levels of the on-board UAV batteries, the number of GPS satellites used by the GPS sensors on-board the AHF controlled helicopter, the CO$_2$ concentrations, and the UAV velocity over time. The Measurement Viewer UIC has also been used in the previous two scenarios to present the latest data received by the RFIC of the DeviceEntity selected in the Local Device Manager.

The example shows that it is possible to directly connect devices without a backend to the UI framework. It is however important that this opportunity is not used in violation with the separation of concerns: behavioral components are not intended to be implemented inside the UI framework but in a software framework of Layer II. Solely measurement data presentation and user interaction is task of the UI framework.
Figure 4.20: Screenshot showing ROBOTUI ERCP being used to monitor a chemical process. The SWTXY-Graph UIC shows the temperature development at three positions around the measurement apparatus. The temperatures are measured via a temperature sensor array controlled by a MCU board.

4.6.4 Summary

The examples show that the reference implementation of the ROBOTUI architecture can be used for the robot management and sensor data representation and analysis of measurements gained by devices using various backends solving different robotics related tasks. It can be also seen that UICs can not only be used to visualize measurement data of multiple RobotConfigurations simultaneously, but that UICs can also be reused for diverse robotics related tasks:

- The information-centered Measurement Viewer UIC can be used independently of the task to be solved to show the latest measurement gained by the RFIC of the selected DeviceEntity in the Local Device Manager UIC. In addition, the graphics-centered SWTXYGraph UIC can be used to present graphs independent of the task of the robot.
- The graphics-centered World Wind Viewer UIC allows sensor data to be viewed at the position it has been measured in the real world in its 3D world visualization. The positions of mobile robots providing their GPS position to the UI framework can also be tracked using this UIC. The World Wind Viewer UIC also allows the presentation of very diverse measurement data with the use of rigid shapes, surfaces, and images.

The scenarios also show what development work is necessary if one wants to monitor robot configurations using backends which are not yet supported by the UI framework, and no GraphicalRepresentation is available inside the UI framework for any of the device types used. The examples can therefore be seen as the worst case scenario regarding the implementation workload. With the ongoing use of the UI framework with
4.7 Conclusions

The CO\textsubscript{2} sensing helicopter UAV represents one part of a comprehensive measurement suite for onshore geological CO\textsubscript{2} storage sites. In order to control and monitor all sensors, sensor networks and robots from a single application, and to allow a measurement data analysis based on the data collected by all individual elements of the measurement suite, the development of a modular UI framework was suggested.

The introduction of dedicated UI frameworks allows individual UIs as well as UI collections to be shared by various backends. In addition, devices driven by different software systems can be monitored together in a single UI. These features are necessary in order to be able to incorporate the possibly large number of diverse sensor and robotics systems (from various vendors) forming a monitoring suite for commercial CO\textsubscript{2} storage sites into a single application.

With the provided implementation of the ROBOTUI architecture, the monitoring application used for a commercial CCS project could look as follows:

- A UI collection shows the position of all sensors, sensor networks and mobile robots used for the monitoring of the onshore CO\textsubscript{2} storage site in a 3D visualization of the storage area (e.g. using the World Wind Viewer UIC). Borehole positions and injection wells are shown as well.
- By selecting an individual sensor, sensor network or mobile robot, the user is provided with the choice to load a UI collection to
  - configure the selected sensor, sensor network, or to control and monitor the behavior of a selected mobile robot or to
  - see a timeline of the measurements provided by the device (e.g. using the SWTXYGraph UIC).
- The latest measurements gained by all individual devices of the measurement suite can be visualized in real-time in a 3D representation of the storage area. This allows the user to detect hot spots quickly and enables one to send mobile robots (including the developed UAV) to the areas in question to retrieve more detailed measurement values.
- Data produced by measurement techniques which do not provide their measurements in real-time can be incorporated into the monitoring application via RFICs that interface to databases that hold the measurement data.

