technische universität
dortmund

Studienarbeit

Ortung einer akustischen Quelle im Roboterfußball mittels Multilateration

Fabian Ostermann

30. Januar 2020

Gutachter:
Dr. Lars Hildebrand

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Software Engineering (LS14)
http://ls14-www.cs.tu-dortmund.de

Inhaltsverzeichnis

Abbildung 1: Impressionen vom RoboCup 2017 aus Japan

Einleitung

Diese Arbeit widmet sich der softwareseitigen Implementierung eines audio-basierten Verfahrens zur Lokalisierung einer Sportpfeife im dreidimensionalen Raum. Die Plattform für die Implementierung bietet der zweibeinige humanoide Roboter NAO der Firma *SoftBank Robotics*[1] in der Version `V6`. Der Roboter besitzt vier im Kopf eingebaute Mikrofone. Eine vollständige Dokumentation der Plattform findet sich online[2]. Der Grund für die Wahl dieses Modells liegt im bereits mehrjährigen Engagement des Autors beim Roboterfußball-Team *NaoDevils Dortmund*. Dieses ist am Roboterinstitut der TU Dortmund (IRF) angesiedelt und langjähriges Mitglied der ROBOCUP-Federation[3]. Diese Initiative für intelligente Robotiksysteme besteht seit 1997 und kürt seit 2009 in der *Standard Platform League* (SPL) jedes Jahr einen Weltmeister unter internationalen NAO-Teams (siehe Abbildung 1). Hinzu kommen außerordentliche Wettbewerbe, die sogenannten *Challenges*, die bestimmte Teilaufgaben des Roboterfußballs hervorheben, um deren Entwicklung explizit zu fördern. Zur Einordnung: Beispiele aus den vergangenen Jahren sind der erschwerte Strafstoß-Wettbewerb (2018/2017), Kommunikation ohne WLAN (2016), Eckstöße (2015), Pfeifenerkennung (2014) und Durchführung eines Passes (2013). Eckstöße und Pfeifenerkennung sind mittlerweile fester Bestandteil der Regeln im Hauptwettbewerb geworden.

Die Challenge aus diesem Jahr beinhaltet die Lokalisation einer Pfeife. Hintergrund der Aufgabenstellung ist die Nutzung der Pfeife als Signal für die Roboter, dass der Anstoß ausgeführt werden darf. Die standardmäßige Kommunikation der Schiedsrichter mit den Robotern über WLAN wird hierfür verzögert, sodass ein Team durch die Erkennung

[1] `https://www.softbankrobotics.com/emea/en` (letzter Abruf: 30.1.2020)
[2] `http://doc.aldebaran.com/2-8/home_nao.html` (letzter Abruf: 30.1.2020)
[3] `https://www.robocup.org/` (letzter Abruf: 30.1.2020)

des Pfeiftons über die Mikrofone einen Vorteil im Spielverlauf erhält. Das Problem: Bei einem Großevent wie dem RoboCup befinden sich mehrere Felder nebeneinander. In der Vergangenheit führte ein Pfiff auf dem Nachbarfeld regelmäßig zum falschen Anlaufen der Roboter. Folge ist dann eine Zeitstrafe für die beteiligten Roboter. Die Arbeit an der Challenge soll es den Teams in Zukunft ermöglichen zu erkennen, ob auf dem eigenen Feld gepfiffen wurde.

Mithilfe der in dieser Arbeit präsentierten Implementierung erreichten die *NaoDevils Dortmund* beim RoboCup 2019 den 2. Platz in der *Directional Whistle Challenge* der SPL. Die Platzierungen aller Teams sowie die erzielten Punkte finden sich in Tabelle 1.

Platz	Punkte	Team	Universität
1.	8,5	Berlin United	Humboldt-Universität zu Berlin
2.	7,9	**NaoDevils**	TU Dortmund
3.	7,7	TJArk	Tongji-Universität (CHN)
4.	7,4	HULKs	TU Hamburg
5.	6,5	Bembelbots	Goethe-Universität, Frankfurt
6.	5,7	Nao-Team HTWK	HTWK Leipzig
7.	4,0	B-Human	Universität Bremen
	0	rUNSWift	UNSW Sydney (AUS)
	0	MiPal	Griffith U. (AUS)/Pompeu Fabra U. (ESP)
	0	Camellia Dragons	Aichi Prefectural University (JPN)

Tabelle 1: Platzierungen, Punkte und Zugehörigkeit aller teilnehmenden Teams der *Directional Whistle Challenge*[4]

Eine vollständige Beschreibung der Vorgaben sowie der Regeln zur Punktebestimmung wird in Abschnitt 1 gegeben. Danach werden verschiedene Lösungsmöglichkeiten diskutiert (Abschnitt 2). Zudem wird das gewählte Verfahren der *Time Difference of Arrival* (TDOA) mathematisch definiert. Abschnitt 3 behandelt die praktische Umsetzung des Verfahrens und die Beschränkungen der Nao-Plattform. Danach wird in Abschnitt 4 das konkrete Berechnungskonzept der Lösung mithilfe eines approximativen Suchverfahrens erläutert. Die Möglichkeit einer Lösung mithilfe der Gauss-Elimination wird ebenso vorgestellt und verglichen. In Abschnitt 5 wird das Verfahren abschließend evaluiert, indem die Daten des Testprogramms ausgewertet und ein Vergleich zu den anderen Teams hergestellt wird. Zudem wird die Implementierung mithilfe simulierter

[4]Offizielle Ergebnisse online einsehbar: `https://spl.robocup.org/technical-challenges-2019/` (letzter Abruf: 30.1.2020)

Eingabedaten statistisch auf Robustheit geprüft und auftretende Problemfelder genauer eingegrenzt. Die Arbeit endet mit einer Zusammenfassung der wichtigsten Ergebnisse und einem Ausblick für zukünftige Forschungsthemen und Anwendungen.

1. Inhalt und Reglement der Challenge

Das Ziel der *Directional Whistle Challenge* ist die korrekte Lokalisierung des Schiedsrichters zum Zeitpunkt seines Pfiffs mithilfe der Sensoren der NAOs. Die vorgegebenen Regeln dienen der Bewertung der Qualität der vorgeführten Verfahren und so dem kompetitiven Vergleich der teilnehmenden Teams. Die spezifische Aufgabenstellung soll die potentielle Nutzung akustischer Merkmale zur Umgebungserfassung untersuchen und fördern. In diesem Abschnitt soll das offizielle Reglement [TC, 2019b] anhand der Aspekte organisatorischer Aufbau, praktischer Ablauf und Punktewertung erläutert werden.

1.1. Aufbau und Ablauf

Wie im gewöhnlichen Spiel[5] darf jedes Team bis zu fünf Roboter mit den Trikotnummern 1 bis 5 aufstellen. Die Nummer entscheidet auf welche Position der Roboter vom Schiedsrichter gestellt wird. Zu einer Position auf dem Feld gehört auch immer die Rotation. Die möglichen Positionen wurden erst kurz vor der Challenge bekanntgegeben und sind in Abbildung 2 eingezeichnet. Die Außenlinie des Feldes, das genauso auch im gewöhnlichen Spiel benutzt wird, misst $9\,\mathrm{m} \times 6\,\mathrm{m}$. Die Roboter dürfen sich während der Challenge nicht bewegen. Lediglich das Drehen des Kopfes ist erlaubt.

Die Roboter dürfen beliebig Daten über WLAN austauschen. Auf gleichem Weg muss das Ergebnis der Erkennung zur Punktewertung per spezifiziertem UDP-Paket an die vorgesehene Testapplikation[6] gesendet werden. Jedes Paket muss zwei Informationen enthalten: Eine 2D-Koordinate mit Ursprung im Anstoßpunkt sowie einen Wahrheitswert, der angibt ob der Pfiff zum eigenen Feld „gehört“[7].

Zu Beginn eines jeden Versuchs werden die Roboter von den Schiedsrichtern auf den vorgesehenen Positionen auf dem Feld platziert. Dann wird mit einer gewöhnlichen

[5] Die Regeln des eigentlichen Hauptwettbewerbs werden an den nötigen Stellen erläutert. Für Details sei auf das offizielle Regelwerk [TC, 2019a] verwiesen.

[6] *Directional Whistle Tester*, entwickelt vom TC-Team B-Human: `https://github.com/bhuman/DirectionalWhistleTester` (letzter Abruf: 30.1.2020)

[7] Obwohl für Audioverfahren vorgesehen, werden durch die getrennte Bewertung der Feldzugehörigkeitserkennung und der Ermittlung der konkreten Koordinate verschiedenste (kreative) Ansätze ermöglicht. Beispielsweise wäre eine außergewöhnliche Lösung die visuelle Erkennung der Pfeife im Mund des Schiedsrichters.

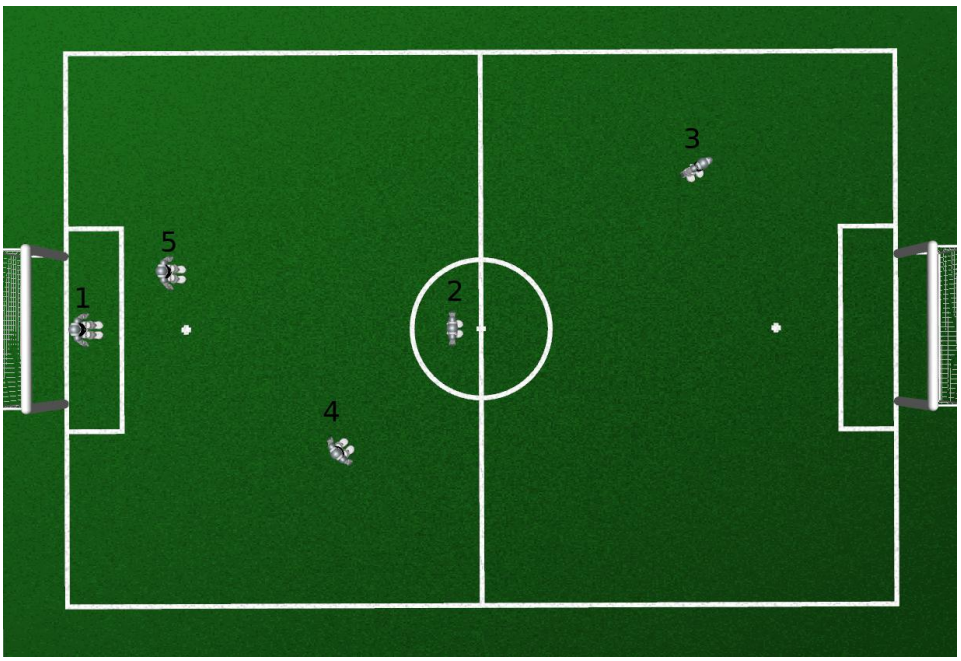


Abbildung 2: Festgelegte Roboterpositionen während der Challenge [TC, 2019b, S.3]

Sportpfeife nacheinander von acht verschiedenen festgelegten Positionen aus gepfiffen. Dabei kann sich der Schiedsrichter auf dem selben Feld wie die Roboter oder aber einem angrenzenden Feld befinden. Die Testapplikation wird zeitgleich mit jedem Pfiff aktiviert. Als Lösung wird das erste Paket betrachtet, das innerhalb von 5 Sekunden eingeht. Nach Ablauf der 5 Sekunden geht der Schiedsrichter auf die nächste Position.

1.2. Punktewertung

Die Testapplikation berechnet automatisch zu jedem der acht Lösungsversuche eine Bewertung in Form von einer erreichten Punktzahl. Sie berechnet sich aus drei Summanden. Der Erste belohnt die korrekte Angabe der *Feldzugehörigkeit* mit 1 Punkt (*Feldpunkt*). Die beiden anderen Bewerten die Genauigkeit der 2D-Koordinate. Dazu wird diejenige Roboterposition, die der realen Pfiffposition beim aktuellen Versuch am Nächsten liegt, als Ursprung eines Polarkoordinatensystems betrachtet.

Der zweite Summand nutzt die Winkelwerte (*Richtungsmaßpunkt*). Weicht die übermittelte Lösungskoordinate (umgewandelt ins Polarkoordinatensystem) um weniger als 5° von der wahren Lösung ab, ergibt das 1 Punkt. Von 5° bis 30° fällt die Wertung linear von 1 Punkt auf 0 Punkte ab. Die lineare Punkteformel $p(x)$ für den Winkel x in Grad lautet demnach:

$$p(x) = 1 - \frac{x-5}{30-5} \quad \text{für } 5° < x < 30° \tag{1}$$

Der dritte Summand betrachtet den Distanzwert der Polarkoordinaten (*Distanzmaßpunkt*). Wenn die Distanz um weniger als 5 % abweicht, wird 1 Punkt vergeben. Zwischen 5 % und 30 % wird analog zu Gleichung (1) linear auf 1 bis 0 Punkte interpoliert.

Die Summen der acht Einzelversuche addieren sich zur Gesamtpunktzahl. So können insgesamt bis zu 24 Punkte erzielt werden. Besonderheit: Im Falle eines Gleichstandes wird diejenige Mannschaft bevorteilt, die weniger Roboter aufgestellt hat.

2. Mögliche Lösungsansätze

Der Begriff der *Ortung* erweitert das Konzept der bloßen *Detektion* eines Signals um die Bestimmung seines Ursprungsortes und somit der Position des emittierenden Objektes. In der Robotik wird das Schlagwort der *Indoor Positioning Systems* verwendet, das Ansätze verschiedenster Messmethoden, wie Bilderkennung, Funkortung, Echoortung und weitere Sensormethoden, umfasst. Klassisches Problem ist die Ortung von Personen oder Objekten mit dem Ziel der Interaktion.

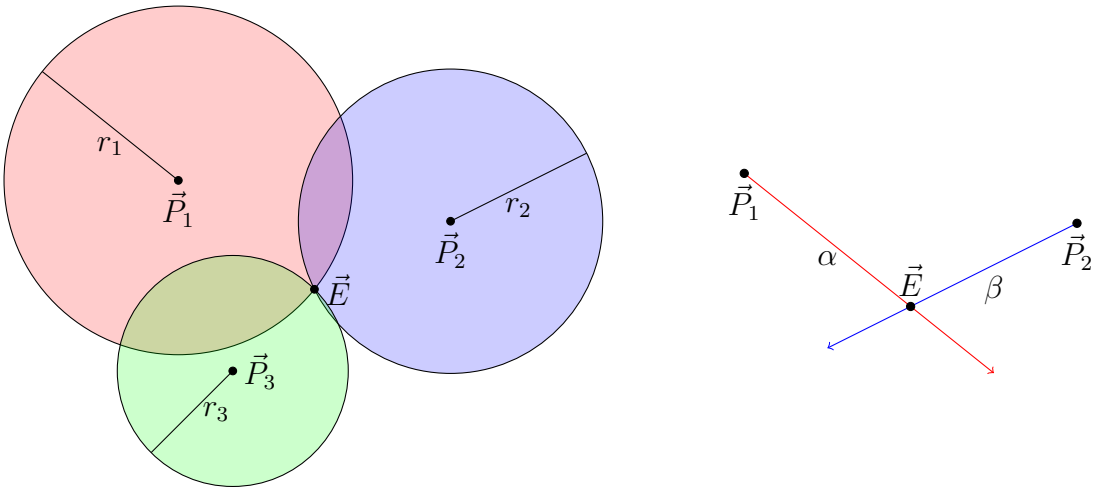
Ortung darf nicht mit dem Begriff der *Lokalisierung* verwechselt werden, die ein ebenso populäres Problem der autonomen Robotik darstellt. Dabei handelt es sich allerdings um die Bestimmung der eigenen Position. Bei der *Directional Whistle Challenge* sind die Roboterpositionen vorgegeben und stellen somit eine gesicherte Information dar (vgl. Abschnitt 1.1). Die globale Ortung benötigt zusätzlich die Eigenlokalisierung. Die Implementierung von Jochmann u. a. [2012] kann dafür in aktualisierter Form [Hofmann u. a., 2016, Abschn. 3.4] in einer realen Spielsituation angewandt werden.

Der Ansatz dieser Arbeit baut auf der bestehenden Detektion des akustischen Pfeifensignals (siehe Abschnitt 3.3) auf und erweitert ihn um die Positionsortung. Grundlegend für das Ortungsverfahren ist die physikalisch nahezu konstante Ausbreitungsgeschwindigkeit des Schalls, die eine Abstandsschätzung auf Grundlage von Laufzeitmessungen ermöglicht. Ein solches Verfahren ist in der Theorie identisch für jede Art der Wellenausbreitung.

Das prominenteste Beispiel einer (Outdoor-)Positionsbestimmung ist das *Global Positioning System* (GPS). Allerdings handelt es sich hier um ein passives Lokalisierungsverfahren. Das Empfangsgerät berechnet seine eigene Position mithilfe der Funksignale mehrerer Sender mit bekannter Position.

Das Problem der 3D-Pfeifenortung ähnelt eher der Geräteortung im *Global System for Mobile Communications* (GSM-Ortung). Hier wird das Funksignal eines Mobiltelefons an mehreren Basisstationen erfasst. Aufgrund der physikalisch-konstanten Ausbreitungsgeschwindigkeit elektromagnetischer Wellen (Lichtgeschwindigkeit) ist eine Entfernungsmessung durch Zeitmessung möglich.

Egal ob das Ziel die Ortung oder Lokalisierung ist, das mathematische Prinzip dahinter ist die sogenannte *Lateration*. Mithilfe von mindestens drei bekannten Positionen $\vec{P}_i$ in der Ebene (bzw. vier im Raum) und der jeweiligen Abstandsinformation r_i zur gesuchten Position des Emitters $\vec{E}$ ist eine eindeutige Positionsbestimmung möglich (vgl. Abbildung 3a). Die Mengen aller Punkte mit dem jeweiligen Abstand zu den bekannten Positionen implizieren drei Kreise (im Raum vier 2-Sphären). Der gesuchte Punkt liegt auf dem eindeutigen Schnittpunkt dieser Kreise.



(a) Trilateration (entfernungsbasiert) (b) Triangulation (winkelbasiert)

Abbildung 3: Schematik der grundlegenden Verfahren zur Positionsbestimmung

Im Anwendungsfall sind möglicherweise zwei bekannte Positionen in der Ebene zur Positionsbestimmung des Emitters ausreichend. Bei zwei implizierten Kreisen kann durch eine weitergehende Plausibilitätsprüfung gegebenenfalls einer der beiden resultierenden Schnittpunkte verworfen werden. Ein Beispiel für unseren konkreten Anwendungsfall: Die Pfeife kann sich nur in einer Höhe von 0 m bis maximal[8] 2,51 m befinden. Ein negativer Wert ist aufgrund des ebenen Spielfelds nicht möglich.

Die *Angulation* ist das Winkel-Pendant der Lateration als Positionsbestimmung mittels Entfernungsmessungen. Sie bestimmt die Postion des Emitters anhand von Winkel-

[8]Sultan Kösen (*1982), „größter lebender Mensch“ laut *Guinness-Buch der Rekorde*

messungen zu den bekannten Positionen (vgl. Abbildung 3b). Großer Vorteil des Verfahrens ist die einfachere Berechnungsweise des Schnittpunktes der Winkelstrahlen. Zudem sind in 2D zwei Empfänger zur eindeutigen Positionsbestimmung ausreichend (*Kreuzpeilung*). Angewandt auf die Ortung einer Schallquelle ist die Messung des Einfallswinkels allerdings nicht trivial. Auf der NAO-Plattform ist bestenfalls eine ungefähre Winkelmessung anhand von Amplitudenvergleich oder einer weiteren Lateration über die vier eingebauten Mikrofone möglich.

Im folgenden Abschnitt 2.1 werden zuerst die mathematischen Grundlagen der Lateration erläutert. Eine anwendungsbezogene Einschätzung der Implementierbarkeit auf der NAO-Plattform folgt in Abschnitt 2.2. Auch die (ergänzende) Einsetzbarkeit der Angulation wird dort genauer diskutiert.

2.1. Geometrie des Laterationsverfahrens

Die Lateration durch Sphären (vgl. Abbildung 3a) kann mithilfe synchronisierter Uhren in den Empfangsgeräten und dem Emitter erreicht werden. Entscheidend ist die Bestimmung der sogenannten *Time of Arrival* (TOA). Sie stellt den genauen Zeitpunkt T_i dar, an dem das (Schall-)Ereignis vom jeweiligen Empfänger i registriert wird. Mithilfe der Information über den Sendezeitpunkt T_E, der sogenannten *Time of Transmission* (TOT), können die absoluten Distanzen R_i zwischen dem Emitter $\vec{E}$ und den Positionen $\vec{P_i}$ der Empfänger berechnet werden.[9] Im $\mathbb{R}^n$ werden $n+1$ Empfänger zur Bestimmung von $\vec{E}$ benötigt:

$$\begin{aligned} R_i = |\vec{P_i} - \vec{E}| &= c_s(T_i - T_E) \\ &\text{mit } i \in \{1, 2, \ldots, n+1\} \\ &\text{und } c_s = 343{,}42\,\mathrm{m\,s^{-1}} \text{ (Schallgeschindigkeit)} \end{aligned} \tag{2}$$

Um die Zeitdifferenzen (in Sekunden) in Entfernungen (in Metern) zu überführen, wird die Differenz mit der physikalisch (nahezu) konstanten Ausbreitungsgeschwindigkeit c_s multipliziert. Die Geschwindigkeit des Schalls in trockener Luft ist geringfügig temperaturabhängig. Die angegebene Geschwindigkeit wurde mit der Formel $c_s = (331{,}3 + 0{,}606\tau)\mathrm{m\,s^{-1}}$ für eine Temperatur von $\tau = 20\,(°\mathrm{C})$ berechnet.

[9] R ist die in der TDOA-Literatur übliche Bezeichnung für die Entfernung zum Emitter [vgl. Bucher u. Misra, 2002, S.508].

Die Synchronisation der Empfangsgeräte zur Bestimmung der jeweiligen TOAs ist in jedem Fall notwendig. Ihre Hardware muss eine hinreichende Genauigkeit ermöglichen. Die Synchronisation der Uhren der NAO-Plattform wird in Abschnitt 3.1 behandelt.

Die Bestimmung der TOT ist weitaus problematischer. In den zuvor erwähnten, verbreiteten Systemen der GPS-Lokalisierung und GSM-Ortung sind die gängigen Anwendergeräte (meist einfache Smartphones) nicht in der Lage hinreichend präzise Zeitstempel zu generieren. Im betrachteten Fall der Pfeifen-Ortung ist der Emitter sogar unbekannt und völlig unzugänglich.

Abhilfe schafft die Umgehung der TOT-Berechnung durch die Bestimmung der *Time Difference of Arrival* (TDOA). Sie ergibt sich aus den Differenzen der einzelnen TOAs und kann mathematisch folgendermaßen definiert werden:

$$T_{i,j}^{\Delta} = T_i - T_j \quad \text{(TDOA)} \tag{3}$$

$$c_s T_{i,j}^{\Delta} = |\vec{P}_i - \vec{E}| - |\vec{P}_j - \vec{E}| = R_i - R_j \tag{4}$$

Anstelle einer Sphäre für den Fall der einfachen Lateration durch Distanzen, ergibt die TDOA-Methode geometrisch eine Hyperbel (im Raum ein halber einschaliger Hyperboloid). Die Empfänger P_i und P_j befinden sich grade auf den beiden Brennpunkten. Abbildung 4 veranschaulicht die Definition aus Gleichung (4). Die rote Hyperbel beschreibt die Menge aller möglichen Positionen für $\vec{E}$ bei gegebenen Empfängerpositionen $\vec{P}_i$ und $\vec{P}_j$ sowie gemessenem Laufzeitunterschied $T_{i,j}^{\Delta}$.

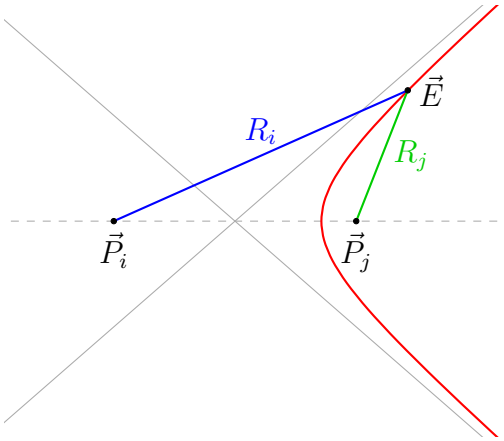


Abbildung 4: Veranschaulichung der Hyperbeldefinition aus Gleichung (4)

Die Berechnung auf Grundlage der bekannten Empfängerpositionen und Messung der TOAs ist kein lineares Problem mehr. Möglichkeiten für eine computergestützte Lösung sind Approximation durch einen Suchalgorithmus oder mathematische Linearisierung.

Beide werden in Abschnitt 4 erläutert. Zudem werden zur eindeutigen Positionsbestimmung durch TDOAs theoretisch nun $n + 2$ Empfänger im $\mathbb{R}^n$ benötigt.

Abbildung 5 zeigt den 2D-Fall mit vier Empfängern und drei resultierende Hyperbeln, die sich im Punkt $\vec{E}$ schneiden. Die drei Hyperbeln beziehen sich stets auf den Ausgangspunkt $\vec{P}_1$. Insgesamt ergeben vier Empfänger sechs mögliche Hyperbeln in Bezug zum Schnittpunkt $\vec{E}$. In der Theorie kann der Bezugspunkt beliebig gewählt werden, der Schnittpunkt bleibt unverändert. Man beachte die Verschiedenartigkeit der geometrischen Ausprägungen der Hyperbeln. Bei der roten Hyperbel, die sich zwischen $\vec{P}_1$ und $\vec{P}_2$ aufspannt, handelt es sich fast um eine Gerade. Die blaue Hyperbel um P_3 hingegen ist geometrische stark gestreckt und nähert sich fast zwei Strahlen in einem ziemlich spitzen Winkel an.

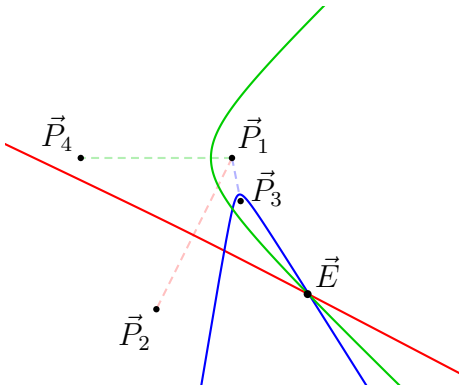


Abbildung 5: TDOA-Hyperbeln schneiden sich in Emitterposition $\vec{E}$

2.2. Der Anwendungsfall: Multilateration

Die Beispiele GPS und GSM nutzen aufgrund der bereits beschriebenen Probleme bei der Bestimmung der TOT tatsächlich Varianten des vorgestellten TDOA-Konzepts zur Positionsbestimmung. Beide Systeme sind dabei auf ihre besonderen Merkmale optimiert. GSM nutzt unter anderem ein Verfahren namens *Uplink Time Difference of Arrival* (U-TDOA), das eingehende Signale des Mobilgeräts an verschiedenen Basisstationen auswertet. Ein anderes Verfahren namens *Enhanced Observed Time Difference* (E-OTD) vermischt den Ansatz von U-TDOA mit der einfacheren Lateration durch Kreise, bei GSM *Timing Advance* (TA) genannt, welches die (ungenauere) „hin-und-zurück“-Laufzeit zur Distanzbestimmung nutzt.

GPS besteht aus mehr als 30 Satelliten, die ein geeichtes Zeitsignal senden. Da sie sich nicht auf geostationärer Umlaufbahn befinden und Daten von durchschnittlich 8–10 Sa-

telliten gleichzeitig bezogen werden, ist die mathematische Modellierung komplexer als die aus Abschnitt 2.1. Das *Assisted Global Navigation Satellite System* (A-GNSS) ist eine Kombination der GPS- und GSM-Verfahren für Mobilgeräte. Es steigert die Präzision und kann vorübergehende Ausfälle, zum Beispiel durch Funkschatten, kompensieren.

Wenn in der Praxis wie bei GPS für die Lateration mit TDOA-Informationen mehr als die notwendigen $n + 2$ TOAs genutzt werden, spricht man von *Multilateration*. Mit realen Daten schneiden sich die drei Kreise aus Abbildung 3a nicht *exakt* in einem Punkt. Eine einzelne endgültige Lösung trotz der Ungenauigkeiten zu bestimmen ist wesentlicher Bestandteil des Verfahrens (siehe Abschnitt 4.3). Eine mathematische Überbestimmtheit kann in der Praxis zudem Messfehler korrigieren und Ausreißer filtern. Diese Art der Optimierung ist bei der Implementierung des Verfahrens für die NAO-Plattform, deren Hardware alles andere als Laborgenauigkeit zulässt, unverzichtbar.

Das erste Ortungssystem auf Grundlage akustischen Schalls war die militärische *Schallmesstechnik*, die erstmals im Ersten Weltkrieg Anwendung fand. Die plötzliche und enorm starke Druckwelle eines Artilleriegeschützes (*Knall*) wird an mehreren verteilten Mikrofon-Messstellen detektiert. Die Laufzeitunterschiede ergeben die Koordinaten des Ursprungsorts (analog zu Abbildung 5).

Die NAOs besitzen vier Mikrofone in der Kopfpartie. Theoretisch wäre damit eine Lateration durch TDOAs denkbar. Die beiden am weitesten auseinanderliegenden Mikrofone befinden sich in einem Abstand von $l_{max} = 10{,}59\,\text{cm}$ [vgl. Aldebaran, 2017, Microphones]. Das bedeutet einen maximalen Laufzeitunterschied von $\nu_s l_{max} = 0{,}3\,\text{ms}$. Bei der Hardwareunterstützung einer maximalen Abtastrate von $44\,100\,\text{Hz} = 0{,}02\,\text{ms}$ ist das eine maximale Laufzeitdifferenz von 15 Abtaststellen. Zur Angulation eines Knalls mit der Genauigkeit von $\frac{180°}{15} = 12°$ wäre das theoretisch ausreichend. Der Startzeitpunkt des sich aufbauenden Pfeiftons kann allerdings zeitlich nicht eindeutig bestimmt werden. Eine detaillierte Problemdarstellung der exakten Bestimmung des Startzeitpunkts folgt in Abschnitt 3.3 (siehe dort Abbildung 7).

Weitere Probleme sind die nicht vollständig symmetrische Anordnung im Gehäuse (die hinteren Mikrofone liegen tiefer), die omni-direktionale Arbeitsweise der verbauten Mikrofone, starke Störgeräusche innerhalb des Gehäuses durch Lüfter sowie das Übersprechen der Mikrofone durch Gehäusevibration bei lauten Geräuschen. Aufgrund dieser Widrigkeiten wird die Lateration mithilfe eines einzelnen Roboters hier nicht weiter behandelt. Stattdessen soll ein Verfahren auf Grundlage mehrerer auf dem Feld verteilter Roboter implementiert werden. Die Regeln in Abschnitt 1 erlauben es bis zu fünf Ro-

boter zu verwenden. Das Prinzip der zuvor besprochenen Schallmesstechnik erlaubt bei fünf Empfängern die eindeutige Positionsbestimmung in 3D-Koordinaten.

3. Systemaufbau und Modulstruktur

Zur Vereinfachung und Vereinheitlichung der Sensordatenaufbereitung, Informationsverarbeitung und Hardwareansteuerung nutzen die *NaoDevils Dortmund* ein Modulframework [Hofmann u. a., 2018] mit sogenannter *Blackboard*-Funktion. Es basiert auf der 2015er-Codebasis des Bremer SPL-Teams *B-Human* [Röfer u. a., 2015], welche wiederum ihren Ursprung in der Arbeit des *German Team* hat [Röfer u. a., 2007].

Die Implementierung des präsentierten Verfahrens nutzt das bestehende Modulframework. Das Kernkonzept ist die selbstorganisierte Ausführung einzelner Module, die sogenannte *Representations* erzeugen. Die `update`-Funktion eines jeden Modules wird in festgelegter Taktzeit aufgerufen. Jedes Modul muss dabei mindestens eine Representation erzeugen, die im Blackboard schreibgeschützt aber für alle Module zentral zugängig gespeichert wird. Dazu kann jedes Modul Representations aus dem Blackboard anfordern, um deren Informationsinhalte weiterzuverarbeiten. Die Definition eines Moduls legt die angeforderten und bereitgestellten Representations fest. Damit kann ein Graph erzeugt werden, der eine eindeutige Ausführungsreihenfolge bestimmt. Tritt dabei ein Zyklus auf, kann dieser durch Rückgabe der angeforderten Representation aus dem vorherigen Durchlauf gelöst werden.

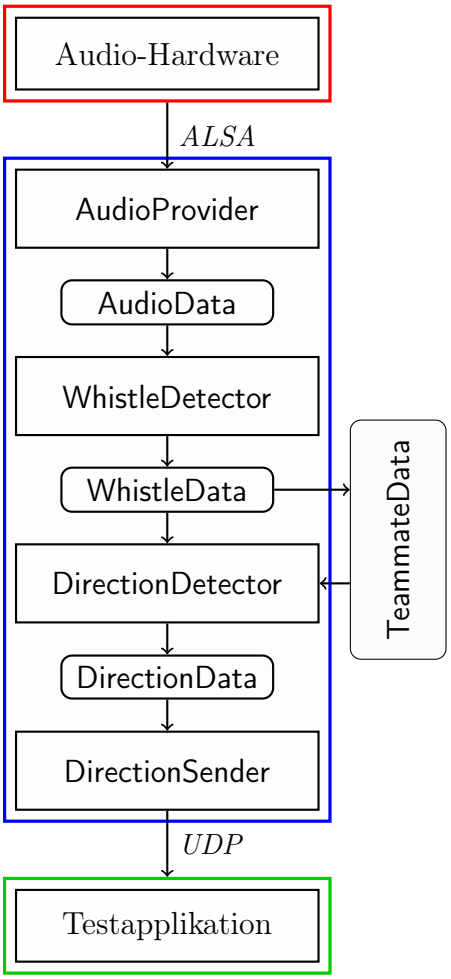


Abbildung 6: Schematische Modulübersicht

Da die für das Verfahren entworfene Modulstruktur linear ist, wird auf weitergehende Beschreibungen der mächtigen Konzepte des gewachsenen Frameworks verzichtet und stattdessen auf die oben angegebene Literatur verwiesen. Abbildung 6 zeigt den zum Verständnis der Implementierung notwendigen Ausschnitt der Modulstruktur. Die vom Framework verwalteten Module sind blau umrandet. Die Representations zum Datenaustausch sind visuell durch abgerundete Ecken sowie die Namensendung „Data“ identifizierbar.

Die Mikrofon-Hardware und die eingebaute Soundkarte des NAOs ist in Abbildung 6 rot gekennzeichnet. Die Ansprache über die *Advanced Linux Sound Architecture* (ALSA) durch das Modul AudioProvider ist in Abschnitt 3.2 beschrieben. Der zur Synchronisation der Audiostreams notwendige Uhrenabgleich der Roboter wird in Abschnitt 3.1 behandelt. Die Detektion des Pfeifsignals im Stream übernimmt der WhistleDetector. Dessen Verarbeitung der AudioData und die erzeugten Informationen werden in Abschnitt 3.3 erläutert.

Der DirectionDetector berechnet aus den Werten in WhistleData eine Positionsschätzung für die Pfeife. Der Hauptalgorithmus des Verfahrens ist in diesem Modul implementiert. Eine ausführliche Beschreibung folgt in Abschnitt 4. Der C++-Quellcode dieses zentralen Moduls ist in Anhang A zu finden.

Zu beachten ist, dass die Modulstruktur in der präsentierten Form auf allen am Ortungsversuch beteiligten Robotern in gleicher Form abläuft. Die WhistleData wird durch ein im Framework bestehendes Modul namens TeamDataSender geteilt und durch den TeammateDataProvider als lokal abrufbare TeammateData aufbereitet [Röfer u. a., 2013, Abschn. 3.5.4]. Verschickt wird eine festgelegte Auswahl an Representations über das verbindungslose *User Datagram Protocol* (UDP) [IETF, 2019, RFC 768]. Dadurch erlangt der DirectionDetector zusätzlichen Zugriff auf die WhistleData der anderen Roboter. Durch die zeitliche Synchronisation der Uhren wird damit die Berechnung aller notwendigen Laufzeitdifferenzen möglich.

Der DirectionSender übernimmt letztlich die Aufgabe die erzeugte Positionsschätzung an die Testapplikation zu übermitteln. Er bereitet die DirectionData als regelkonformes UDP-Paket auf und sendet es innerhalb der geforderten 5 Sekunden (siehe Abschnitt 1.1). Die Testapplikation ist in Abbildung 6 grün gekennzeichnet.

3.1. Synchronisation der Uhren

Die Synchronisation der Audiostreams basiert auf der möglichst präzisen Kalibrierung der NAO-eigenen Uhren. Die Roboter nutzen dafür das *Network Time Protocol* (NTP) [IETF, 2019, RFC 958]. Softwareseitig läuft allerdings nicht die gleichnamige Referenzimplementierung[10], sondern eine spezielle Variante namens *chrony* [Curnow u. a., 2019]. Diese ist auf erschwerte Bedingungen wie unregelmäßige Serververbindung, überlastetes Netzwerk, ungenaue Systemuhren oder schwankende Temperaturen[11] optimiert. Dabei ist vor allem die stetige Schätzung der systemeigenen, meist konstanten Abweichungsrate

[10] *The NTP Project*: `http://www.ntp.org/` (letzter Abruf: 30.1.2020)
[11] Die Frequenz von üblichen Quarzoszillator ist geringfügig temperaturabhängig.

des Uhrtaktes ein entscheidender Vorteil. Die Ungenauigkeit der Hardwareuhr wird durch *chrony* softwareseitig herausgerechnet. Die Roboter können sich während der Challenge (genau wie im Spiel) nicht mit dem Zeitserver synchronisieren. Die für das Positionsbestimmungsverfahren notwendige Genauigkeit kann so bewahrt werden. Nimmt man eine maximale Trennzeit vom Netzwerk bis zum Start der Challenge von drei Stunden an, so beträgt die maximal gemessene Abweichung für diese Zeit auf der NAO-Platform weniger als 1 µs. Multipliziert mit ν_s sind das weniger als 0,3 mm auf die Positionsungenauigkeit.

3.2. Audiostreaming mit ALSA

Das Modul AudioProvider nutzt ALSA (*Advanced Linux Sound Architecture*) [Kysela u. a., 2019] um die Audiodaten von der Soundkarte auszulesen. In regelmäßigen Abständen werden die Spannungssignale der Mikrofone durch einen Analog-Digital-Wandler (ADW) digitalisiert. Einen einzelnen Abtastwert nennt man *Sample*.[12]

Zunächst wird die Verbindung zur Hardware sowie Konfigurationsanweisungen initialisiert. Zum Verbinden dient die ALSA-Bibliotheksfunktion `snd_pcm_open()`. Anschließend können Konfigurationsparameter mit `snd_pcm_hw_params_set_*()` gesetzt werden. Für das präsentierte Verfahren werden eine Abtastrate von $f_s = 44\,100\,\text{Hz}$, alle vier Mikrofonkanäle sowie die Darstellung der Samples als normierte Gleitkommazahlen $s_i \in [-1, 1]$ über ALSA angefordert. Hierbei ist zu beachten, dass die `set_*`-Funktionen einen negativen Fehlerwert zurückgeben können. Dann ist die Hardware nicht in der Lage die geforderte Konfiguration zu erfüllen. In dem Fall setzt ALSA die nächstbeste Konfiguration nach eigenem Ermessen. Zum Beispiel ist die Abstastrate nicht beliebig wählbar. ALSA entscheidet sich für diejenige mit der geringsten Differenz. Die Vorgängerversion `V5` des NAO war zudem nicht in der Lage, hardwareseitig eine Gleitkommadarstellung für vier Kanäle zu erzeugen. Die Umrechnung musste im Framework manuell programmiert werden.

Mit dem Befehl `snd_pcm_readi()` wird der Puffer geleert und sein Inhalt in das Programm überführt. Der Suffix `i` bezeichnet den *interleaved*-Modus, in dem die Samples der Kanäle im Array abwechselnd angeordnet sind. Der Puffer wird von der verbauten Hardware in regelmäßigen Abständen gefüllt und hat dann einen Inhalt von umgerechnet 184 ms. Der AudioProvider wird vom Framework automatisch 30 mal pro Sekunde

[12] Auf das Grundprinzip der Diskretisierung von analogen Spannungssignalen wird hier nicht weiter eingegangen. Für nähere Erklärungen siehe Butz [2015, Abschn. 4.1] oder auch: `https://www.alsa-project.org/alsa-doc/alsa-lib/pcm.html` (letzter Abruf: 30.1.2020)

ausgeführt, kann also durchschnittlich in jedem $184\,\text{ms} \cdot 30\,\text{Hz} \approx 5{,}5$ Durchlauf neue Audiodaten auslesen.