The use of dedicated UI frameworks is not only helpful for the task at hand, but also beneficial for other robotics applications in which devices from various vendors should be used to solve a single task. With a UI framework, these devices can be monitored side by side without the user being aware of the different software systems deployed on the
robots. Even for scenarios where only a single robot is used with a single backend, the use of a UI framework can help preventing the re-implementation of necessary UIs which are not provided by the backend in use. Furthermore, a UI framework can be used to view sensor data gathered by different sensors from various vendors without any actuators or robots.

The introduced modular ROBOTUI architecture describes (independent of programming language and GUI toolkit) a solution of how to build modular UI frameworks which maximize the reuse of UI framework components (RFICs, RUICs, UICs, GraphicalRepresentations, UI Entities) inside the implementation. While UI frameworks based on the ROBOTUI architecture cannot ensure any timing constraints (due to a user in the loop / sporadic user interaction), they still can be used to monitor and interact with both soft and hard real-time systems as long as these provide an interference free read access.
Conclusions

Chapter summary and structure

The previous chapters introduced a small helicopter UAV for the monitoring of onshore geological CO₂ storage sites, discussed the design of an autopilot for beyond line-of-sight operations, introduced a new TDOA-based acoustic localization system to gain the accurate position of the UAV during landing approaches, and showed how modular UI frameworks can be used to incorporate the management of all systems of a heterogeneous measurement suite consisting of sensors, sensor networks and mobile robots from various vendors into a single application. This chapter summarizes the contributions of this thesis and outlines further research directions aimed to make UAV-assisted monitoring of onshore geological CO₂ storage sites a reality in the close future.
5.1 Overview

To enable UAVs to be used to assist the monitoring of onshore geological CO\textsubscript{2} storage sites, one first has to understand the requirements the task poses onto an aerial measurement platform. These requirements then have to be specified and accommodated in the design of the UAV platform. Furthermore, with a variety of sensors already available at CO\textsubscript{2} storage sites, it must be possible to integrate the results of the UAV-based monitoring to the existing measurement suite to allow site operators to get the most of the overall collected data.

This thesis discussed three almost orthogonal problems which however all need to be solved in order to allow UAV-based CO\textsubscript{2} monitoring of onshore geological storage sites to get reality: The design of a suitable aerial measurement platform with a robust UAV autopilot, high accuracy position estimation for UAV landing maneuvers, and the integration of sensor data from multiple heterogeneous sources into a single view. The following section will summarize each of the contributions in more detail.

5.2 Summary of the contributions

UAVs were identified as one possible, economical solution to detect containment breaches in geological CO\textsubscript{2} storage sites by measuring the atmospheric CO\textsubscript{2} concentration or the amount of CO\textsubscript{2} tracers in the air above the storage formation. It was discussed that beyond line-of-sight flights are necessary for the monitoring of the possibly large area over the storage formation. The properties of the CO\textsubscript{2} sensor in use were then identified as key factors for choosing a suitable robot platform. A helicopter UAV was found to meet the necessary platform requirements while providing the possibility to exchange sensors and to switch the size of the UAV platform itself with the least amount of efforts.

A general purpose autopilot for non-acrobatic flight maneuvers for helicopter UAVs was then introduced. The design focuses on sensor flexibility and reliability while providing the user the possibility to implement new flight patterns in a hardware independent manner. The autopilot system assumes an electronic helicopter stabilization system on-board the aerial platform which is capable of stabilizing the UAV in a level posture. Such systems are commonly used by RC pilots to simplify the control of their helicopter models. The stabilization system can also be used by a pilot to fly the UAV radio controlled to test changes to the robot platform. Furthermore, a motor controller providing a governor mode to keep the head speed of the helicopter constant as well as a system providing the global position and the attitude of the UAV are necessary for the autopilot system. There are no further requirements for the motor controller, positioning system and stabilization system (e.g. specific models). The helicopter platform itself as well as all sensors, including the ones used for the position estimation of the UAV, can be freely chosen by the developer. The autopilot grants the developer access to the control loop which is necessary for some extensions, e.g. obstacle avoidance. Furthermore, sensor redundancy can be implemented with the autopilot, which is a necessary safety feature for beyond line-of-sight operation of the helicopter UAV.

The development of an aerial robot platform based on the ALIGN TRED 700E RC helicopter model was then discussed. The previously introduced autopilot system was used with the aerial platform and a VAISALA GMP343 CO\textsubscript{2} probe to form a prototype
5.2. Summary of the contributions

Aerial CO$_2$ measurement system. The introduced UAV was then used in a feasibility study which showed that the setup can be used to detect point source CO$_2$ surface leaks of 100 kg of CO$_2$ per day. Furthermore, it has been shown that it is necessary to fly the developed UAV close to the ground (~1.5 m) in order to detect CO$_2$ surface leaks of this magnitude. During the limited flight time of the helicopter UAV, the discussed setup allows one to monitor an area of approximately 750 m$^2$ in one elevation with a spatial resolution of 2 m by 2 m. With the developed aerial platform being a prototype system, one can expect that even larger areas can be covered with an aerodynamically optimized, weight-reduced redesign of the UAV.

To gain the position of the UAV accurately enough to enable autonomous landings of the aerial platform on a landing pad of the size of approximately 1 m by 1 m, a passive acoustic TDOA-based localization system was proposed. The introduced two approaches were designed for planar sensor arrays which can be integrated into the landing pad without any parts protruding out of the helipad which would pose a threat for the UAV to get caught in. Planar sensor arrays consisting of at least four microphones can be used with the developed methods. The presented algorithms split the positioning problem in two parts: First, the $x$ and $y$ position of the acoustic source relative to the sensor array is calculated which is then used in a second step to gain the elevation of the source. The influence of the microphone array geometry onto the accuracy of the position estimates was discussed, and application specific array geometries beneficial for the helicopter UAV localization task introduced. Furthermore, a trust value for the position estimates which considers the layout of the microphone array was derived. Simulations show that the developed approaches perform equally or even outperform their direct competitors in the targeted task. A conducted indoor experiment showed a median localization error of $\leq 53$ mm if the acoustic source is kept within a radius of 3 m from the center of the sensor array. Furthermore, an outdoor experiment confirmed that it is generally possible to localize a helicopter UAV based on its intrinsic sound in its landing area with the developed approaches as long as quality TDEs are available. For reliable and accurate TDEs (a prerequisite for the introduced approaches) an artificial sound source should however be utilized.

In order to make full use of all sensors, sensor networks and mobile robots forming a comprehensive CCS monitoring suite, it is beneficial to be able to monitor all gathered data and to manage all robots from a single application. This poses a challenge in heterogeneous environments, where systems from different vendors using various software systems can be expected. To solve this problem, dedicated UI frameworks were introduced. This step clearly separates the implementation of the robot behavior (model) from the graphical representation of the robot and its measurements (view) into two dedicated software applications. The user benefits from this by only having to interact with a single application to gain measurement data from the whole monitoring suite and to control all mobile robots independent of their on-board software system and hardware configuration. Furthermore, robotics related UIs can be reused across robots driven by various software frameworks which can reduce the implementation effort considerably. A software architecture describing in a platform and GUI toolkit independent way how such UI frameworks can be implemented was introduced. Finally, a reference implementation was used to show the feasibility of the introduced approach in three robotics related example scenarios, including a GCS implementation for the developed helicopter UAV.
Chapter 5. Conclusions

5.3 UAV-assisted aerial CO₂ monitoring

It was shown that it is possible to use a helicopter UAV to measure the atmospheric CO₂ concentration over onshore geological CO₂ storage sites. CO₂ surface leaks which would be considered very small in the context of naturally occurring CO₂ seeps can be detected if the aerial platform is flown close to the ground (~1.5 m). Furthermore, the discussed helicopter UAV is capable to monitor an area of approximately 750 m² in one elevation with a spatial resolution of 2 m by 2 m.