Um innerhalb der Audiostreams der einzelnen Roboter Startzeitpunkte des Pfeifsignals bestimmen zu können, wird nun die optimierte Systemuhr genutzt. Der AudioProvider schreibt das Audiodaten-Array und den *chrony*-Zeitstempel T_{chrony} zum Zeitpunkt des Abrufs der Daten in die Representation AudioData. Der Stempel gibt den genauen Aufnahmezeitpunkt des letzten Samples an. Daraus kann dann für jeden Sample im Array der Zeitstempel zurückgerechnet werden (Näheres folgt in Abschnitt 3.3). ALSA stellt dazu die Funktion `snd_pcm_htimestamp()` zur Verfügung. Diese wird auf der NAO-Plattform allerdings hardwareseitig nicht unterstützt. Sinn dieser Funktion ist es, den exakten Zeitpunkt der Befüllung des Puffers und nicht nur den Abruf der gepufferten Daten zu liefern. Der `readi()`-Aufruf kann bei 30 Hz bis zu 33 ms nach Befüllung des Puffers erfolgen. Abhilfe schafft die Funktion `snd_pcm_avail_delay()`. Sie gibt die Größe der aktuell verfügbaren Audiodaten im Puffer zurück, die dann an `readi()` übergeben wird. Zusätzlich wird eine Schätzung der vergangenen Zeit zwischen Befüllung und Abruf gegeben. Mithilfe dieser *Delay*-Zeit t_{delay} kann der *chrony*-Zeitstempel in AudioData noch einmal korrigiert werden zu $T_{in} = T_{chrony} - t_{delay}$.

Die Funktion `snd_pcm_delay()` muss mit Vorsicht behandelt werden. Unter Laborbedingungen wurde die Genauigkeit getestet: Ein Knall (Händeklatschen) wird in unmittelbarer Nähe zu zwei synchronisierten Robotern erzeugt. Der Zeitpunkt des Knalls kann bei ruhiger Umgebung über ein einfaches Schwellwertverfahren Sample-genau bestimmt werden. Auswertung der Zeitstempel zeigte für 87 % der Ergebnisse eine Genauigkeit unter 1,2 ms. Die Hälfte lag sogar unter 0,5 ms. Der höchste gemessene Wert war 42 ms. Gelegentliche Ausreißer bei den einzelnen Ortungsversuchen sind also zu erwarten und auf die beschriebenen Hardwareeinschränkungen der NAO-Plattform zurückzuführen. Möglichkeiten zur Eliminierung der Ausreißer werden in Abschnitt 4.3 erläutert.

3.3. Detektion des Pfeifsignals

Die Grundlage des WhistleDetector bildet die Frequenzanalyse des Audiostreams. Das Pfeifsignal besteht aus einer Grundfrequenz f_0 sowie zwei charakteristischen Obertönen bei ziemlich genau $2f_0$ und $3f_0$. Um die Amplituden der einzelnen Frequenzen im

diskreten Audiostream zu berechnen, wird die *Diskrete Fourier-Transformation* (DFT) angewendet.[13]

Die Grundlage des Detektionsverfahrens hat der Autor in einer Projektgruppe erarbeitet [Ostermann u. a., 2016]. Die DFT wird genutzt, um für die verfügbaren Audiosamples die Amplituden einzelner Frequenzen zu bestimmen.[14] Es wird angenommen, dass die Grundfrequenz des schrillen Pfeifsignals im Bereich von 2200 Hz–3800 Hz liegt und dort eine größere Amplitude hat als jedes Störgeräusch. Also ist die größte Amplitude in diesem Bereich gleich f_0. Überschreitet f_0 einen Schwellwert, so werden auch $2f_0$ und $3f_0$ auf Überschreitung von Schwellwerten geprüft. Die drei benötigten Schwellwerte wurden durch Auswertung einer Versuchsreihe bestimmt. Um den Berechnungsaufwand und die Unabhängigkeit zu Störgeräuschen zu verringern, werden die gesamten Audiodaten (ca. 184 ms pro Durchlauf, siehe Abschnitt 3.2) in halb-überlappende Fenster mit jeweils 1024 Samples eingeteilt. Auf jedes Fenster wird vor der Transformation die *von-Hann*-Funktion [Butz, 2015, Abschn. 3.4] angewandt, um mögliche Störfrequenzen an den Fensterrändern zu beseitigen. Der Versatz der Fenster beträgt für die genannten Parameter

$$\frac{1024}{22\,050\,\mathrm{Hz}} \cdot \frac{1}{2} \approx 23{,}22\,\mathrm{ms}. \tag{5}$$

Werden die Schwellwerte in vier aufeinanderfolgenden Fenstern überschritten, gilt das Pfeifsignal als positiv erkannt. Die zeitliche Länge des Pfeifsignal muss also mindestens das Vierfache des Fensterversatzes betragen.

Die Pfeifenerkennung aus der Projektgruppe ist für die NAO-Version `V5` optimiert. Durch die Umstellung auf die Version `V6` musste das Schwellwertverfahren angepasst werden. Die neue Version weißt einen grundsätzlich geringeren Pegel sowie Unterschiede zwischen den Exemplaren auf. Das Schwellwertverfahren wurde deshalb im Zuge dieser Arbeit dynamisiert. Der grundlegende Pegel eines Exemplars wird nun anhand der durchschnittlichen Amplituden der Hintergrundgeräusche bestimmt. Dazu wird der arithmetische Mittelwert der Amplituden im Bereich von 1200 Hz–2000 Hz für das aktuelle Fenster berechnet und zwischengespeichert. Der Gesamtmittelwert über die letzten 20 gespeicherten Fenstermittelwerte ergibt den durchschnittlichen Geräuschpegel des nahe zurückliegenden Zeitraums. Der dynamische Schwellwert für die Grundfrequenz f_0

[13] Im Framework wird KISS FFT genutzt, eine *Fast Fourier Transform*-Bibliothek unter dem Motto „Keep It Simple, Stupid“ (KISS): `https://github.com/mborgerding/kissfft` (letzter Abruf: 30.1.2020)

[14] Auf eine mathematische Einführung der Fourier-Transformation und Erläuterungen zur Anwendung bei der Berechnung von Frequenzspektren soll an dieser Stelle verzichtet werden. Für nähere Erläuterungen zu Theorie und Praxis in der Audioverarbeitung wird auf Butz [2015, insb. Kap. 4 u. 5] verwiesen.

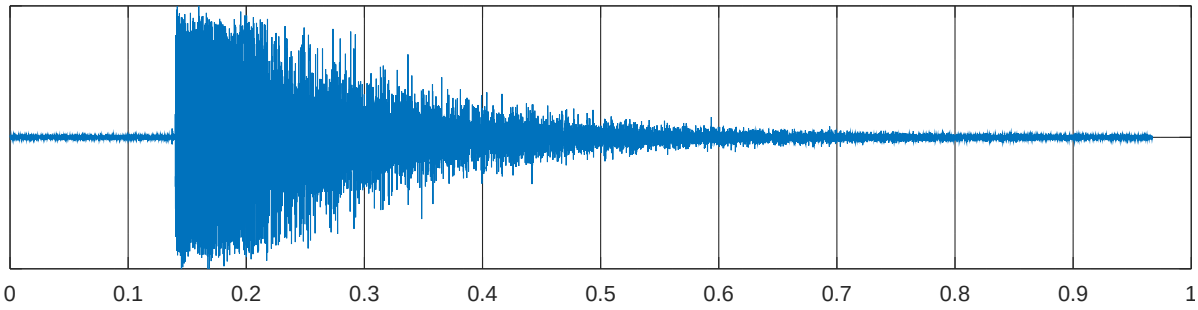
des Pfeifsignals wird dann auf Grundlage von Erfahrungswerten als das 20-fache dieses Ausreißer-bereinigten Geräuschpegels definiert. Der Schwellwert für $2f_0$ wurde durch erneute Auswertung der alten PG-Versuchsreihe auf ein Hundertstel des ersten Schwellwerts festgelegt. Die Berücksichtigung von $3f_0$ wurde aufgrund der geringen Amplitude und der damit erschwerten Messmöglichkeit bei geringerem Pegel durch Grund- und Eigenrauschen sowie Lüftergeräusche (vgl. Abbildung 7: jeweils 0 s–0,1 s) entfernt.

Das dynamisierte Detektionsverfahren kann durch Auswertung der Spielergebnisse des RoboCup 2019 als sehr stabil bezeichnet werden. In 7 Spielen wurde jeweils zum Anfang der beiden Halbzeiten sowie nach jedem Tor angepfiffen. Bei 47 gefallenen Toren ergeben sich insgesamt 61 Testpfiffe. Der Pfiff wurde dabei mit dem beschriebenen Verfahren stets erkannt. Die Entscheidung den 20-fachen Geräuschpegel als Schwellwert zu definieren ist demnach gerechtfertigt.

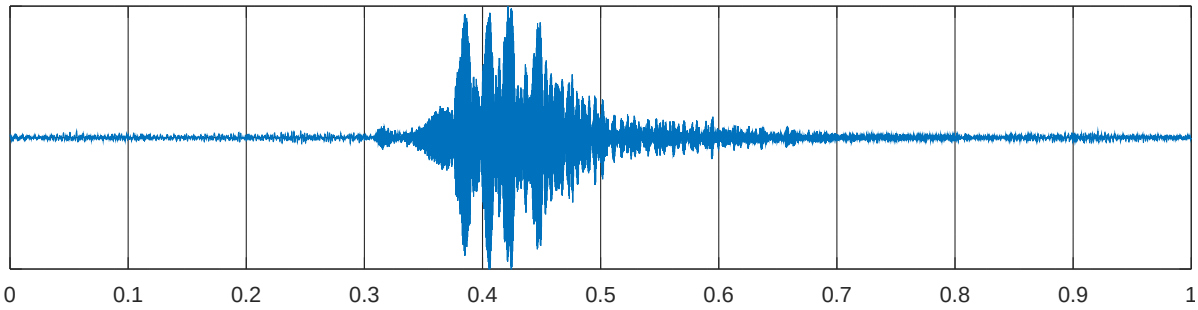
Die Detektion als binäres Klassifikationsproblem muss nun um die Bestimmung des Startzeitpunkts erweitert werden, um das angestrebte Ortungsverfahren mithilfe von Laufzeitdifferenzen anwenden zu können. In ruhiger Umgebung wäre ein Schwellwertverfahren (wie beim Knalltest aus Abschnitt 3.2) ausreichend. Dann müssten bei Detektion des Pfeiftons im Signal lediglich die einzelnen Samples sukzessive auf Überschreitung der Schwellwertamplitude getestet werden. Wie bereits erwähnt ist der WhistleDetector darauf optimiert, trotz höherer Amplituden im Frequenzbereich unter 2000 Hz den Pfeifton anhand seiner Charakteristika isoliert zu analysieren. Zudem ist der Charakter des Pfeifsignals nicht plötzlich. Während das Auftreten eines Knalls (vgl. Abbildung 7a) händisch wie algorithmisch vergleichsweise leicht und vor allem Sample-genau bestimmt werden kann, ist bereits die händische Bestimmung des Startzeitpunkt beim Pfeifsignal mehrdeutig (vgl. Abbildung 7b). Vor der Trillerphase (ab dem ersten der vier hohen Ausschläge) ist das weitaus leisere Anblasen (ab ca. 0,3 s bis zum ersten Ausschlag) zu erkennen.

Ohne die Filterfähigkeiten zu verlieren ist es mit dem WhistleDetector demnach nur möglich das Fenster des ersten Auftretens des Pfeiftons als Startzeitpunkt anzugeben. Der Anfang des Fensters ist ein Sample aus den diskreten Audiodaten S. S stellt dabei eine Abbildung $\{s|s = 0, 1, \ldots, i, \ldots, L-1\} \mapsto [-1, 1]$ dar. Der Zeitstempel T_s für einen beliebigen Sample s an der Position i ist mit der vom AudioProvider bereitgestellten Information über den Aufnahmezeitpunkt T_{in} (siehe Abschnitt 3.2) über die Abtastrate f_s durch

$$T_s = T_{in} - \frac{L - i}{f_s} \tag{6}$$



(a) Wellenform eines einzigen Händeklatschens



(b) Wellenform eines kurzen Pfeifsignals

Abbildung 7: Aufnahmen mit den NAO-Mikrofonen (Zeitskala in s)

definiert. Die doppelte Subtraktion entsteht durch das erwähnte Zurückrechnen im diskreten Audiosignal vom Aufnahmezeitpunkt aus.

Die bisher gewählten Einstellungen erreichen so eine Genauigkeit, die dem Fensterversatz von 23,22 ms (siehe Gleichung (5)) entspricht. Diese Genauigkeit kann gesteigert werden, wenn die Abtastrate auf $f_s = 44\,100\,\mathrm{Hz}$ verdoppelt, die Fenstergröße auf 512 Samples halbiert und der Festerversatz auf ein Zehntel der Größe verringert wird. So wird eine Genauigkeit von

$$\frac{512}{44\,100\,\mathrm{Hz}} \cdot \frac{1}{10} \approx 1{,}16\,\mathrm{ms} \tag{7}$$

erzielt, was in etwa dem 20-fachen der vorherigen Genauigkeit entspricht.

Nachteil der Parameteroptimierung: Für diese Werte ist das Erkennungsverfahren nicht so intensiv validiert wie die Parameter für das normale Spiel. Bei der *Directional Whistle Challenge* wurden zwar alle acht Versuche positiv erkannt, Garantien für die Spielsituation können aufgrund der Abwesenheit von externem Publikum und der insgesamt geringeren Anwesenheit von Beteiligten keinesfalls abgeleitet werden. Zudem ist der durch die höhere Abtastung von Samples und Fenstern gesteigerte Berechnungsaufwand im Spiel kritisch. Die in der Challenge sitzenden Roboter können die gesamte

Rechenleistung auf das Ortungsverfahren verwenden. Im Spiel müssen sie währenddessen laufen (oder zumindest stehen) können und es wird Bildverarbeitung zur Lokalisierung sowie zur Objekterkennung ausgeführt. Durch die häufige Berechnung der rechenintensiven DFT hat der WhistleDetector die höchste Laufzeit aller Module bei dem in dieser Arbeit präsentierten Ortungsverfahren.

4. Algorithmische Lösung

Mithilfe der theoretischen Überlegungen aus Abschnitt 2 soll nun ein Lösungsalgorithmus entworfen werden. Grundlage bildet die in Abschnitt 2.1 eingeführte Gleichung (4). Die Positionen der Empfänger P_i und P_j ist durch die Regeln der Challenge vorgegeben (siehe Abschnitt 1.1, insb. Abbildung 2). Im normalen Spiel müssten diese durch Selbstlokalisierung der Roboter bestimmt werden [Jochmann u. a., 2012], wodurch zusätzliche Positionsungenauigkeiten auftreten. Die Eintreffzeitpunkte des Pfeifsignals T_i und T_j (TOAs) können mithilfe der in Abschnitt 3 beschriebenen Methodik bestimmt und über die geteilten TeammateData den einzelnen Robotern zur Berechnung der Laufzeitdifferenzen $T_{i,j}^{\Delta}$ (TDOAs) bereitgestellt werden.

So bleibt die räumliche Position des Emitters $\vec{E}$ als einzige Unbekannte in Gleichung (4) übrig. Die Linearisierung der Gleichung nach $\vec{E}$ ist möglich und wird im folgenden Abschnitt 4.1 vorgestellt. Ein approximativer Ansatz mit dem Ziel der optimierten Behandlung von Messungenauigkeiten und Fehlerkorrektur wird anschließend in Abschnitt 4.2 erläutert. Die besonderen Anforderungen an Wartbarkeit und Flexibilität im Kontext des Roboterfußballs werden diskutiert und fließen in die Verfahrenskonstruktion ein. Das approximative Verfahren garantiert die 5-Sekunden-Beschränkung bei der Ergebnisübermittlung an die Testapplikation und ermöglicht durch geschickte Modifikation ein effizientes Verfahren zur Elimination von Ausreißern, welches in Abschnitt 4.3 erläutert wird.

4.1. Exakte Lösung durch Linearisierung

Um das beschriebene Problem in angemessener Zeit algorithmisch zu lösen, kann das Modell in ein lineares Gleichungssystem überführt werden. Die Grundidee basiert lose auf den Ausführungen in Bucher u. Misra [2002]. Ausgangspunkt bildet die Entfernungsgleichung (2). Ihre konkrete Berechnung im dreidimensionalen für den Empfänger am Ort P_i lautet

$$R_i = |\vec{P_i} - \vec{E}| = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}. \tag{8}$$

Um die Komplexität unter der Wurzel zu beschränken, wird der Ursprung des Koordinatensystems der bekannten Empfängerposition P_i auf die Position eines der Empfänger umgelegt. So ergibt sich für diesen Zentralempfänger am Ort P_0 eine vereinfachte Gleichung für die Entfernung zum Emitter:

$$R_0 = |\vec{0} - \vec{E}| = |\vec{E}| = \sqrt{x^2 + y^2 + z^2} \tag{9}$$

Werden alle benötigten Entfernungsdifferenzen aus den gemessenen Laufzeitunterschieden in Bezug zu T_0 berechnet, entfallen x_0, y_0 sowie z_0 aus allen TDOAs. Die in Gleichung (4) eingeführte Umrechnung von Entfernungs- und Laufzeitzeitdifferenzen wird so konkret zu

$$\theta_i = c_s T_{i,0}^{\Delta} = R_i - R_0 = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} - \sqrt{x^2 + y^2 + z^2}. \tag{10}$$

θ_i steht also in Verbindung mit der gemessenen TDOA und bezieht R_i stets auf den Zentralempfänger an Position P_0. Die Gleichung ist allerdings noch nicht linear. Zunächst wird die gesamte Gleichung quadriert, aufgelöst und umgeformt:

$$\begin{aligned} & R_i^2 = (\theta_i + R_0)^2 \\ \Leftrightarrow \quad & R_i^2 = \theta_i^2 + 2\theta_i R_0 + R_0^2 \\ \Leftrightarrow \quad & 0 = \theta_i^2 + 2\theta_i R_0 + R_0^2 - R_i^2 \\ \Leftrightarrow \quad & 0 = \theta_i + 2R_0 + \frac{R_0^2 - R_i^2}{\theta_i} \end{aligned} \tag{11}$$

Als erster Schritt der Linearisierung wird nun diese Gleichung für alle Empfänger ($2 \leq i < N$) von der Gleichung für den Empfänger an Position P_1 subtrahiert:

$$0 = \theta_i - \theta_1 + \frac{R_0^2 - R_i^2}{\theta_i} - \frac{R_0^2 - R_1^2}{\theta_1} \tag{12}$$

Als zweiter Schritt wird die euklidische Entfernungsgleichung aus Gleichung (8) quadriert und aufgelöst:

$$R_i^2 = x_i^2 + y_i^2 + z_i^2 - x2x_i - y2y_i - z2z_i + x^2 + y^2 + z^2 \tag{13}$$

Die Summe der quadrierten Unbekannten x^2, y^2 und z^2 am Ende der Gleichung ist nach Definition R_0^2 (siehe Gleichung (9)). Einsetzen und Umformen ergibt:

$$\begin{aligned} & R_i^2 = x_i^2 + y_i^2 + z_i^2 - x2x_i - y2y_i - z2z_i + R_0^2 \\ \Leftrightarrow \qquad & R_0^2 - R_i^2 = -x_i^2 - y_i^2 - z_i^2 + x2x_i + y2y_i + z2z_i \end{aligned} \tag{14}$$

Diese Formel für $R_0^2 - R_i^2$ kann nun die beiden Zähler aus Gleichung (12) ersetzen. Durch Auseinanderziehen der Brüche, Umordnen der Summanden und Ausklammern der Unbekannten x, y und z erhält man:

$$\begin{aligned} 0 = \quad & \theta_i - \theta_1 + \frac{-x_i^2 - y_i^2 - z_i^2 + x2x_i + y2y_i + z2z_i}{\theta_i} \\ & - \frac{-x_1^2 - y_1^2 - z_1^2 + x2x_1 + y2y_1 + z2z_1}{\theta_1} \\ \Leftrightarrow \qquad 0 = \quad & \theta_i - \theta_1 - \frac{x_i^2 + y_i^2 + z_i^2}{\theta_i} + \frac{x2x_i}{\theta_i} + \frac{y2y_i}{\theta_i} + \frac{z2z_i}{\theta_i} \\ & + \frac{x_1^2 + y_1^2 + z_1^2}{\theta_1} - \frac{x2x_1}{\theta_i} - \frac{y2y_1}{\theta_i} - \frac{z2z_1}{\theta_1} \\ \Leftrightarrow \qquad 0 = \quad & \theta_i - \theta_1 - \frac{x_i^2 + y_i^2 + z_i^2}{\theta_i} + \frac{x_1^2 + y_1^2 + z_1^2}{\theta_1} \\ & + x\left(\frac{2x_i}{\theta_i} - \frac{2x_1}{\theta_i}\right) + y\left(\frac{2y_i}{\theta_i} - \frac{2y_1}{\theta_i}\right) + z\left(\frac{2z_i}{\theta_i} - \frac{2z_1}{\theta_1}\right) \end{aligned} \tag{15}$$

Daraus ergibt sich das folgende lineare Gleichungssystem zur Lösung der Position des Emitters $\vec{E} = (x, y, z)^T$:

$$\begin{aligned} 0 &= xA_i + yB_i + zC_i + D_i \\ A_i &= \frac{2x_i}{\theta_i} - \frac{2x_1}{\theta_1} \\ B_i &= \frac{2y_i}{\theta_i} - \frac{2y_1}{\theta_1} \\ C_i &= \frac{2z_i}{\theta_i} - \frac{2z_1}{\theta_1} \\ D_i &= \theta_i - \theta_1 - \frac{x_i^2 + y_i^2 + z_i^2}{\theta_i} + \frac{x_1^2 + y_1^2 + z_1^2}{\theta_1} \end{aligned} \tag{16}$$

Das Gleichungssystem ist mithilfe des gaußschen Eliminationsverfahren [Gauss, 1865, Zweites Buch, Abschn. 3] algorithmisch lösbar. Ziel ist die Umformung in Stufenform durch vorgegebene Zeilenoperationen. Das Verfahren besitzt eine eindeutige Anweisungsstruktur. Die numerische Anfälligkeit für Rundungsfehler kann durch Spaltenpivotisie-

rung [vgl. Meister, 2015, Abschn. 3.1] eliminiert werden. Alternativ kann auch die stabilere QR-Zerlegung oder ein Splitting-Verfahren angewendet werden [Meister, 2015].

Der Berechnungsaufwand des Eliminationsverfahrens ist für n Gleichungen bei 3 Unbekannten in $O(n^3)$. Da die Anzahl der Empfänger mit konstant 5 vernachlässigbar klein ist und somit nur 3 Gleichungen entstehen, ist ein kubisches Wachstum akzeptabel im Hinblick auf die 5-Sekunden-Beschränkung.

Die große Problematik im Anwendungsfall sind die Messungenauigkeiten. Die Hyperbeln aus Abbildung 5 werden sich bei realer Datengrundlage nicht in genau einem Punkt schneiden, sodass das Gleichungssystem keine eindeutige Lösung definiert. Bei Anwendung der gezeigten Linearisierung wird diese Uneindeutigkeit durch die Subtraktion in Gleichung (12) gelöst. Nun sind jedoch alle Gleichungen nicht nur von P_0 sondern ebenfalls von P_1 abhängig. Sollte einer dieser beiden Empfänger starke Ungenauigkeit einbringen, so ist die Lösung ebenso unbrauchbar.

Eine Güteüberprüfung der Lösung kann durch Mehrfachberechnung unter Vertauschung der Empfängerindizierung geschehen. Dabei wird jedes mögliche Empfängerpaar einmal als (R_i, R_j) sowie (R_j, R_i) betrachtet. Je größer die Unterschiede der berechneten Emitterpositionen sind, desto höher war die Datenungenauigkeit. Da aber immer alle 5 Empfänger zur Lösungsbestimmung berücksichtigt werden müssen, kann keine klare Aussage über den Fehlerursprung gemacht werden. Damit sind Korrekturmöglichkeiten zunächst ausschlossen. Zudem vergrößert sich der Berechnungsaufwand für 5 Empfänger um den Faktor $5 \cdot 4 = 20$. Das Ziel des im Folgenden dargestellten approximativen Ansatzes ist es deshalb, einen Vorteil durch die Ortung mit weniger als 5 Robotern zu erreichen.

4.2. Approximative Lösung durch Greedy-Suche

Die Grundidee des approximativen Ansatzes basiert auf der Erkenntnis, dass die Regeln der *Directional Whistle Challenge* nicht darauf abzielen eine möglichst positionsgetreue Lösung zu finden. Vielmehr motivieren sie durch das Polarkoordinaten-gestützte Bewertungsverfahren (siehe Abschnitt 1.2) eine Optimierung auf die höherwertigen Informationen Richtung und Distanz sowie die Einschätzung der Feldzugehörigkeit. Die Stabilität dieser Informationen soll wichtiger sein als die mathematische Präzision. Denn eine praxisorientierte Zielsetzung ist sinnvoll im Hinblick auf die Anforderungen, die an ein Verfahren bei tatsächlicher Anwendung in einem normalen Roboterfußballspiel der SPL gestellt würden.

Ziel des approximativen Ansatzes ist es, die notwendige Anzahl an Robotern zur Generierung von Lösungen zu reduzieren. Zudem sollen eine Reihe von Parametern die Anpassung an sich ändernde Rahmenbedingung ermöglichen. Ein Verfahren mit nachvollziehbarer und dynamischer Struktur birgt einen erheblichen Vorteil, wenn Fehler auftreten. Bei Roboterfußballturnieren kann die Zeit zwischen zwei Spielen weniger als drei Stunden betragen. Tritt bei einem Spiel ein Problem auf, muss eine Verfahrensoptimierung im Idealfall innerhalb einer solchen Zeitspanne zu bewerkstelligen sein. Diese Anforderung muss beim Entwurf eines Verfahrens für den Roboterfußball stets berücksichtigt werden und stellt eine besondere Hürde dar.

Der im Folgenden präsentierte Lösungsansatz arbeitet auf Grundlage einer immer feiner werdenden Rastersuche im Raum mit Greedy-Strategie. Der endliche Suchraum ist beschränkt auf einen Suchquader S_0 der Grundfläche $30\,\mathrm{m} \times 30\,\mathrm{m}$ um den Anstoßkreismittelpunkt in einer Höhe von 1,6 m bis 1,9 m über dem Spielfeld. Durch gleichmäßige Verrasterung dieses Quaders mit einer Schrittlänge von $l_0 = 1\,\mathrm{m}$ um dessen Mittelpunkt $\vec{M}_0$ werden $31 \cdot 31 \cdot 1 = 961$ initiale Lösungskandidaten definiert.[15]

Mithilfe einer Bewertungsstrategie auf Grundlage der aus den Messdaten entstehenden Hyperbeln wird jedem Lösungskandidaten eine Fehlerabschätzung zugeordnet. Dazu muss die theoretische Laufzeitdifferenz einer Kandidatenposition $\vec{K}$ mit der tatsächlich gemessenen TDOA $T^{\Delta}_{i,j}$ verglichen werden. Anhand der geometrischen Definition aus Gleichung (4) lässt sich für eine beliebige Kandidatenposition $\vec{K}$ und zwei Empfänger an den Positionen $\vec{P}_i$ und $\vec{P}_j$ durch Subtraktion die Fehlerfunktion $\epsilon_{i,j}(\vec{K})$ definieren als

$$\epsilon_{i,j}(\vec{K}) = c_s T^{\Delta}_{i,j} - |\vec{P}_i - \vec{K}| - |\vec{P}_j - \vec{K}|. \tag{17}$$

Je kleiner $|\epsilon_{i,j}|$, desto näher liegt $\vec{K}$ an der Hyperbel der möglichen Positionen von $\vec{E}$. Für $\vec{K} = \vec{E}$ wird $\epsilon_{i,j} = 0$. Da bis hierher lediglich die TOTs zweier Empfängern betrachtet wurden, gilt der Umkehrschluss jedoch nicht. Ebenso bietet $\epsilon_{i,j}$ keine vergleichbare Güteinformation für eine Position $\vec{K}$ in Bezug zur gesuchten Emitterposition $\vec{E}$, denn auch aus $|\epsilon_{i,j}(\vec{K}_a)| < |\epsilon_{i,j}(\vec{K}_b)|$ kann *nicht* $|\vec{K}_a - \vec{E}| < |\vec{K}_b - \vec{E}|$ gefolgert werden. Eine Veranschaulichung bietet Abbildung 8. $\vec{K}_a$ ist zwar eindeutig näher an $\vec{E}$, doch $\vec{K}_b$ hat den geringeren Fehlerwert $\epsilon_{i,j}(\vec{K}_b) = 0$ bezüglich der Lage zur Hyperbel.

Um eine aussagekräftige Bewertung zu erhalten, müssen alle möglichen Empfängerpaare betrachtet werden. Wie in Abschnitt 2.1 erläutert sind in der Theorie die Messdaten aller $N = 5$ Empfänger im $\mathbb{R}^3$ für eine eindeutige Lösung notwendig. Im Unterschied

[15] Alle in diesem Abschnitt angegebenen Parameter finden sich im Quellcode in Anhang A.1 in den Zeilen 18–30.

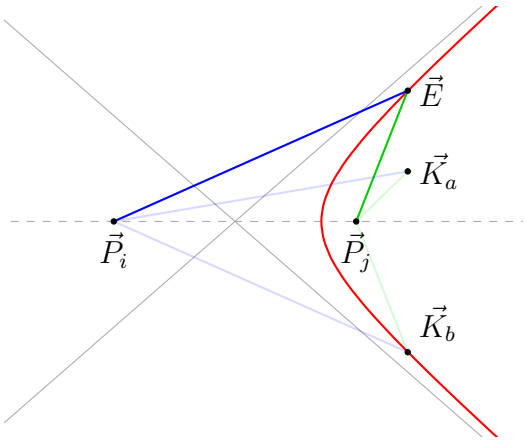


Abbildung 8: Problematische Positionierung zweier Lösungskandidaten

zum linearisierten Modell aus dem vorherigen Abschnitt 4.1 haben alle Roboter innerhalb der Berechnungsstruktur die gleiche Funktion. Dabei gilt für zwei beliebige i und j, dass $\epsilon_{i,j} = -\epsilon_{j,i}$. Zur Einschätzung der Güte ist allein der Fehler im Hinblick auf die Hyperbel entscheidend. So sind die Paare gemäß der Assoziativität vertauschbar, denn der Fehler wird durch den Betrag $|\epsilon_{i,j}|$ berechnet. So ergeben sich für N Empfänger $\frac{N(N-1)}{2}$ nötige Berechnungen der einzelnen $\epsilon_{i,j}$. Im Anwendungsfall mit 5 Robotern sind das 10 Berechnungen pro Kandidat.

Diese Überlegungen führen zur Definition der Gesamtfehlerfunktion ξ als die Summe aller Einzelfehler $\epsilon_{i,j}$ eines Lösungskandidaten $\vec{K}$ als

$$\xi(\vec{K}) = \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} |\epsilon_{i,j}(\vec{K})|. \tag{18}$$

Nun gilt auch, wenn $|\vec{K}_a - \vec{E}| < |\vec{K}_b - \vec{E}|$, dann folgt daraus $\xi(\vec{K}_a) < \xi(\vec{K}_b)$ und andersherum. Liegt ein Kandidat in Bezug zu einer der Hyperbeln sehr gut, obwohl die Entfernung zum Emitter groß ist (z.B. $\vec{K}_b$ in Abbildung 8), so ist ein relativ großer Fehler in Bezug auf die anderen Hyperbeln zu erwarten. Liegt ein Kandidat sehr nah an der gesuchten Emitterposition $\vec{E}$, so minimiert sich der Gesamtfehler ξ.

Mithilfe von $\xi(\vec{K})$ werden nun die Lösungskandidaten aus der Menge der initialen Kandidaten im Suchraster bewertet. Da die initiale Rasterung eine Schrittlänge von $l_0 = 1\,\mathrm{m}$ hat, ist die Genauigkeit des besten Kandidaten im Vergleich zur gesuchten Emitterposition erwartbar gering. Eine Verbesserung wird durch anschließende Verschiebung und Verkleinerung des Suchquaders S_0 sowie Verkürzung der Rasterschrittlänge l_0 erreicht. Dazu wird der *Verfeinerungsfaktor* ϱ definiert. Die Ausmaße des Suchquaders

werden ebenso wie die Schrittlänge mit ϱ multipliziert. Der Mittelpunkt M_1 des neuen Suchquaders wird auf die Kandidatenposition $\vec{K}_{opt}$ mit dem geringsten Fehler $\xi(\vec{K}_{opt})$ verschoben. Der verkleinerte und verschobene Suchquader $S_1 = \varrho \cdot S_0$ bestimmt dann mit der verkürzten Schrittlänge $l_1 = \varrho \cdot l_0$ ein feineres Raster und damit eine neue Menge an Lösungskandidaten.

Die Verfeinerung durch ϱ und die Bewertung der entstehenden Kandidaten wird solange wiederholt, bis die Schrittlänge l das Genauigkeitslimit l_{min} unterschreitet. Bei der *Directional Whistle Challenge* wurde die Einstellung $\varrho = 0{,}9$ und $l_{min} = 1\,\text{cm}$ genutzt. Eine schrittweise Darstellung der Verfeinerungsstrategie, die für feste Parameter deterministisch ist, bietet Tabelle 2. Die Verfeinerung der Parameter S und l an einen einzelnen Parameter zu binden schafft die angesprochene Flexibilität für den Anwendungsfall. In der Challenge kann eine hohe Genauigkeit und kleinschrittige Verfeinerung genutzt werden, da genug Rechenkapazität vorhanden ist. So ist die Gesamtanzahl von 42101 zu prüfenden Lösungskandidaten in 44 Verfeinerungsschritten sehr groß. In einem normalen Spiel müssen die Roboter viele weitere Aufgaben zusätzlich erledigen. Da die Laufzeit des Verfahrens sich direkt proportional zur Anzahl der Lösungskandidaten verhält, kann sie bei Bedarf durch Herabsetzen des Parameters ϱ verkürzt werden. Ein Wert von $\varrho = 0{,}5$ beispielsweise führt in 7 Schritten zu 6727 Lösungskandidaten (16,0 % des vorherigen Wertes), $\varrho = 0{,}1$ in 3 Schritten zu 2883 (6,8 %).

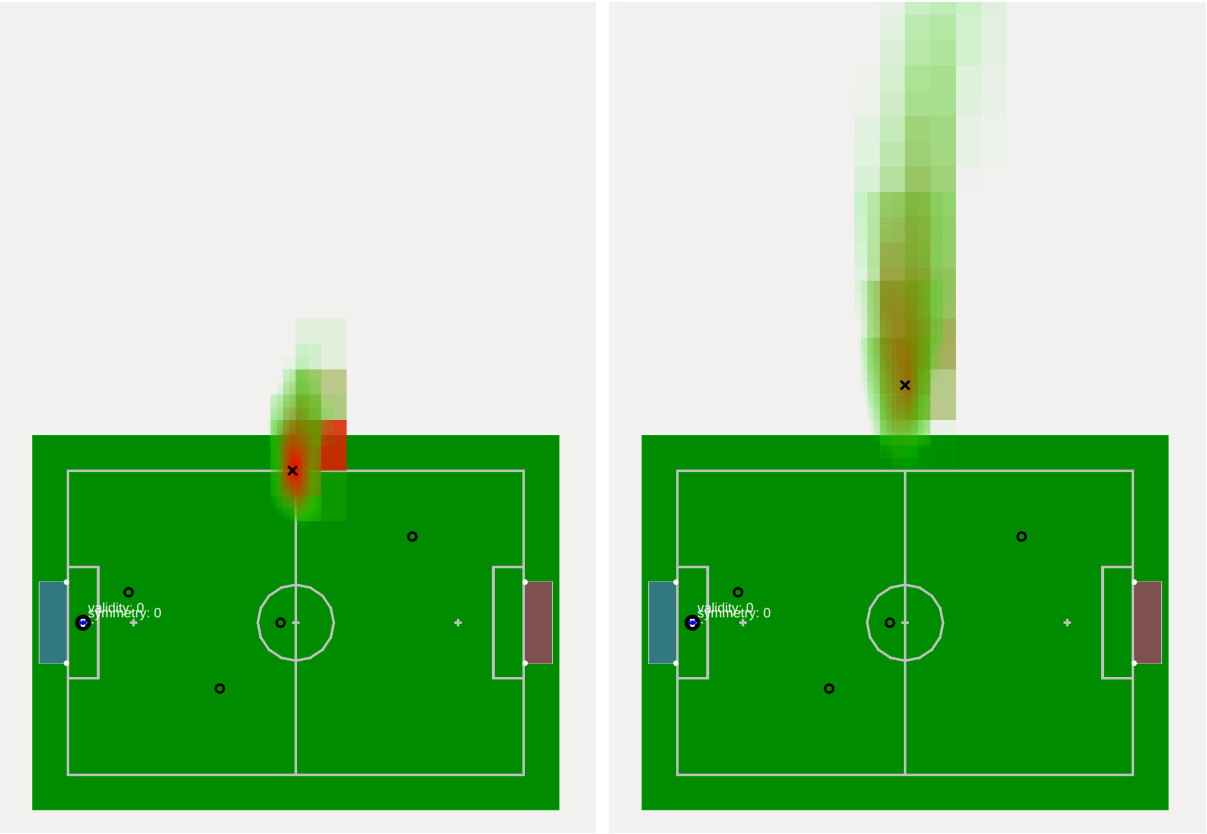
m	l_m [mm]	S_X [mm]	S_Y [mm]	S_Z [mm]	$\#\vec{K}$	$\Sigma\#\vec{K}$
0	1000.00	30000.00	30000.00	600.00	961	961
1	900.00	27000.00	27000.00	540.00	961	1922
2	810.00	24300.00	24300.00	486.00	961	2883
3	729.00	21870.00	21870.00	437.40	961	3844
⋮	⋮	⋮	⋮	⋮	⋮	⋮
41	13.30	399.08	399.08	7.98	961	40179
42	11.97	359.18	359.18	7.18	961	41140
43	10.78	323.26	323.26	6.47	961	42101

Tabelle 2: Schrittweise Darstellung der Verfeinerungstrategie

Die Genauigkeit des Ergebnisses verschlechtert sich mit der Erhöhung der Verfeinerungsgeschwindigkeit. Das scheint möglicherweise auf den ersten Blick der Intuition zu widersprechen, da die Abbruchsbedingung der Verfeinerung an die finale Schrittlänge l_{min} gebunden ist und der Mittelpunkt des Suchquaders stets *gierig* den Koordinaten des besten Kandidaten folgt. Das Problem ist das Auftreten starker Krümmungen bei

den Hyperbeln, die aus den gemessenen TDOAs entstehen. Da der Fehler $\epsilon_{i,j}$ wie bereits gezeigt keine vergleichbare Aussage über den Abstand zu $\vec{E}$ macht, muss der beste Kandidat K_{opt} eines Suchrasters S_m nicht der geometrisch am nächsten gelegene Kandidat zur gesuchten Emitterposition sein. Die potentielle Qualität der Lösung hängt demnach von der *Gierigkeit* (Greediness) des Verfeinerungsfaktors ab. Deshalb wurde bei der *Directional Whistle Challenge* mit $\varrho = 0{,}9$ ein entsprechend großer Wert verwendet.

Durch die von ϱ beschränkte Verkleinerung des Suchquaders ist es möglich, dass bei schweren Messfehlern bei der Bestimmung der TOTs (wie in Abschnitt 3 analysiert) der beste Lösungskandidat und somit der Mittelpunkt $\vec{M}_{m+1}$ des nächsten Suchquaders S_{m+1} immer weiter aus dem initialen Suchquader herauswandert. Da durch den Aufbau der Challenge davon ausgegangen werden kann, dass die Position des Pfeifenden nicht außerhalb der initialen Quadergrundfläche S_0 liegen kann, wird die Wanderung auf die erwähnten Ausmaße von 30 m × 30 m und auf eine Höhe von 1,6 m–1,9 m begrenzt. So kann auch der Fehler der endgültigen Lösung nicht unendlich groß werden.



(a) exakt (b) simulierte Messungenauigkeit

Abbildung 9: Visualisierung der Greedy-Suche im Simulator *SimRobot*[16]

[16] *SimRobot* ist ein NAO-3D-Simulator für die SPL [Röfer u. a., 2013, Abschn. 8.1]

Abbildung 9 zeigt einen simulierten Durchlauf der Positionssuche. Dazu werden zunächst durch Umkehrung der Formeln auf denen Gleichung (17) basiert für die festgelegte Position am oberen Ende der Mittellinie die theoretischen TDOAs $T_{i,j}^{\Delta}$ zu den Empfängerpositionen berechnet. Die Empfänger (in Abbildung 9b als Kreise gekennzeichnet) befinden sich auf den vorgegebenen Positionen (siehe Abschnitt 1.1, Abbildung 2). Die grünen bis roten Quadrate repräsentieren jeweils einen Lösungskandidaten $\vec{K}$ mit entsprechender Kantenlänge l_i des jeweils i-ten Suchlaufs. Je geringer der Gesamtfehler $\xi(\vec{K})$, desto mehr verfärbt sich das Quadrat von grün zu rot. Enorme Fehler von $\xi > 4000$ werden nicht gezeichnet. Abbildung 9a visualisiert einen Durchlauf in einer *perfekten* Welt. Das heißt, die Methode wurde ohne Messungenauigkeiten simuliert. Die Position des Pfeifenden (großes schwarzes Kreuz) konnte präzise erkannt werden und die gierige Suche des Kandidaten erweist sich durch die tiefrote Ballung um die Lösung als systematisch korrekt.