These contributions were made possible by making the CO₂ sensor the primary factor of the whole setup. Choosing a sensor with enough accuracy to detect small CO₂ surface leaks in combination with a fast response rate allows the developed helicopter UAV to be flown continuously over the area of interest. With a stop-and-go measurement approach avoided, the UAV can cover a significantly larger area in the same limited flight time. Furthermore, the previously negative influence of the main rotor onto the CO₂ concentration measurements (due to the dilution of the pre-experimental CO₂ concentration with “fresh” air sucked on by the rotor blades during a measurement phase in hover) as described by Neumann et al. in [63] can now be considered a positive factor (measuring the average CO₂ concentration of a volume instead of a point measurement, guaranteed airflow through the sensor’s measurement chamber). Developing the aerial platform around the chosen CO₂ sensor however made a larger and heavier UAV platform necessary than has been used with other approaches discussed in the literature (e.g. [59]). A reduction of the sensor weight in combination with a faster response rate would enable the use of smaller UAVs and the coverage of even larger areas.

5.4 Outlook and future research

The contributions of this thesis provide solutions which bring human-guided, autonomous UAVs for the atmospheric monitoring of onshore geological CO₂ storage sites closer to reality. During the development of the contributions discussed in this thesis, new areas which motivate further investigations have been discovered:

- To improve the sensor redundancy implementation of the developed CO₂ sensing UAV, a reliable non-GPS based positioning system should be incorporated into the aerial platform setup, e.g. a simultaneous localization and mapping (SLAM) approach could be utilized. The feasibility of the alternative positioning system for the task at hand has to be proven with real-world experiments in the targeted environment.

- The feasibility study of the developed helicopter UAV for the monitoring of onshore geological CO₂ storage sites showed that it is necessary to fly the aerial platform close to the ground in order to detect small CO₂ leaks. To further enhance the safety of the overall approach, the inclusion of an obstacle avoidance technique should therefore be considered. While the environment around a CCS site will be well known by the operator and therefore the flight path of the UAV can be planned accordingly to avoid static obstacles in the environment, one has to incorporate moving obstacles as well. Furthermore, with an obstacle avoidance system in place, the UAV could fly closer to problematic areas which one would otherwise avoid to ensure the safety of the aerial platform.
5.4. Outlook and future research

- The developed approach would strongly benefit from the development of lightweight sensors which can measure the atmospheric CO$_2$ concentration or tracer concentration with high accuracy and low response time. Sensing CO$_2$ tracers instead of the CO$_2$ concentration itself would allow the UAV to detect even smaller leaks and therefore storage containment breaches earlier. Furthermore, a high measurement update rate would allow the UAV to fly faster which would result in the coverage of a larger area in the same amount of time.

With the results presented in this thesis, it is already possible to monitor specific areas of interest of an onshore geological CO$_2$ storage site with a helicopter UAV. If one includes the improvements discussed above, it can be hoped that the whole area above and beyond the CO$_2$ storage complex can be monitored in a single flight by a single aerial vehicle in the close future.
Appendices
The relationship between Fang’s and Schmidt’s algorithm

Solving Equation 3.29, which represents the formula for the major axis of a general conic, with respect to $y$ results in

$$y = -\frac{\alpha_{ijk}}{\beta_{ijk}}x - \frac{\gamma_{ijk}}{\beta_{ijk}}z + \frac{\delta_{ijk}}{\beta_{ijk}}.$$  \hfill (A.1)

Assuming that Fang’s coordinate system is being used for Schmidt’s algorithm, the positions of the microphones $i, j, \text{ and } k$ would be

$$p_i_{\text{fang}} = [0 \ 0 \ 0]^T$$  \hfill (A.2)

$$p_j_{\text{fang}} = [x_{j_{\text{fang}}} \ 0 \ 0]^T$$  \hfill (A.3)

$$p_k_{\text{fang}} = [y_{k_{\text{fang}}} \ z_{k_{\text{fang}}} \ 0]^T$$  \hfill (A.4)