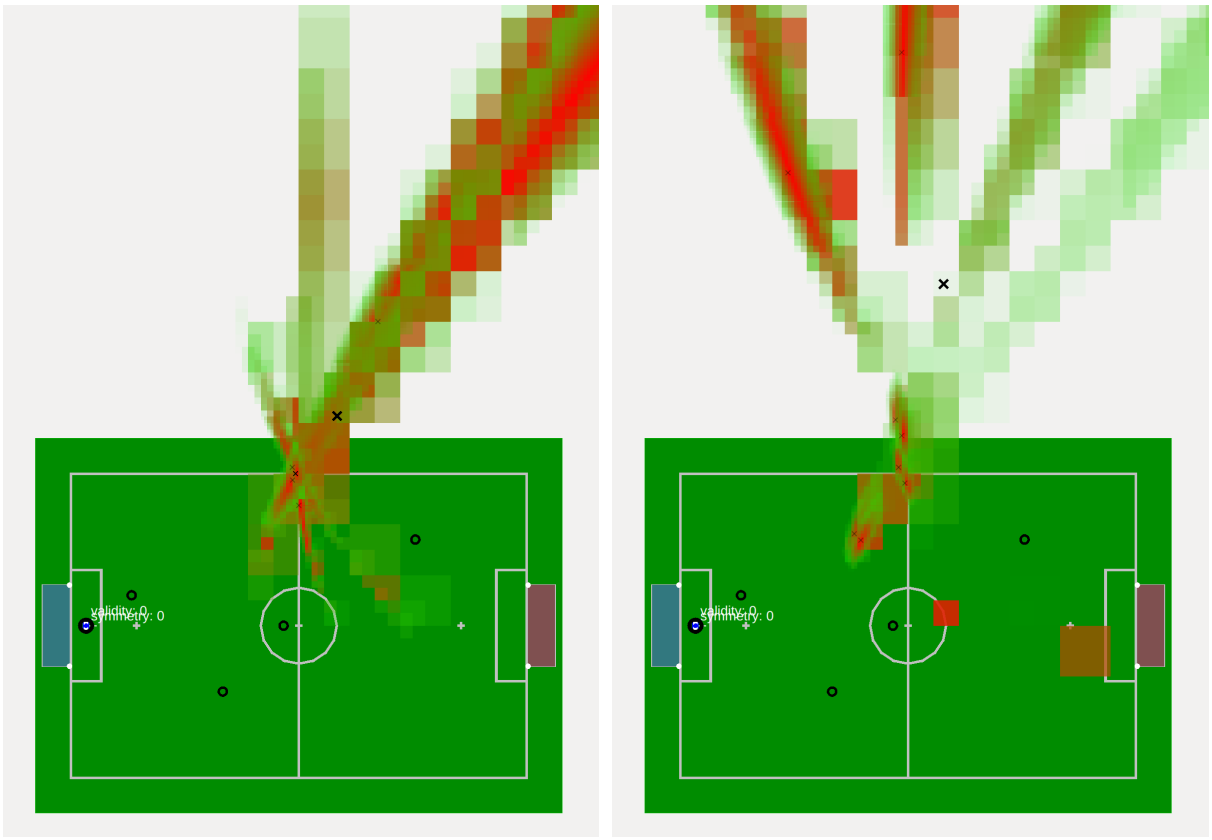
Abbildung 9b hingegen zeigt eine klare Abweichung von der korrekten Lösung. Hier wurden Messungenauigkeiten aus einer Normalverteilung um 0 mit Varianz $\sigma^2 = 1\,\mathrm{ms}$ simuliert und zu den theoretischen TDOAs addiert (näheres folgt in Abschnitt 5.2). Das wirkt sich wie erwartbar direkt auf die Güte der Lösung aus. Die nicht so tiefroten Suchquadrate visualisieren den insgesamt höheren Gesamtfehler ξ der einzelnen Kandidaten. Die Ausweitung des grünen Bereichs verdeutlicht die größere Unsicherheit des Ergebnisses.

Die korrekte Erkennung der Zugehörigkeit zum eigenen Feld, die durch die Regeln mit genau einem Punkt belohnt wird, prüft, ob die Lösungskoordinaten innerhalb des eigenen Feldes liegen. Das Spielfeld hat eine Größe von $9\,\mathrm{m} \times 6\,\mathrm{m}$. Der Schiedsrichter steht dabei meist auf einer beliebigen Seite an der Mittellinie, aber außerhalb der Spielfeldbegrenzung. Die Spielregeln [TC, 2019a] legen diese Position allerdings nicht fest. Die Schiedsrichter der Liga haben sich auf diese symmetrische Position geeinigt, damit keines der Teams bevorteilt wird. Das vorgestellte Verfahren ordnet einen Pfiff, der innerhalb einer zentrierten Fläche von $10{,}4\,\mathrm{m} \times 7{,}4\,\mathrm{m}$ um den Anstoßkreismittelpunkt geortet wird, als zum eigenen Feld gehörig ein.

4.3. Eliminierung von Ausreißern

Die Methodik aus Abschnitt 4.2 liefert für fünf Empfänger gute Ergebnisse, wenn die messbedingten Abweichung der TOAs (vgl. Abschnitt 3) hinreichend gering sind. Bei schwerwiegenden Messfehlern, wie beispielsweise in Abbildung 9b gezeigt, ist eine sinn-

volle Lösung aufgrund der Informationslage fast unmöglich. In diesem Abschnitt wird ein neuartiger Ansatz vorgeschlagen damit umzugehen.



(a) exakt (b) simulierte Messungenauigkeit

Abbildung 10: Visualisierung der Ausreißerbehandlung in *SimRobot*

Die Grundidee besteht darin, die Anzahl der Roboter zur Bestimmung einer möglichen Position erneut zu reduzieren. Der beschriebene Ansatz der Summierung ξ der Einzelfehler $\epsilon_{i,j}$ aus Gleichung (18) für alle möglichen Empfängerpaare wird weiterhin genutzt. Zur Lösungsfindung werden jedoch nur noch drei der fünf zur Verfügung stehenden Roboter betrachtet. Die Beschränkung auf eine Höhe von 1,6 m bis 1,9 m wird hierbei als eine Quasibeschränkung auf eine Ebene gesehen. Diese Informationsgrundlage ermöglicht theoretisch (siehe Abschnitt 2.1) eine 2D-Lösung mithilfe einer Plausibilitätsannahme.

Das Verfahren wird nun für alle möglichen Robotertripel genau einmal ausgeführt. Bei fünf Robotern entstehen $\binom{5}{3} = 10$ unterschiedliche Tripel. Jedes Tripel ist dann unabhängig von möglichen Messfehlern der beiden übrigen Empfänger. In Abbildung 10a sind die entstehenden 10 Tripellösungen einer perfekten Simulation durch kleine schwarze Kreuze markiert. Die Gesamtlösung (großes schwarzes Kreuz) ist dann das arithmetische Mittel der Tripellösungen. Diese Lösung kann wie in Abbildung 10a erkennbar bereits in

der perfekten Simulation eine gewisse Abweichung besitzen. Dieser Umstand wird hier akzeptiert, da bei Auftreten eines Fehlers sich nur die Güte der vom Fehler betroffenen Tripellösungen weiter verschlechtert. Das arithmetische Mittel ist hingegen eher konstant und ermöglicht trotz aller Ungewissheiten eine Aussage über die Feldzugehörigkeit zu treffen. Außerdem behebt es das Problem der fehlenden Plausibilitätsannahme. In Abbildung 10b sind zwei freistehende rote Quadrate in der rechten Feldhälfte (am Mittelkreis sowie am Elfmeterpunkt) erkennbar, die trotz relativ geringem Fehler ξ nicht zum Tripellösungskandidaten gewählt wurden. Sie sind auf die Uneindeutigkeit der Lösung in 2D bei nur 3 Robotern zurückzuführen.

Die beiden Abbildungen 10a und 10b bieten visuell einen guten Eindruck der zugrunde liegenden Hyperbeln. Vor allem die Problematik der in die Länge gezogenen Lösungsbereiche ist deutlich erkennbar. Zukünftige Verbesserungen des Verfahrens könnten auf den implizierten Hauptachsen basieren und auf Grundlage ihrer Schnittpunkte Aussagen über die Güte möglicher Lösungen treffen. Mit Blick auf Abbildung 10a und dem Vorhandensein ziemlich exakter Tripellösungen stellt sich zudem die Frage, ob eine Clusteranalyse der Tripellösungen bessere Ergebnisse liefern könnte als das arithmetische Mitteln. Die in der Praxis auftretenden Verteilungen erwiesen sich bisher als zu unvorhersehbar um eine sinnvolle Auswahl für einen Algorithmus zu treffen. Das arithmetische Mittel wird hier als stabiler gegen ungünstige Konstellationen gesehen. Die Wahl der Gesamtlösung aus der Menge der möglichen Cluster ist eine ebenso problematische Entscheidung.

5. Evaluation

In diesem Abschnitt wird die Genauigkeit des Verfahrens quantitativ wie qualitativ evaluiert und mit den Verfahren der anderen teilnehmenden Teams an der diesjährigen *Directional Whistle Challenge* verglichen. Hauptaugenmerk liegt dabei auf der Analyse von Vor- und Nachteilen verschiedener Verfahren. Über deren exakte Methodik kann allerdings größtenteils nur gemutmaßt werden. Dem Autor ist allerdings bekannt, dass alle anderen Teams auf Angulationsverfahren gesetzt haben. Dadurch können einige eindeutige und weiterreichende Erkenntnisse, vor allem im Vergleich mit dem vorgestellten laterationsbasierten Ortungsverfahren, gewonnen werden.

Die praktische Evaluation basiert vollständig auf den offiziellen Logdaten[17] des *Directional Whistle Tester* und wird im nachfolgenden Abschnitt 5.1 erläutert. Im anschließen-

[17] Online abrufbar auf der offiziellen SPL-Website: `https://spl.robocup.org/wp-content/uploads/2019/07/DirectionalWhistleTesterLog.txt` (letzter Abruf: 30.1.2020)

den Abschnitt 5.2 wird dann die grundlegende Methodik des vorgestellten Algorithmus mit Mitteln der Statistik und Computersimulation tiefgehender untersucht.

5.1. Praktische Evaluation

Im offiziellen Endergebnis der *Directional Whistle Challenge* erreichten die *NaoDevils Dortmund* den zweiten Rang (vgl. Einleitung, Tabelle 1). Dazu wurden jeweils acht Versuche von unterschiedlichen Pfiffpositionen mit jedem Team durchgeführt. Die Positionen waren dabei für alle Teams dieselben, wurden allerdings stets in einer neuen zufälligen Reihenfolge eingenommen (vgl. Abschnitt 1).

Abbildung 11 zeigt eine visualisierte Darstellung der Logdaten unserer[18] Versuche. Die blauen Punkte markieren die Lösungen, die vom präsentierten Ortungsverfahren erstellt und an die Testapplikation übermittelt wurden. Die mit ihnen verbundenen roten Punkte sind die dazugehörigen tatsächlichen Pfiffpositionen. Diese kurz vor dem Wettbewerb festgelegten Positionen liegen zur Hälfte innerhalb und zur Hälfte außerhalb des Spielfeldes. Die vergebenen Wertungspunkte für jeden einzelnen Versuch (siehe Abschnitt 1.2) stehen an den jeweiligen Verbindungsstrecken. Die grauen Dreiecke entsprechen der Anzahl und Positionierung der genutzten Roboter. Alle Lösungen wurden mithilfe der in Abschnitt 4.3 vorgestellten Ausreißereliminierung erzeugt.

Der erste Blick auf Abbildung 11 lässt vermuten, dass das Prinzip der Punktevergabe einen besonderen Einfluss auf den Ausgang der Challenge hatte, denn die Lösungen scheinen auf den ersten Blick „eher schlecht“ zu sein. Die Abstände zwischen roten und blauen Punkten der einzelnen Versuche korrelieren augenscheinlich nicht mit der Höhe der Punktzahlen.

Diese Diskrepanz muss genauer betrachtet werden. Tabelle 3 zeigt eine Aufschlüsselung der Gesamtpunktzahl sowie eine Statistik zur geometrischen Gesamtdistanz des Fehlers, sprich der Summe der Abstände der blauen zu den roten Punkten. Erstaunlich ist, dass unsere Ortung hier mit einer Gesamtdistanz von 84,67 m den höchsten Gesamtfehler aufweist. Bei den anderen Teams korreliert die Platzierung mit der Gesamtdistanz. Lediglich *Berlin United*, der Sieger der Challenge, ist einen guten Meter schlechter als die Drittplatzierten *TJArk*.

Die Frage stellt sich, warum wir im kompetitiven Vergleich so stark vom Punktesystem profitiert haben. Die korrekte Feldzugehörigkeit liegt mit 4 Feldpunkten exakt im Durchschnitt und ist genauso gut wie bei den drei schlechtesten Teilnehmern. Zieht man diese

[18] Zur besseren Lesbarkeit wird in diesem Abschnitt „uns“ bzw. „wir“ für die Zugehörigkeit des Autors zu den *NaoDevils Dortmund* benutzt und bezieht sich demnach auf das vorgestellte Ortungsverfahren.

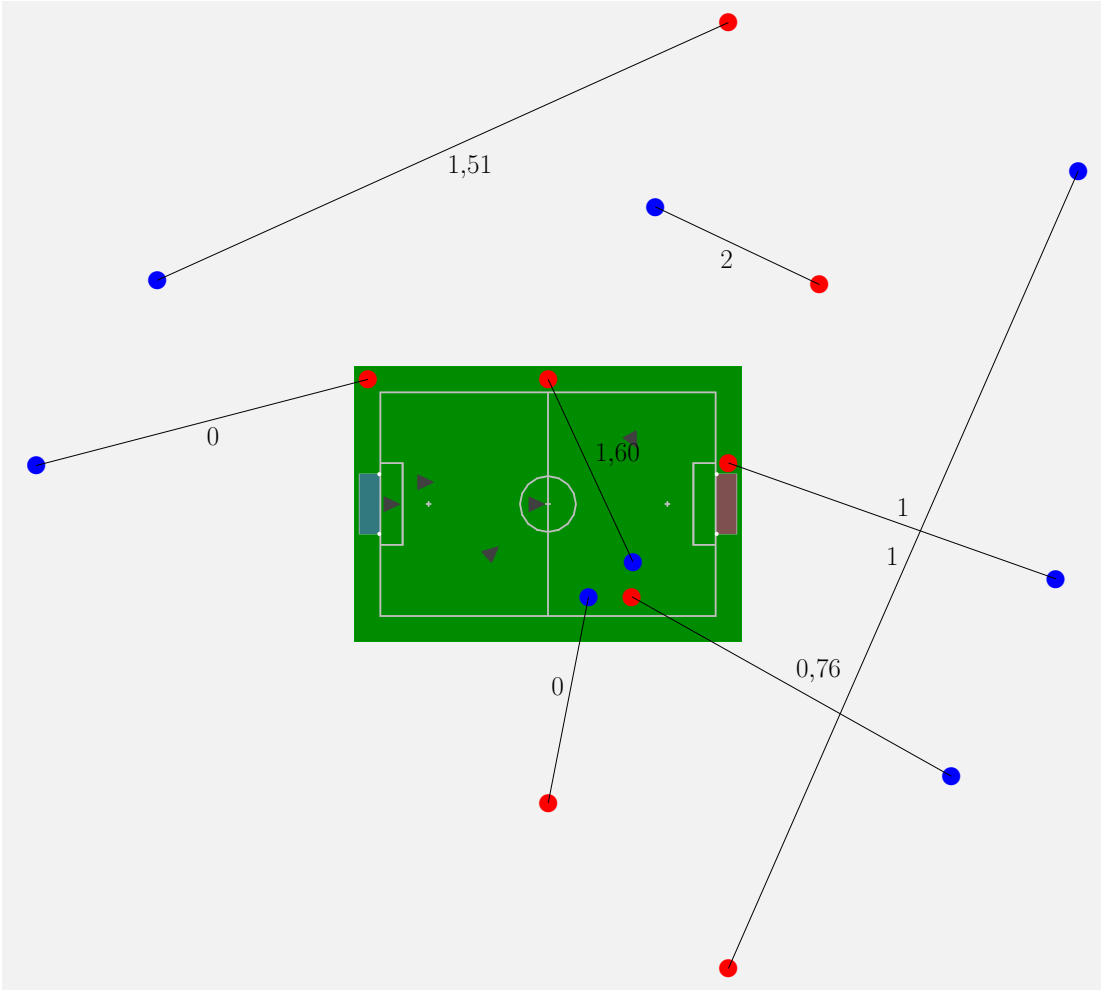


Abbildung 11: Lösungen (blau) der *NaoDevils Dortmund* in Relation zum Spielfeld

Punkte von der Gesamtpunktzahl ab, so bleiben die Maßpunkte für die Richtungs- und Distanzmaße nach Gleichung (1) übrig. Diese Art der Bewertung bezieht die konkrete Schwierigkeit der Position und die qualitative Güte der erstellten Information ein, indem die berechneten Winkel- und Längenwerte prozentual behandelt werden. Das heißt, bei größerem Abstand darf auch ein größerer Fehler zur Erreichung der gleichen Punktzahl gemacht werden. Durch die gute Maßpunktzahl von 3,87 haben wir uns von den Teams unterhalb der Podestplätze abgesetzt.

5.1.1. Einschätzung der anderen Teams

Um unseren Vorteil durch die Maßpunkte besser verstehen zu können, werden zunächst in Abbildung 12 die geloggten Lösungsversuche der anderen Teams visualisiert. Die Darstellungsweise funktioniert analog zu Abbildung 11. Die geometrische Anordnung der

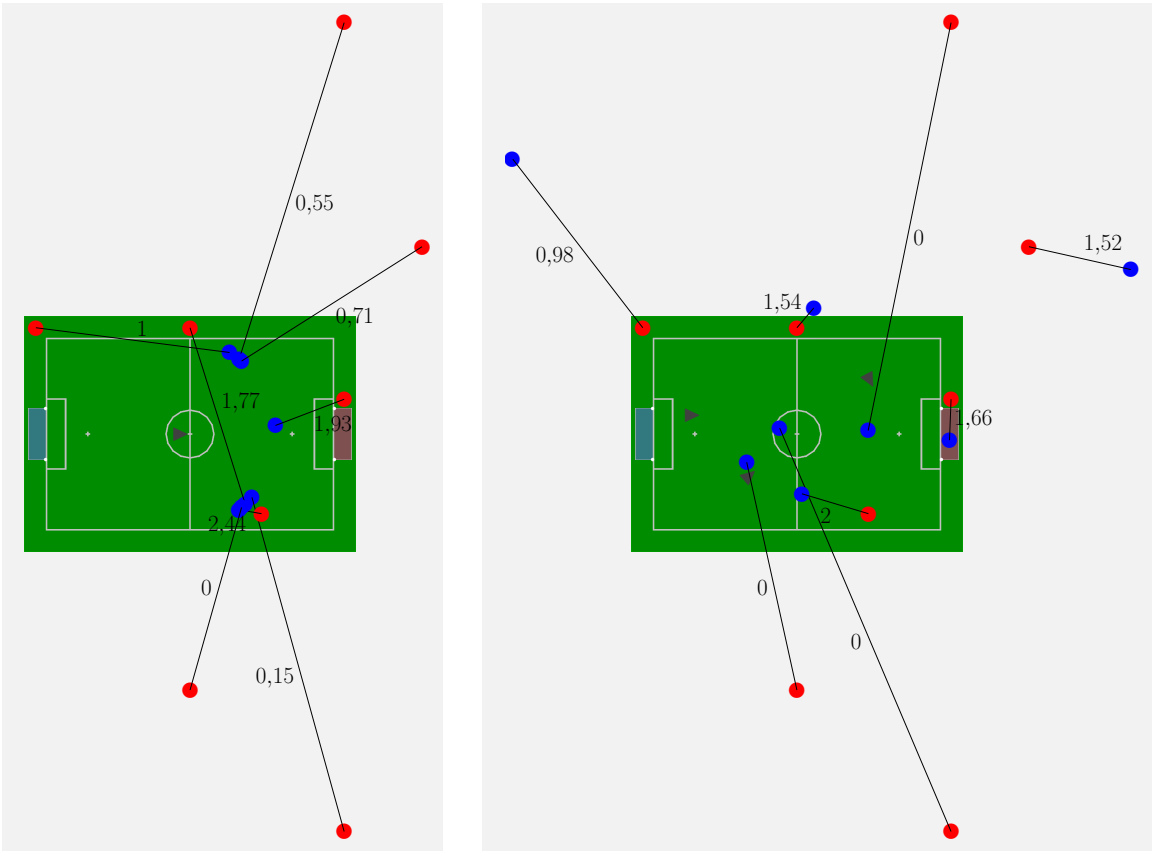
Team	Punkte	ΣDistanz [m]	Feldpkt.	Maßpkt.	#Roboter	ØZeit
Berlin	8,55	49,69	4	4,55	1	3991
NaoDevils	7,87	84,67	4	3,87	5	4603
TJArk	7,70	48,51	3	4,70	3	3158
HULKs	7,39	57,42	5	2,39	4	2927
Bembelbots	6,54	63,28	4	2,54	3	4589
HTWK	5,74	72,81	4	1,74	1	3797
B-Human	4,00	73,31	4	0,00	1	4415

Tabelle 3: Statistische Analyse der Logdaten

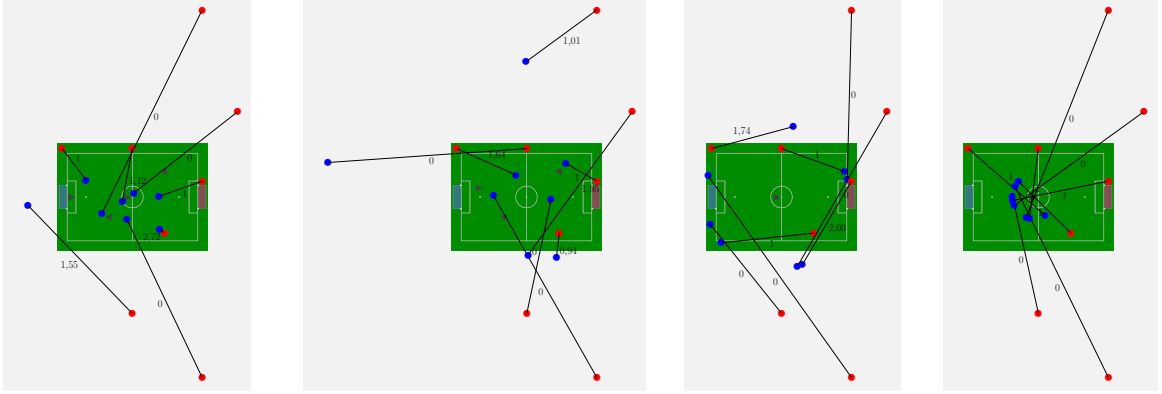
Lösungen der siegreichen Berliner (Abbildung 12a) liegt auf einem Kreis um den Roboter mit der Nummer 2 (vgl. Abschnitt 1.1, Abbildung 2). Da *Berlin United* nur diesen einen Roboter verwendet hat, liegt es nahe, dass hier der Versuch einer Richtungsortung des Pfeifsignals stattgefunden hat und der Schnitt der Richtung mit einem definierten Umkreis als Lösung betrachtet wurde. Möglicherweise wurde der Suchbereich sogar noch weiter eingeschränkt, zum Beispiel auf Positionen in der rechten Feldhälfte also vor dem Roboter. Das würde die beiden großen Punktverlust durch falsche Richtungsortung der Pfiffpositionen auf Höhe der Mittellinie sowie an der Ecke links oben erklären. Aufgrund der anderen guten Richtungsmaßpunkte hat *Berlin* eine insgesamt sehr gute Maßpunktzahl erreicht. Dazu kommen hohe Distanzmaßpunkte für die Positionen nahe des implizierten Kreises. Durch die Einschränkung auf Lösungen im eigenen Feld wurden automatisch auch die Hälfte der möglichen Feldpunkte erreicht. Das klare Siegesrezept von *Berlin United* war, dass sie nur bei einem einzigen Versuch keine Punkte geholt haben.

Das Team *B-Human* zeigt in Abbildung 12f ebenso eine Kreisanordnung der Lösungsversuche und scheint einen ganz ähnlichen Ansatz verfolgt zu haben. Jedoch war die Richtungsortung nicht so stabil wie die von *Berlin United*. Das Team *HTWK* hat realitätsnähere Lösungen forciert, indem sie typische Positionen der Schiedsrichter am Spielfeldrand angenommen haben. Abbildung 12e zeigt, dass die blauen Punkte alle nahe der Seitenlinien positioniert sind. Vor allem für die entfernteren Positionen bringt dieser Ansatz selbstverständlich Nachteile.

Die Lösungen der *HULKs* (Abbildung 12c) sammeln sich hingegen in der Mitte des Feldes. Die Pfiffposition unten rechts innerhalb der Spielfeldrandmarkierungen weist darauf hin, dass Lösungen dort sehr genau sind, denn diese Lösung erreichte mit 2,72 die höchste Punktzahl im gesamten Wettbewerb.Da sonst alle Pfiffe von einer Position



(a) *Berlin United* (b) *TJArk*



(c) *HULKs* (d) *Bembelbots* (e) *HTWK* (f) *B-Human*

Abbildung 12: Lösungsversuche (blau) aller teilnehmenden Teams

jenseits der Seitenlinien erfolgt sind, brachte dieser Ansatz, der offensichtlich optimiert ist Positionen innerhalb des Feldes zu erkennen, nicht viele Punkte ein. Die Bembelbots erreichten ein ordentliches Ergebnis mit 2,54 Maßpunkten und das mit nur drei Robotern.

Den dritten Platz erreichte *TJArk*. Mit drei Robotern wurden vier beachtlich gute Versuche mit einer Punktzahl je größer 1,5 erzielt. Über die Art des Verfahrens ist leider nichts genaues bekannt. Hervorzuheben ist, dass *TJArk* die höchste Gesamtmaßpunktzahl mit 4,70 erzielt hat. Dennoch sind auch drei Versuche punktlos geblieben. Der an sich gute Versuch mittig über dem Feld sollte besonders beachtet werden (in Abbildung 12b mit einer Punktzahl von 1,54 markiert), denn dieser ist keinen Meter von der tatsächlichen Position entfernt. Die Punktzahl jedoch ist allein durch Maßpunkte entstanden, denn *TJArk* hat die Zugehörigkeit zum eigenen Feld negativ beantwortet. Das ist eine ungünstige Fehleinschätzung, denn der blaue Punkt ist nicht weit vom Grün des Rasens entfernt. In der Realität eines RoboCups kann dort de facto kein Schiedsrichter eines anderen Feldes stehen, da die Grünfläche typischerweise (wie auch diesmal) von Banden und Tischen umstellt ist. Hätte *TJArk* hier auf eigene Feldzugehörigkeit entschieden, wäre die Punktzahl für den Versuch auf 2,54 gestiegen. Damit hätten sie ihre Gesamtpunktzahl um einen ganzen Punkt verbessert und wären sogar Sieger des Wettbewerbs geworden. Dieser Umstand ist ein perfektes Beispiel für die entscheidende Rolle einer realitätsbezogenen Modellierung im Roboterfußball.

5.1.2. Rückschlüsse für eigenes Ergebnis

Es steht weiterhin die Frage im Raum, wie unser Algorithmus den zweiten Platz geholt hat. Dies wird nun durch weitere Aufschlüsselung der Punktevergabe gezeigt. Dass das präsentierte Verfahren die schnellste durchschnittliche Antwortzeit gezeigt hat (Tabelle 3, Spalte ØZeit), sei hier nur am Rande erwähnt.

Die Erklärung dafür, dass wir als Zweitplatzierte die größte Gesamtfehlerdistanz haben, findet sich in der gesonderten Betrachtung der Versuche mit Pfiffpositionen auf dem eigenen Feld gegenüber derer außerhalb. Die Gesamtpunktzahl ist im ersten Fall die niedrigste aller Teams wie an Tabelle 4 abzulesen ist. Dabei liegt das Problem eher bei den Feldpunkten als bei den Maßpunkten. Tabelle 5 offenbart, wo die vergleichsweise große Stärke unseres Ansatzes liegt: Wir konnten uns mit 4,51 Punkten für die Versuche außerhalb des eigenen Feldes weit von den anderen Teams absetzen, vor allem von der starken Konkurrenz durch *Berlin United* und *TJArk*. Wir sind das einzige Team, das hier 3 Feldpunkte erreichen konnte. Bei den Maßpunkten kann nur *Berlin United* mithalten.

Unser Ansatz ist demnach eindeutig besser geeignet für die Messung auf größeren Distanzen. Der Grund dafür liegt in der unterschiedlichen Methodik. Während die anderen Teams vollständig auf Angulation gesetzt haben, nutzen wir die Lateration (siehe Abbildung 3a und 3b). Die prozentuale Maßpunkteformel aus Gleichung (1) stellt für

Team	Punkte	ΣDistanz [m]	Feldpkt.	Maßpkt.	#Roboter
Berlin	7,14	15,00	4	3,14	1
NaoDevils	3,36	33,84	1	2,36	5
TJArk	6,18	10,99	2	4,18	3
HULKs	5,84	10,08	4	1,84	4
Bembelbots	3,58	22,44	2	1,58	3
HTWK	5,74	23,68	4	1,74	1
B-Human	4,00	23,75	4	0,00	1

Tabelle 4: Statistische Analyse der Versuche *innerhalb* des Spielfeldes

Team	Punkte	ΣDistanz [m]	Feldpkt.	Maßpkt.	#Roboter
Berlin	1,41	34,69	0	1,41	1
NaoDevils	4,51	50,83	3	1,51	5
TJArk	1,52	37,52	1	0,52	3
HULKs	1,55	47,34	1	0,55	4
Bembelbots	2,96	40,84	2	0,96	3
HTWK	0,00	49,13	0	0,00	1
B-Human	0,00	49,56	0	0,00	1

Tabelle 5: Statistische Analyse der Versuche *außerhalb* des Spielfeldes

entfernte Punkte zwar einen Ausgleich dar, doch vor allem durch akustische Reflexionen wird die Angulation über dem Maß erschwert. Die Einschätzung der Entfernung ist hier bei der Lateration stabiler.

Zudem erstaunt es, dass wir als einziges Team das Potential der Roboteranzahl ausgeschöpft haben. Alle anderen nutzten weniger als fünf Roboter und waren somit nicht in der Lage aus der Überbestimmtheit (vgl. Abschnitt 2.2) Nutzen zu ziehen. Die Implementierung einer solchen Ergebnisoptimierung, die bei uns durch die Fehlereliminierung aus Abschnitt 4.3 Anwendung fand, hätte mutmaßlich zu besseren Ergebnissen bei den anderen Teams geführt.

Abschließend lässt sich feststellen, dass das Hauptziel, die korrekte Einschätzung der Feldzugehörigkeit, bei allen Teams verfehlt wurde. Eine erreichte Gesamtfeldpunktzahl von 4 bei fast allen Teams (einmal 5 bei den *HULKs*) bei insgesamt 8 Versuchen ist genauso gut wie eine zufällig geratene Lösung und somit unbrauchbar. Da die Angulationsverfahren aber für feldnahe Punkte und die Lateration für entfernte Punkte jeweils eindeutig besser abgeschnitten haben, ist die Zusammenführung beider Ansät-

ze der nächst logische Schritt, um das Ziel der Einschätzung der Feldzugehörigkeit zu erreichen. Allerdings müssen die Beschränkungen der NAO-eigenen Audiohardware überwunden werden. Ohne hinreichende Synchronisation wird die Lateration an sich nicht weiter optimiert werden können.

5.2. Simulierte Evaluation

Die folgende Analyse soll Aussagen zur Robustheit und Anwendbarkeit ermöglichen. Um das approximative Verfahren auf seine Genauigkeit zu untersuchen, wird eine große Menge unterschiedlicher Ausprägungen aller beteiligter Variablen simuliert. Der Vorteil gegenüber der praktischen Evaluation ist hierbei, dass eine Größenordnung erreicht wird, die in der Realität durch ein Versuchsexperiment nicht annähernd abgedeckt werden kann. So können Problemfelder eruiert und systematisch eingegrenzt werden.

5.2.1. Simulationskonzept

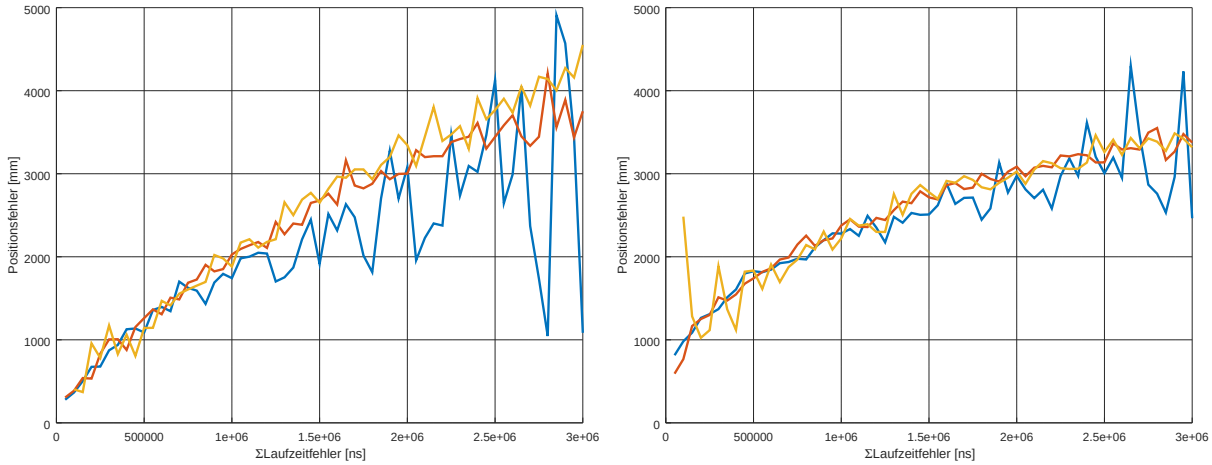
Die Simulation benutzt das implementierte Modul DirectionDetector (siehe Abschnitt 3, Abbildung 6) und führt mit zufallsgenerierten Eingaben Stichprobentests durch. Die benötigten und zwischen den Robotern geteilten Zeitstempel der Pfifferkennung aus der Representation WhistleData werden künstlich erzeugt. Zunächst wird eine simulierte Emitterposition $\vec{E}_{sim}$ generiert. Diese wird zufällig gleichverteilt aus einem quaderförmigen Raum um den Anstoßpunkt mit zentrierter Grundfläche von $10\,\mathrm{m} \times 10\,\mathrm{m}$ und einer Höhe zwischen 1,6 m–1,9 m gezogen. Um die Dimensionalität der Simulation zu begrenzen, werden die fünf Roboterpositionen P_i mit $i \in \{1, 2, 3, 4, 5\}$ fest auf die Positionen der Challenge gesetzt (siehe Abschnitt 1.1, Abbildung 2). Durch Umkehrung der Gleichung (2) können dann die Zeitstempel T_i als TOAs simuliert werden und dienen als Eingabe für das Modul.

Um eine Statistik zum Verhalten des Verfahrens bei auftretenden Abweichungen zu erstellen, wird in 50 % der Fälle T_i manipuliert, um eine Messungenauigkeit zu simulieren. So entstehen sowohl Datenpunkte einer perfekten Welt ohne Ungenauigkeiten, als auch solche mit eins, zwei, drei, vier und fünf manipulierten Stempeln. Ein manipulierter Stempel T_i^{err} wird aus der Normalverteilung $\mathcal{N}(T_i, 1\,\mathrm{ms})$ gezogen. Der Erwartungswert $\mu = T_i$ sorgt dabei für eine normalverteilte Verschiebung um den Ursprungswert. Durch die Wahl von $\sigma = 1\,\mathrm{ms}$ gilt in 68,27 % der Fälle, dass der *Laufzeitfehler* unter $|T_i - T_i^{err}| < 1\,\mathrm{ms}$ liegt. So bleibt der Fehler in der selben Größenordnung wie die realitätsbasierte Fehlerabschätzung aus Gleichung (7).

Um den Einfluss des Verfeinerungsfaktors ϱ zu untersuchen, wird pro erzeugter Position $\vec{E}_{sim}$ der Faktor auf einen gleichverteilten Zufallswert $\varrho \in [0{,}1; 0{,}9]$ gesetzt. Zudem wird das Verfahren für jede Position einmal mit und einmal ohne Ausreißereliminierung (siehe Abschnitt 4.3) angewandt, um vergleichende Rückschlüsse über die vorgeschlagene Ausreißerbehandlung zu ziehen.

5.2.2. Auswertung

Mit dem beschriebenen Stichprobenprinzip wurden insgesamt 262 500 Lösungen erzeugt. Als Hauptkriterium für die Bewertung einer Lösung wird im Folgenden der *Positionsfehler* betrachtet. Dieser wird definiert als der zweidimensionale euklidische x-y-Abstand zwischen der simulierten Emitterposition $\vec{E}_{sim}$ und der berechneten Lösungsposition. Die Höhe wird nicht einbezogen, da die Position des Schiedsrichters auf der Ebene des Spielfeldes von Interesse ist.



(a) ohne Ausreißerbehandlung (b) mit Ausreißerbehandlung

Abbildung 13: Zusammenhang von Laufzeitfehler und Positionsfehler bei einem (blau), zwei (rot) bzw. drei (gelb) manipulierten Zeitstempeln

Zunächst soll der in Abbildung 13 dargestellte Zusammenhang zwischen dem entstandenen Positionsfehler und dem simulierten Laufzeitfehler untersucht werden. Um die große Datenmenge sinnvoll visualisieren zu können, werden die Stichproben in Klassen zusammengefasst. Die Datenpunkte werden dabei der Summe der fünf Laufzeitfehler nach in 60 äquidistante Intervallklassen zwischen 0 ms–3 ms unterteilt. Innerhalb der Klassen bildet das arithmetische Mittel über alle Positionsfehler der eingeordneten Datenpunkte den erwarteten Gesamtfehler. Diese repräsentative Größe wird dann als Kurve im Diagramm visualisiert.

Abbildung 13a und 13b zeigen die Ergebnisse für das Verfahren einmal mit und einmal ohne Behandlung von Ausreißern. Die blaue Kurve zeigt die Datenpunkte, bei denen genau einer der fünf Zeitstempel manipuliert wurde. Bei der roten und gelben Kurve wurden jeweils genau zwei beziehungsweise drei manipuliert. Die Diagramme zeigen, dass die Anzahl der manipulierten Stempel in beiden Fällen keinen Einfluss auf den erwarteten Positionsfehler hat. Lediglich die Summe des Fehlers steht in einem positiven Zusammenhang. Die großen Schwankungen der blauen Kurve bei wachsendem Gesamtlaufzeitfehler erklären sich durch die geringere Anzahl an Stichproben in diesem Bereich. Kleinere Absolutfehler sind aufgrund der Normalverteilung wahrscheinlicher. Die gelbe Kurve zeigt dementsprechend eine Schwankung in der Nähe des Ursprungs.

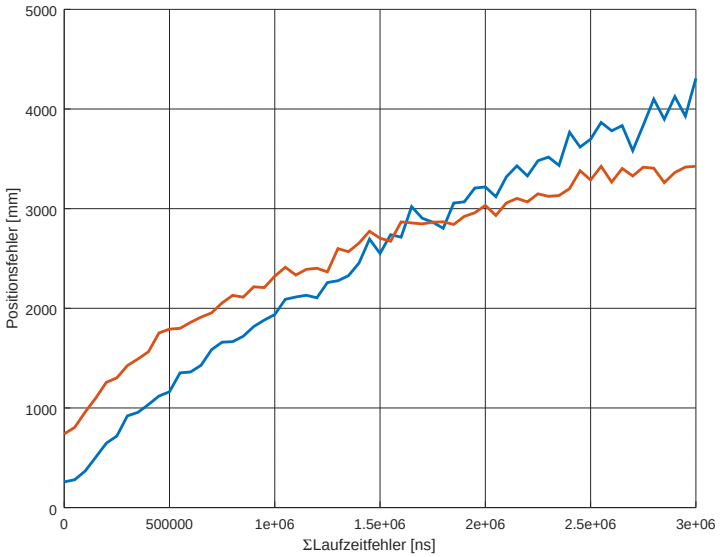
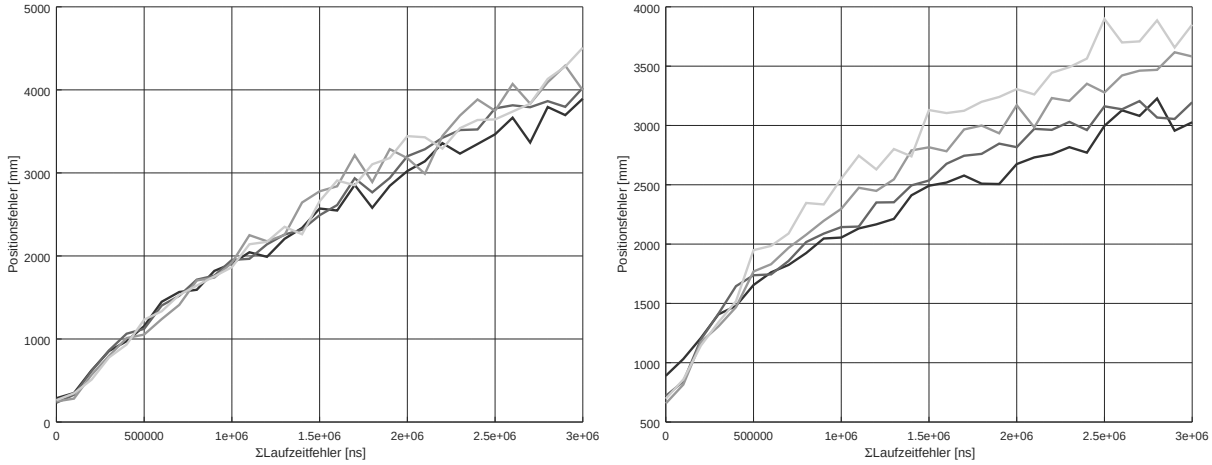


Abbildung 14: Vergleich des Zusammenhangs von summiertem Laufzeitfehler und Positionsfehler mit (rot) und ohne (blau) Ausreißerbehandlung

Abbildung 14 bietet den direkten Vergleich aller Datenpunkte mit jeweils summiertem Gesamtlaufzeitfehler für alle fünf Empfänger. Die blaue Kurve entsteht aus den Datenpunkten ohne Ausreißerbehandlung, die rote Kurve aus denen mit. Ohne Laufzeitfehler, das heißt in der perfekten Welt, hat das unbereinigte approximative Verfahren einen erwarteten Positionsfehler von 25,8 cm ($\approx \frac{1}{4}$m). Die Ausreißereliminierung schneidet hier mit einem Fehler von 74,0 cm ($\approx \frac{3}{4}$m) etwa dreimal so schlecht ab. Die Erklärung liegt darin, dass die Ausreißerbehandlung darauf basiert, mehrere Lösungen zu mitteln, die aus den Zeitstempeln genau dreier Empfänger entstehen und somit stets unterbestimmt sind (siehe Abschnitt 4.3). Den Beweis für die Nützlichkeit der Optimierung geben die rechten Enden der Kurven. Hier zeigt sich, dass mit wachsender Größe des Gesamtlaufzeitfehlers die Ausreißereliminierung das unbereinigte Verfahren übertrifft. Ab einem

Laufzeitfehler von etwa 1,5 ms ist demnach das bereinigte Verfahren zu bevorzugen. Mit dem auf der NAO-Plattform erwartbaren Fehler von 1,16 ms pro Empfänger (siehe Gleichung (7)) ist die Ausreißereliminierung somit nicht nur sinnvoll sondern zwingend notwendig.