Using A.2 to A.4 in Equations 3.30 to 3.33 results in

$$\alpha_{ijk} = x_{j_{\text{fang}}}d_{ik} + x_{k_{\text{fang}}}d_{ji}$$  \hfill (A.5)

$$\beta_{ijk} = y_{k_{\text{fang}}}d_{ji}$$  \hfill (A.6)

$$\gamma_{ijk} = 0.$$  \hfill (A.7)
Calculating the distances of the microphones to the origin (Equation 3.3) using the information provided by A.2 to A.4, one gets

\[
R_{i,\text{fang}} = 0
\]
\[
R_{j,\text{fang}} = x_{j,\text{fang}}
\]
\[
R_{k,\text{fang}} = \sqrt{x_{k,\text{fang}}^2 + y_{k,\text{fang}}^2}
\]

which results in

\[
\delta_{ijk} = \frac{1}{2}(d_{ij}d_{ik}(-d_{ik} - d_{ji}) + x_{j,\text{fang}}^2d_{ik} + (x_{k,\text{fang}}^2 + y_{k,\text{fang}}^2)d_{ji})
\]

in the error-free case. With \(\gamma_{ijk} = 0\) one can see that the z component of Equation A.1 falls away, while

\[
\alpha_{ijk} = \frac{1}{d_{ij}d_{ik}(-d_{ik} + d_{ji}) + x_{j,\text{fang}}^2d_{ik} - (x_{k,\text{fang}}^2 + y_{k,\text{fang}}^2)d_{ji}}
\]
\[
\beta_{ijk} = \frac{1}{2y_{k,\text{fang}}}(k^2 - d_{ik}^2 + d_{ik}d_{ij}(1 - (\frac{x_{j,\text{fang}}}{d_{ij}})^2))
\]

with

\[
k = \sqrt{x_{k,\text{fang}}^2 + y_{k,\text{fang}}^2}.
\]

Equation A.13 and A.15 can be readily identified as the parameters \(\alpha\) (Equation 3.55) and \(\beta\) (Equation 3.56) of Fang’s position estimation planes. The intermediate step of calculating position estimation planes is therefore shared between Fang’s and Schmidt’s algorithms.
The link between Bucher and Misra’s method and Schmidt’s approach

Solving Equation 3.29, which represents the formula for the major axis of a general conic, with respect to $y$ results in

$$y = -\frac{\alpha_{ijk}}{\beta_{ijk}} x - \frac{\gamma_{ijk}}{\beta_{ijk}} z + \frac{\delta_{ijk}}{\beta_{ijk}}. \quad (B.1)$$

Using the relations

$$d_{kj} = -d_{ik} - d_{ji} \quad (B.2)$$
$$d_{ji} = -d_{ij} \quad (B.3)$$

which are valid for the error-free case, one can formulate

$$\begin{align*}
\alpha_{ijk} & = \frac{d_{ik}(x_j - x_i) - d_{ij}(x_k - x_i)}{d_{ij}(y_k - y_i) - d_{ik}(y_j - y_i)} \\
\beta_{ijk} & = \frac{d_{ij}(y_k - y_i) - d_{ik}(y_j - y_i)}{d_{ij}(y_k - y_i) - d_{ik}(y_j - y_i)} \quad (B.4) \\
\gamma_{ijk} & = \frac{d_{ik}(z_j - z_i) - d_{ij}(z_k - z_i)}{d_{ij}(y_k - y_i) - d_{ik}(y_j - y_i)} \quad (B.5) \\
\delta_{ijk} & = \frac{d_{ik}(d_{ij}^2 + R_i^2 - R_j^2) - d_{ij}(d_{ik}^2 + R_i^2 - R_k^2)}{2(d_{ij}(y_k - y_i) - d_{ik}(y_j - y_i))}. \quad (B.6)
\end{align*}$$

Equations B.4 to B.6 can be readily identified as the parameters $\alpha$ (Equation 3.66), $\beta$ (Equation 3.66) and $\gamma$ (Equation 3.68) of Bucher and Misra’s position estimation planes. The intermediate step of calculating position estimation planes is therefore the common link between Bucher and Misra’s method and Schmidt’s algorithm.


Bibliography


Bibliography