(a) ohne Ausreißerbehandlung (b) mit Ausreißerbehandlung

Abbildung 15: Einfluss des Verfeinerungsfaktors ϱ

Als nächstes wird der Einfluss des in der Simulation gewürfelten Verfeinerungsfaktors ϱ erörtert. Abbildung 15 zeigt zwei Diagramme analogen Prinzips sowie erneut mit (rechts) und ohne Bereinigung (links). Die jeweils vier grauen Kurven beschreiben die Datenpunkte vier verschiedener ϱ-Klassen. Diese sind äquidistant mit einer Länge von jeweils 0,2 auf das Intervall $\varrho \in [0{,}1; 0{,}9]$ aufgeteilt. Die hellste der vier Kurven beschreibt das erste Intervall $[0{,}1; 0{,}3]$, die dunkelste das letzte ($[0{,}7; 0{,}9]$). Es gilt, je „blasser“ die Kurve, desto kleiner der Faktor ϱ. In Abbildung 15b ist eine klare Tendenz zu erkennen: Je kleiner der Faktor ϱ gewählt ist, desto stärker steigt der erwartbare Positionsfehler mit zunehmendem Laufzeitfehler. Die Annahme aus Abschnitt 4.2, dass die Genauigkeit bei einer Verkleinerung von ϱ sinkt, ist damit bestätigt. Für die Kurven in Abbildung 15a, die die Ausreißer nicht behandeln, ist diese Abhängigkeit nicht eindeutig zu erkennen, wenngleich nicht auszuschließen.

Ein weiterer wichtiger Zusammenhang ist der zwischen Entfernung der Emitterposition $\vec{E}_{sim}$ zum Mittelpunkt des Feldes und der Güte der Lösung. Während der Durchführung der *Directional Whistle Challenge* (siehe Abschnitt 5.1) fiel eindeutig auf, dass mit steigendem Abstand der Pfiffposition zu den Empfängern die Lösungen immer ungenauer wurden. Abbildung 16 stellt den Zusammenhang zwischen der euklidischen Entfernung von $\vec{E}_{sim}$ zum Feldmittelpunkt und dem Positionsfehler in einer perfekten Simulation dar. Es fällt auf, dass die Genauigkeit des unbereinigten Verfahrens (blaue Kurve)

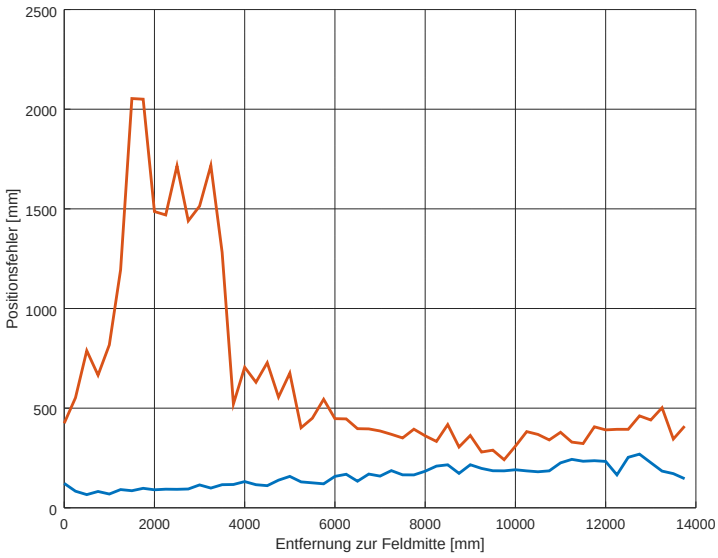


Abbildung 16: Vergleich des Zusammenhangs zwischen Positionsfehler und Entfernung des gesuchten Emitters zur Feldmitte mit (rot) und ohne (blau) Ausreißerbehandlung

mit steigender Entfernung trotz fehlender Messungenauigkeiten tatsächlich sinkt. Der erwartete Positionsfehler übersteigt dabei nicht einmal die 25 cm und bleibt damit in einem akzeptablen Rahmen. Die rote Kurve offenbart jedoch ein konzeptionelles Problem. Bei einer Entfernung zwischen 1,5 m–3,5 m vom Feldmittelpunkt versagt der vorgestellte Ansatz der Ausreißereliminierung. Der erwartete Fehler liegt hier bei bis zu 2 m. Bei näherer Untersuchung sind die Problempositionen dabei gehäuft in der linken Feldhälfte zu verorten. Aufgrund der Unterbestimmtheit der Einzellösungen kommt es vor allem zwischen den beiden Robotern in Tornähe zu sehr ähnlichen Symmetriefehlern, die dann durch Mittelung allein nicht mehr auszugleichen sind. Die Einschätzung, dass die Wahl des arithmetischen Mittels zur Findung der Gesamtlösung problematisch ist (siehe Abschnitt 4.3, letzter Absatz), ist demnach korrekt und stellt einen wichtigen Verbesserungsansatz dar.

6. Zusammenfassung

Die vorliegende Arbeit hat gezeigt, dass die akustische Ortung von Schallquellen durch Multilateration mithilfe der NAO-Roboter als Plattform grundsätzlich möglich ist. Die Einbettung einer Implementierung des Verfahrens in ein Modulframework wie das der *NaoDevils Dortmund* [Hofmann u. a., 2018] ist unkompliziert und sinnvoll (siehe Ab-

schnitt 3). Auch die Synchronisation der internen Uhren stellte sich als unproblematisch heraus.

Größere Schwierigkeiten bereitete die Verarbeitung des Audiostreams. Die Latenz der verbauten Audiohardware konnte selbst unter Laborbedingungen mithilfe der Soundarchitektur ALSA nicht gänzlich zufriedenstellend reduziert werden. Hinzu kommt die Ungenauigkeit bei der Bestimmung des exakten Zeitpunkts des Pfiffs. Das Modul WhistleDetector arbeitet auf Grundlage einer Frequenzanalyse um Störgeräusche hinreichend zu eliminieren. Aufgrund des anschwellenden Charakters des Pfeifsignals entsteht durch die Fenstertechnik eine zusätzliche Ungenauigkeit von maximal 1,16 ms.

Die Analyse der *Directional Whistle Challenge* in Abschnitt 5.1 offenbarte im Vergleich mit den anderen teilnehmenden Teams eine gute Performance des vorgestellten approximativen Verfahrens. Eine Auswertung der Resultate, die auf dem Punkteverfahren aus Abschnitt 1 basieren, zeigt einen klaren Vorteil des Laterationsverfahrens bei entfernteren Postionen. Die anderen Teams wendeten allesamt ein Angulations-basiertes Verfahren an. Das Gesamtresultat des Wettbewerbs war eher ernüchternd. Den besten durchschnittlichen Distanzfehler über alle Versuche hatte *TJArk* mit 6,06 m. Das Hauptziel, die Einschätzung der Feldzugehörigkeit, wurde mit durchschnittlich 4 von 8 Feldpunkten und einmal 5 bei den *HULKs* nicht erreicht.

Die simulierte Evaluation in Abschnitt 5.2 zeigt einen klaren Zusammenhang zwischen den schlechten Resultaten und den auftretenden Messungenauigkeiten. So stellt die vorgestellte Ausreißereliminierung eine nachweisbare Optimierung für das approximative Verfahren dar, macht aber für Positionen zwischen den Robotern in der derzeitigen Implementierung einen systematischen Fehler. Das Verfahren beruht auf der Annahme, dass generierte Zeitstempel unbrauchbar sein können und versucht diese herauszufiltern. Die daraus resultierende Unterbestimmtheit der Zwischenlösungen erzeugt symmetrische Fehler auf den Hyperbelkurven.

Alles in allem zeigt die vorgestellte Implementierung Verbesserungspotential. Die Hauptproblematiken beruhen auf den Hardwarebeschränkungen. Sollte die Audiohardware der NAO-Plattform in Zukunft eine Aufrüstung erfahren, ist das Laterationsverfahren in jedem Fall ein sinnvoller Lösungsansatz für das akustische Ortungsproblem. Beim jetzigen Stand ist mehr Forschung auf Softwareseite notwendig, um die nötige Audiosynchronisation weiter zu stabilisieren. Ansätze wie die Verknüpfung der Lateration mit anderen Verfahren werden im abschließenden Abschnitt 7 gegeben.

7. Ausblick

Die Evaluation der Arbeit an der *Directional Whistle Challenge* legt einige direkte Verbesserungsansätze nahe. Die Weiterentwicklung der Ausreißereliminierung aus Abschnitt 4.3 wurde bereits erwähnt. Die Ergebnisse der simulierten Evaluation aus Abschnitt 5.2 zeigen, dass die Wahl des arithmetischen Mittels zur Stabilisierung unzureichend ist. Hier könnte ein Clustering-Algorithmus bessere Resultate erzielen, insbesondere wenn die unterbestimmten Zwischenlösungen als mehrdeutig betrachtet würden. So könnten bei Unsicherheit der Symmetrie mehrere Zwischenlösungen pro Roboter-Tripel generiert und auf Plausibilität im Kontext einer möglichen Gesamtlösung verglichen werden.

Die genutzte Pfifferkennung auf Grundlage einer Frequenzanalyse, die für diese Arbeit verwendet und leicht verbessert wurde, könnte durch weitere Forschung optimiert oder gar substituiert werden. Die Konstruktion eines Neuronalen Netzes zur Lösung des Problems scheint vielversprechend, stellt allerdings ein eigenes Forschungsthema dar. Durch die Modulstruktur wäre das Einpflegen eines neuen Ansatzes in das vorliegende Verfahren vergleichsweise einfach (siehe Abschnitt 3).

Die Audiohardware müsste grundlegend ausgetestet werden, zum Beispiel durch die Installation verschiedener Treiber. Für die Kamerahardware ist derartiges bereits durchgeführt worden und mündete in der äußert gewinnbringenden Implementierung eines eigenen Kameratreibers[19]. Der Umfang einer solchen Arbeit sollte jedoch nicht unterschätzt werden.

Beim Testen des Latenzausgleichs von ALSA in Abschnitt 3.2 mithilfe eines Knalltests kam ein weiterer Ansatz zur Synchronisation auf. Positioniert man die Roboter-Mikrofone in kleinem Abstand zueinander, kann der Knall als Referenzzeitpunkt dienen. Ein ähnliches Verfahren nutzen Filmemacher beim Schlagen der berühmten Filmklappe zur Bild-Ton-Synchronisation. Problem bei den NAOs: Der Audiostream kann bei Überlastung abreißen. In diesem Fall ist die Synchronisation verloren.

Weiterhin scheint die Kombination des Laterationsansatzes mit anderen Verfahren sehr vielversprechend. In Abschnitt 2 (insbesondere Abbildung 3b) wurde bereits das Verfahren der Angulation vorgestellt. Die anderen Teams der *Directional Whistle Challenge* zeigten, dass der Ansatz Vorteile haben kann. *Berlin United* hat den entscheidenden Anteil seiner Gesamtpunktzahl durch Richtungsmaßpunkte erreicht und somit

[19] Informationen zu den Treibern im NAO-Kernel und weitere Details zur Arbeit am Kameratreiber finden sich bei Hofmann u. a. [2016, Abschn. 3.6.3] sowie unter: `https://github.com/NaoDevils/NaoKernel` (letzter Abruf: 30.1.2020)

die Machbarkeit der akustischen Angulation auf der NAO-Plattform bewiesen. Mithilfe der vier Mikrofone und einer Kreuzkorrelation zwischen den Signalen kann die Richtung bestimmt werden. Fünf mögliche Positions- und Richtungsinformationen ermöglichen dabei ebenso die Fehleroptimierung durch Überbestimmtheit. Die Notwendigkeit einer Roboter-übergreifenden Uhrensynchronisation entfällt bei diesem Verfahren. Die größte Herausforderung stellt hier die Robustheit gegen Störgeräusche und somit die Isolierung des Pfeifsignals im Audiostream dar. Hinzu kommt, dass Angulationsverfahren bei weit entfernten Positionen schlechter arbeiten. Der beste Ansatz ist deshalb die simultane Anwendung beider Verfahren zur Bestimmung der Gesamtlösung. Zudem sei darauf hingewiesen, dass in einer realen Spielsituation die globalen Roboterpositionen nicht sicher sind. Das heißt, es muss eine Eigenlokalisation (siehe Abschnitt 2, erster Absatz) der einzelnen Roboter stattfinden. Dies birgt gravierende Ungenauigkeiten.

Sollten aufgrund der Beschränkungen der Audiohardware auf der NAO-Plattform keine hinreichend exakten und robusten Resultate erzielt werden, kann schließlich auch die Hinzunahme weiterer Sensoren wie der Kamera angestrebt werden. Ein Beispiel dafür ist die visuelle Erkennung des Schiedsrichters im Kamerabild oder gar der Pfeife in seinem Mund. Die Schiedsrichter in Spielen der SPL nehmen zum Anpfiff so gut wie ausnahmslos eine der beiden Positionen an den Enden der Mittellinie ein. Dem Spielprotokoll folgend werden die Roboter über den nahenden Anpfiff zudem per WLAN informiert [vgl. TC, 2019a, Abschn. 3.6]. Wird nun mithilfe der akustischen Sensoren, also der Mikrofone, ein Pfiff erkannt, aber kein Schiedsrichter mit Pfeife auf einer der beiden typischen Positionen *gesehen*, ist davon auszugehen, dass der Pfiff auf einem Nachbarfeld gefallen ist. Momentan könnte solch eine praxisorientierte Herangehensweise für ein Spiel im Hauptwettbewerb der SPL mutmaßlich die vielversprechendste sein.

A. Quellcode des Moduls WhistleDirectionDetector

Grundfunktionen des Frameworks [Hofmann u. a., 2018; Röfer u. a., 2015] sind im Code durch die Farbe Violett gekennzeichnet.

A.1. C++-Header

```
1/** @file WhistleDirectionDetector.h */
2
3#pragma once
4
5#include "Representations/Configuration/FieldDimensions.h"
6#include "Representations/Infrastructure/FrameInfo.h"
7#include "Representations/Infrastructure/TeamInfo.h"
8#include "Representations/Infrastructure/TeammateData.h"
9#include "Representations/Modeling/RobotPose.h"
10#include "Representations/Infrastructure/RobotInfo.h"
11#include "Modules/Modeling/WorldModelGenerator/FixedPositioner.h"
12#include "Tools/Module/Module.h"
13#include "Representations/Modeling/WhistleDortmund.h"
14#include "Representations/Modeling/WhistleDirection.h"
15
16STREAMABLE(WhistleDirectionDetectorParams,
17{,
18  // environment constants
19  (float)(470.f) MIC_HEIGHT,
20  (float)(1600.f) MIN_HUMAN_SIZE,
21  (float)(1900.f) MAX_HUMAN_SIZE,
22  (Vector2f)(Vector2f(5200.f, 3700.f)) EXTENDED_OWN_FIELD_SIZE,
23
24  // approximation parameters
25  (bool)(true) USE_TRIPLES,
26  (float)(1000.f) INITIAL_GRID_SIZE, // size of the grid at first
        search step
27  (Vector3f)(Vector3f(15000.f, 15000.f, MAX_HUMAN_SIZE -
        MIN_HUMAN_SIZE)) INITIAL_SEARCH_SPACE, // the search space
        for approx. solution, cuboid around bottom center (0,0,
        MIC_HEIGHT)
28  (Vector3f)(Vector3f(0.f, 0.f, 1750.f)) INITIAL_CENTER,
29  (float)(0.9f) REFINEMENT_FACTOR, // search space and grid size are
         multiplied by this parameter, lower=faster
30  (float)(10.f) FINAL_PRECISION,
31
32  // debug setting
33  (bool)(false) CALC_EVERY_FRAME,
34  (bool)(false) ORACLE_MODE, // ground truth mode
35  (Vector3f)(Vector3f(4500,-3000,1720)) ORACLED_EMITTER,
36  (int)(4) NUM_ORACLED_TEAMMATES,
37  (float)(1000000.f) MAX_ORACLE_TIME_ERROR,
38  (float)(1000000000.f) MAX_ERROR_TO_DRAW,
39  (float)((MIN_HUMAN_SIZE + MAX_HUMAN_SIZE)/2) Z_ERROR_DRAW_HEIGHT,
40  (float)(FINAL_PRECISION*10) Z_ERROR_DRAW_HULL,
41});
42
43MODULE(WhistleDirectionDetector,
44{,
45  REQUIRES(FieldDimensions),
46  REQUIRES(FrameInfo),
47  REQUIRES(OwnTeamInfo),
48  REQUIRES(TeammateData),
49  REQUIRES(RobotPose),
50  REQUIRES(RobotInfo),
51  REQUIRES(PoseByNumber),
52  REQUIRES(WhistleDortmund),
53  PROVIDES(WhistleDirection),
54  LOADS_PARAMETERS(
55  {,
56    (WhistleDirectionDetectorParams) params,
57  }),
58});
59
60class WhistleDirectionDetector : public WhistleDirectionDetectorBase
61{
62public:
63  WhistleDirectionDetector();
64  unsigned startupTime;
65private:
66
67  // sound velocity (speed of sound) in mm/ms at 20 degree celsius
        in dry air
68  const float SOUND_VELOCITY = 331.300f + 0.606f * 20.f;
69
70  // global fields
71  std::vector<float> distanceDiffs = std::vector<float>(0);
72  std::vector<Vector3f> coordinates = std::vector<Vector3f>(0);
73  std::vector<uint64_t> timestamps = std::vector<uint64_t>(0);
74  Vector3f oracledEmitterOld = Vector3f::Zero();
75  uint64_t dummyTimestamp;
76
77  // functions
78  void update(WhistleDirection& whistleDirection);
79  /** perform coordinate approximation based on grid estimation */
80  Vector3f calcCoordByApprox(const std::vector<size_t> &indexes);
81  /** validate a coordinate candidate by determine average error */
82  float validateCoordinates(Vector3f candidate, const std::vector<
        Vector3f> &localCoordinates, const std::vector<float> &
        localDistanceDiffs);
83  /** reads coordinates and calculates TDOAs from team robots,
84   * if data is insufficient, returns false */
85  bool prepareData(WhistleDirection& whistleDirection);
86  /** calculates distance based on TDOA in millimeter */
87  float calcDistanceDiff(uint64_t t1, uint64_t t2);
88  /** calculates time diff of two timestamps in ns */
89  int64_t timeDiff(uint64_t t1, uint64_t t2);
90
91  // debug drawings
92  void drawErrorsOnField(Vector3f candidate, float error, float
        gridSize)
93  {
94    if (error > params.MAX_ERROR_TO_DRAW)
95      return;
96    if (candidate.z() < params.Z_ERROR_DRAW_HEIGHT - params.
          Z_ERROR_DRAW_HULL
97        || candidate.z() > params.Z_ERROR_DRAW_HEIGHT + params.
              Z_ERROR_DRAW_HULL)
98      return;
99    float heat = 255.f - 255.f*error/params.MAX_ERROR_TO_DRAW;
100   unsigned char alpha = static_cast<unsigned char>(heat/gridSize);
101   unsigned char color = static_cast<unsigned char>(heat);
102   RECTANGLE2("module:WhistleDirectionDetector:approxOnField",
103     Vector2i(candidate.x(), candidate.y()),
104     gridSize,
105     gridSize,
```

```
    0,
    10,
    Drawings::noPen,
    ColorRGBA(0,0,0,0),
    Drawings::solidBrush,
    ColorRGBA(color, 255 - color, 0, alpha));
}

void drawBestCandidate(Vector3f candidate)
{
  CROSS("module:WhistleDirectionDetector:approxOnField",
        candidate.x(),
        candidate.y(),
        70,
        50,
        Drawings::solidPen,
        ColorRGBA(0,0,0,255));
}

void drawTripleCandidate(Vector3f candidate)
{
  CROSS("module:WhistleDirectionDetector:approxOnField",
        candidate.x(),
        candidate.y(),
        40,
        20,
        Drawings::solidPen,
        ColorRGBA(0,0,0,80));
}

void drawReceiver(Vector3f rec, bool isMe)
{
  CIRCLE("module:WhistleDirectionDetector:approxOnField",
         rec.x(), rec.y(), isMe?120:80, isMe?80:50,
         Drawings::solidPen, ColorRGBA::black,
         Drawings::noBrush, ColorRGBA(0,0,0,0));
}
};
```

A.2. C++-Programmcode

```
/** @file WhistleDirectionDetector.cpp */

#include "WhistleDirectionDetector.h"
#include "Tools/Settings.h"
#include <cinttypes>
#include "Tools/Math/Probabilistics.h"
#include "Tools/Math/GaussianDistribution3D.h"

MAKE_MODULE(WhistleDirectionDetector, cognitionInfrastructure)

WhistleDirectionDetector::WhistleDirectionDetector()
{
 startupTime = SystemCall::getCurrentSystemTime();
 dummyTimestamp = SystemCall::getSystemTimeNanos();
}

void WhistleDirectionDetector::update(WhistleDirection&
      whistleDirection)
{
 DECLARE_DEBUG_DRAWING("module:WhistleDirectionDetector:
       approxOnField", "drawingOnField");

 // delay first calculation from system startup
 if (SystemCall::getCurrentSystemTime() - startupTime < 5000)
  return;

 if (params.ORACLE_MODE)
  whistleDirection.oracledCoord = params.ORACLED_EMITTER;

 whistleDirection.sendThisFrame = false;
 if (prepareData(whistleDirection) || params.CALC_EVERY_FRAME)
 {
  whistleDirection.detectionTime = theWhistleDortmund.
        trueDetectionTime;

  if (params.USE_TRIPLES)
  {
    // try different triples
    std::vector<size_t> triple({0,1,2});
    std::vector<Vector3f> tripleSolutions;
    for (size_t i=0; i<coordinates.size(); i++)
      for (size_t j=i+1; j<coordinates.size(); j++)
        for (size_t k=j+1; k<coordinates.size(); k++)
        {
          triple[0] = i;
          triple[1] = j;
          triple[2] = k;
          tripleSolutions.push_back(calcCoordByApprox(triple));
        }
    Vector3f sumVector = Vector3f::Zero();
    for (size_t i=0; i<tripleSolutions.size(); i++)
    {
      sumVector += tripleSolutions[i];
      drawTripleCandidate(tripleSolutions[i]);
    }
    sumVector /= tripleSolutions.size();

    whistleDirection.whistleCoordinates = sumVector;
  }
  else
  {
    whistleDirection.whistleCoordinates = calcCoordByApprox(std::
          vector<size_t>(0));
  }
  drawBestCandidate(whistleDirection.whistleCoordinates);

  whistleDirection.ownField =
      ((std::abs(whistleDirection.whistleCoordinates.x()) <= params
            .EXTENDED_OWN_FIELD_SIZE.x())
      && (std::abs(whistleDirection.whistleCoordinates.y()) <=
            params.EXTENDED_OWN_FIELD_SIZE.y()));

  whistleDirection.lastCalcTime = SystemCall::getCurrentSystemTime
        ();
  if (theRobotInfo.number == 1)
    whistleDirection.sendThisFrame = true;

  for (size_t i=0; i<coordinates.size(); i++)
    drawReceiver(coordinates[i], i==0);
 }
 whistleDirection.distanceDiffs = distanceDiffs;
 whistleDirection.coordinates = coordinates;
 whistleDirection.timestamps = timestamps;
}

Vector3f WhistleDirectionDetector::calcCoordByApprox(const std::
      vector<size_t> &indexes)
{
 if (params.ORACLE_MODE && theRobotInfo.number != 1)
  return Vector3f::Zero();
```

```

// wrong indexes do not crash the calculation
for (size_t i=0; i<indexes.size(); i++) {
    if (indexes[i] >= coordinates.size())
    {
      OUTPUT_WARNING("INPUT␣ERROR␣(WhistleDirectionDetector::
            calcCoordByApprox):␣index␣at␣" << i << ":" << indexes[i
            ] << "␣too␣large.␣return␣default␣value");
      return params.INITIAL_CENTER;
    }
}

std::vector<float> localDistanceDiffs = std::vector<float>(0);
std::vector<Vector3f> localCoordinates = std::vector<Vector3f>(0);
// order of diff vectors: (01,02,03,04,12,13,14,23,24,34) but
      robots from *numbers only
if (indexes.empty())
{
  localDistanceDiffs = distanceDiffs;
  localCoordinates = coordinates;
}
for (size_t i=0; i<timestamps.size(); i++)
{
  if (std::find(indexes.begin(), indexes.end(), i) != indexes.end()
        )
  {
    localCoordinates.push_back(coordinates[i]);
    for (size_t j=i+1; j<timestamps.size(); j++)
    {
      if (std::find(indexes.begin(), indexes.end(), j) != indexes.
            end())
      {
        float distance = calcDistanceDiff(timestamps[i], timestamps
              [j]);
        localDistanceDiffs.push_back(distance);
      }
    }
  }
}

float gridSize = params.INITIAL_GRID_SIZE;
Vector3f searchSpace = Vector3f(params.INITIAL_SEARCH_SPACE);
Vector3f center = Vector3f(params.INITIAL_CENTER);

Vector3f bestCandidate = Vector3f(params.INITIAL_CENTER);
float bestError = validateCoordinates(bestCandidate,
      localCoordinates, localDistanceDiffs);

while (gridSize >= params.FINAL_PRECISION)
{
  for (float x=center.x()-searchSpace.x(); x<=center.x()+
        searchSpace.x(); x+=gridSize)
  {
    for (float y=center.y()-searchSpace.y(); y<=center.y()+
          searchSpace.y(); y+=gridSize)
    {
      for (float z=std::max(params.MIN_HUMAN_SIZE, center.z()-
            searchSpace.z()); z<=std::min(params.MAX_HUMAN_SIZE,
            center.z()+searchSpace.z()); z+=gridSize)
      {
        Vector3f candidate = Vector3f(x,y,z);
        float error = validateCoordinates(candidate,
              localCoordinates, localDistanceDiffs);
        drawErrorsOnField(candidate, error, gridSize);
        if (error < bestError)
        {
          bestCandidate = candidate;
          bestError = error;
        }
      }
    }
  }

  center = Vector3f(bestCandidate);
  searchSpace = Vector3f(searchSpace.x()*params.REFINEMENT_FACTOR,
        searchSpace.y()*params.REFINEMENT_FACTOR, searchSpace.z()*
        params.REFINEMENT_FACTOR);
  // clip to initial search space
  if (center.x() + searchSpace.x() > params.INITIAL_SEARCH_SPACE.x
        ())
    center.x() = params.INITIAL_SEARCH_SPACE.x() - searchSpace.x();
  if (center.x() - searchSpace.x() < -params.INITIAL_SEARCH_SPACE.x
        ())
    center.x() = -params.INITIAL_SEARCH_SPACE.x() + searchSpace.x()
          ;
  if (center.x() + searchSpace.y() > params.INITIAL_SEARCH_SPACE.y
        ())
    center.y() = params.INITIAL_SEARCH_SPACE.y() - searchSpace.y();
  if (center.y() + searchSpace.y() > params.INITIAL_SEARCH_SPACE.y
        ())
    center.y() = params.INITIAL_SEARCH_SPACE.y() - searchSpace.y();
  // set gridSize
  gridSize = gridSize*params.REFINEMENT_FACTOR;
}

return bestCandidate;
}

float WhistleDirectionDetector::validateCoordinates(Vector3f
      candidate, const std::vector<Vector3f> &localCoordinates,
      const std::vector<float> &localDistanceDiffs)
{
float error = 1.f;
// order of diff vectors: (01,02,03,04,12,13,14,23,24,34)
// accumulate error compared to hyperbel function defined by given
       candidate and two receivers (0 and i)
size_t distanceDiffsIndex = 0;
for (size_t i=0; i<localCoordinates.size(); i++)
  for (size_t j=i+1; j<localCoordinates.size(); j++)
  {
    float singleError = std::abs(
          localDistanceDiffs[distanceDiffsIndex] - ((
                localCoordinates[i]-candidate).norm() - (
                localCoordinates[j]-candidate).norm()));
    error += singleError;
    distanceDiffsIndex++;
}
ASSERT(error >= 1.f);
return error;
}

bool WhistleDirectionDetector::prepareData(WhistleDirection&
      whistleDirection)
{
distanceDiffs.clear();
coordinates.clear();
timestamps.clear();

// DEBUG DUMMY DATA
oracledEmitterOld = Vector3f(params.ORACLED_EMITTER);
if (params.ORACLE_MODE)
{
  Vector3f GT = params.ORACLED_EMITTER;

  if (params.CALC_EVERY_FRAME)
     dummyTimestamp = SystemCall::getSystemTimeNanos();
  timestamps.push_back(dummyTimestamp);
  coordinates.push_back(
       Vector3f(theRobotPose.translation.x(),theRobotPose.
             translation.y(), params.MIC_HEIGHT)); // my
             coordinates
  ASSERT(thePoseByNumber.byNumber.size() == 6);
  if (theTeammateData.teammates.size() < 4) // use virtual robots
  {
```

```
    for (int i = 0; i < params.NUM_ORACLED_TEAMMATES+1; i++)
    {
      if (i != theRobotInfo.number - 1)
      {
        Vector3f coordinate = Vector3f(
              thePoseByNumber.byNumber[i].translation.x(),
              thePoseByNumber.byNumber[i].translation.y(),
              params.MIC_HEIGHT);
        float oracleDistanceDiff = (coordinates[0] - GT).norm() - (
              coordinate - GT).norm();
        int64_t timeDiff = static_cast<int64_t>(oracleDistanceDiff
              / SOUND_VELOCITY * 1000000.f);
          int64_t timeError = static_cast<int64_t>(randomGauss() *
                params.MAX_ORACLE_TIME_ERROR - params.
                MAX_ORACLE_TIME_ERROR / 2.f);
          timeDiff += timeError;
        if (std::abs(timeDiff) < 60000000) // else more than 20m
        {
          timestamps.push_back(timestamps[0] - timeDiff);
          coordinates.push_back(coordinate);
        }
      }
    }
  }
  else // use simulated robots and local teamcomm simulation
  {
    for (auto& mate : theTeammateData.teammates)
    {
      if (mate.status >= Teammate::ACTIVE)
      {
        Vector3f coordinate = Vector3f(mate.pose.translation.x(),
              mate.pose.translation.y(), params.MIC_HEIGHT);
        float oracleDistanceDiff = (coordinates[0] - GT).norm() - (
              coordinate - GT).norm();
        int64_t timeDiff = static_cast<int64_t>(oracleDistanceDiff
              / SOUND_VELOCITY * 1000000.f);
        int64_t timeError = static_cast<int64_t>(randomGauss() *
              params.MAX_ORACLE_TIME_ERROR - params.
              MAX_ORACLE_TIME_ERROR / 2.f);
        timeDiff += timeError;
        if (std::abs(timeDiff) < 60000000) // else more than 20m
        {
          timestamps.push_back(timestamps[0] - timeDiff);
          coordinates.push_back(coordinate);
        }
      }
    }
  }
}

// Real Teammate data
else
{ // order of coordinates (0,1,2,3,4)
  // distance difference to whistle in meter (to myself is 0)
  timestamps.push_back(theWhistleDortmund.trueDetectionTime);
  coordinates.push_back(
    Vector3f(theRobotPose.translation.x(),theRobotPose.translation.
          y(), params.MIC_HEIGHT)); // my coordinates
  // add pairs (01,02,03,04)
  for (auto &mate : theTeammateData.teammates)
  {
    if (mate.status >= Teammate::ACTIVE
        && std::abs(timeDiff(timestamps[0], mate.whistle.
              trueDetectionTime)) < 60000000) // else more than 20m
    {
      timestamps.push_back(mate.whistle.trueDetectionTime);
      coordinates.push_back(
            Vector3f(mate.pose.translation.x(), mate.pose.
                  translation.y(), params.MIC_HEIGHT));
    }
  }
  ASSERT(timestamps.size() == coordinates.size());

}

// order of diff vectors: (01,02,03,04,12,13,14,23,24,34)
for (size_t i=0; i<timestamps.size(); i++)
  for (size_t j=i+1; j<timestamps.size(); j++)
  {
      float distance = calcDistanceDiff(timestamps[i], timestamps[j
            ]);
      distanceDiffs.push_back(distance);
  }

//check for enough teammate data
if (coordinates.size() < 5)
  return false;

// check if something has changed
if (coordinates.size() != whistleDirection.coordinates.size())
  return true;
for (size_t i=0; i<timestamps.size(); i++)
{
  if (timestamps[i] != whistleDirection.timestamps[i]
      || coordinates[i] != whistleDirection.coordinates[i])
    return true;
}
// new oracle calculation requested
if (oracledEmitterOld != params.ORACLED_EMITTER)
  return true;
// nothing changed
return false;
}

float WhistleDirectionDetector::calcDistanceDiff(uint64_t t1,
      uint64_t t2)
{
int64_t td = timeDiff(t1, t2);
return static_cast<float>(td)/1000000.f * SOUND_VELOCITY;
}

int64_t WhistleDirectionDetector::timeDiff(uint64_t t1, uint64_t t2)
{
// calculates time difference of two unsigned 64 bit integers
return ((t1 >= t2) ? static_cast<int64_t>(t1 - t2) : -static_cast<
      int64_t>(t2 - t1));
}
```

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[Aldebaran 2017] Aldebaran: *NAO Documentation*. Version 2.1. 43 rue du Colonel Pierre Avia, Paris, Frankreich: SoftBank Robotics Europe, 2017. `http://doc.aldebaran.com/2-1/home_nao.html`. – letzter Abruf: 30.1.2020

[Bucher u. Misra 2002] Bucher, Ralph ; Misra, D.: A Synthesizable VHDL Model of the Exact Solution for Three-dimensional Hyperbolic Positioning System. In: *VLSI Design* 15 (2002), Nr. 2, S. 507–520

[Butz 2015] Butz, Tilman: *Fourier Transformation for Pedestrians*. 2. Auflage. Basel, Schweiz : Springer International Publishing, 2015

[Curnow u. a. 2019] Curnow, Richard P. ; Lichvar, Miroslav u. a.: *chrony – Documentation*. TuxFamily.org, 2019. `https://chrony.tuxfamily.org/documentation.html`. – letzter Abruf: 30.1.2020

[Gauss 1865] Gauss, C.F.: *Theorie der Bewegung der Himmelskörper welche in Kegelschnitten die Sonne umlaufen*. Übersetzung des lateinischen Originals (1809) von C. Haase. Hannover : Carl Meyer, 1865

[Hofmann u. a. 2018] Hofmann, Matthias ; Schwarz, Ingmar ; Urbann, Oliver ; Larisch, Aaron: Nao Devils Dortmund – Team Report 2018 / TU Dortmund, Robotics Research Institute, Section Information Technology. Version: 2018. `https://github.com/NaoDevils/CodeRelease/blob/CodeRelease2018/TeamReport2018.pdf`. 2018. – Forschungsbericht. – letzter Abruf: 30.1.2020

[Hofmann u. a. 2016] Hofmann, Matthias ; Schwarz, Ingmar ; Urbann, Oliver ; Rensen, Fabian ; Larisch, Aaron ; Moos, Arne ; Hemmers, Janine: Nao Devils Dortmund – Team Report 2016 / TU Dortmund, Robotics Research Institute, Section Information Technology. Version: 2016. `https://github.com/NaoDevils/CodeRelease/blob/CodeRelease2016/TeamReportNaoDevils2016.pdf`. 2016. – Forschungsbericht. – letzter Abruf: 30.1.2020

[IETF 2019] IETF, Internet Engineering Task Force: *IETF Documents*. https://tools.ietf.org/html/, 2019. – letzter Abruf: 30.1.2020

[Jochmann u. a. 2012] Jochmann, Gregor ; Kerner, Sören ; Tasse, Stefan ; Urbann, Oliver: Efficient Multi-hypotheses Unscented Kalman Filtering for Robust Localization. In: Röfer, Thomas (Hrsg.) ; Mayer, N. Michael (Hrsg.) ; Savage, Jesus

(Hrsg.) ; Saranli, Uluc (Hrsg.): *RoboCup 2011: Robot Soccer World Cup XV* Bd. 7416. Berlin, Heidelberg : Springer, 2012 (Lecture Notes in Computer Science)

[Kysela u. a. 2019] Kysela, Jaroslav ; Bagnara, Abramo ; Iwai, Takashi ; van de Pol, Frank: *ALSA project - the C library reference.* ALSA Project, 2019. `https://www.alsa-project.org/alsa-doc/alsa-lib/`. – letzter Abruf: 30.1.2020

[Meister 2015] Meister, Andreas: *Numerik linearer Gleichungssysteme – Eine Einführung in moderne Verfahren.* 5. Auflage. Wiesbaden : Springer Spektrum, 2015

[Ostermann u. a. 2016] Ostermann, Fabian ; Mücke, Janina ; Plenne, Andrej: *PG 598: Self-Aware Soccer Playing Humanoid Robots – Status Report,* Chapter 2: *Whistle Sound Detection in an Audio Stream.* September 2016. – unveröffentlicht

[Röfer u. a. 2007] Röfer, Thomas ; Brose, Jörg ; Göhring, Daniel ; Jüngel, Matthias ; Laue, Tim ; Risler, Max: GermanTeam 2007. In: Visser, Ubbo (Hrsg.) ; Ribeiro, Fernando (Hrsg.) ; Ohashi, Takeshi (Hrsg.) ; Dellaert, Frank (Hrsg.): *RoboCup 2007: Robot Soccer World Cup XI Preproceedings.* Atlanta, GA, USA : RoboCup Federation, 2007

[Röfer u. a. 2013] Röfer, Thomas ; Laue, Tim ; Müller, Judith ; Bartsch, Michel ; Batram, Malte J. ; Böckmann, Arne ; Böschen, Martin ; Kroker, Martin ; Maass, Florian ; Münder, Thomas ; Steinbeck, Marcel ; Stolpmann, Anderas ; Taddiken, Simon ; Tsogias, Alexis ; Wenk, Felix: Team Report and Code Release 2013 / Deutsches Forschungszentrum für Künstliche Intelligenz / Universität Bremen. Version: 2013. `https://github.com/bhuman/BHumanCodeRelease/raw/coderelease2015/CodeRelease2015.pdf`. 2013. – Forschungsbericht. – letzter Abruf: 30.1.2020

[Röfer u. a. 2015] Röfer, Thomas ; Laue, Tim ; Richter-Klug, Jesse ; Maik-Schünemann ; Stiensmeier, Jonas ; Stolpmann, Andreas ; Stöwing, Alexander ; Thielke, Felix: Team Report and Code Release 2015 / Deutsches Forschungszentrum für Künstliche Intelligenz/Universität Bremen. Version: 2015. `https://github.com/bhuman/BHumanCodeRelease/raw/coderelease2013/CodeRelease2013.pdf`. 2015. – Forschungsbericht. – letzter Abruf: 30.1.2020

[TC 2019a] TC, RoboCup Technical Committee: *RoboCup Standard Platform League (NAO) Rule Book.* 2019 `http://spl.robocup.org/wp-content/uploads/downloads/Rules2019.pdf`. – letzter Abruf: 30.1.2020

[TC 2019b] TC, RoboCup Technical Committee: *RoboCup Standard Platform League (NAO) Technical Challenges*. 2019 `http://spl.robocup.org/wp-content/uploads/downloads/Challenges2019.pdf`. – letzter Abruf: 30.1.2020

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 3. Februar 2020

Fabian Ostermann