Strategische Positionierung von Robotern im RoboCup

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)

Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik

eingereicht von:	Steffen Kaden
geboren am:	30.07.1990
in:	Berlin Lichtenberg
Gutachter(innen):	Prof. Hans-Dieter Burkhard
	Prof. Verena V. Hafner
eingereicht am:	.....
verteidigt am:	.....

Zusammenfassung

Mit dem Fortschritt im Roboter-Fussball-Wettbewerb RoboCup Soccer wird die Interaktion der Teammitglieder miteinander für den Erfolg im Wettbewerb immer wichtiger. Teamwork wird zum Schlüssel des Erfolgs und ohne Teamwork ist das Ziel, den 2050 amtierenden FIFA Weltmeisters zu besiegen, nicht erreichbar.

Ein Teilproblem des Teamworks ist die Positionierung der Spieler auf dem Feld. Es sind keine komplexen Spielzüge möglich, ohne ein Positionierungskonzept zu haben, das besagt, wo und wann sich ein Spieler befinden sollte. Für einen Roboter müssen damit im wesentlichen zwei Fragen beantwortet werden. An welcher Position sollte er stehen und wie kommt er dahin? Um diese Fragen zu beantworten, wird ein neues Konzept für die Beschreibung der Situation auf dem Feld vorgestellt. Die Situation wird in Form einer Karte repräsentiert, die es dem Roboter ermöglicht, sich entsprechend einer festgelegten Positionierungsstrategie zu positionieren. Um die Karte zu erstellen, wird das Feld mit Hilfe eines Voronoidiagramms diskretisiert, wodurch eine flexible Unterteilung möglich wird. Das Voronoidiagramm definiert gleichzeitig einen Graphen. Jeder Knoten des Graphens wird mit einem Gewicht versehen, welches mit Hilfe eines Skalarfeldes bestimmt wird. Das Skalarfeld beschreibt dabei die verwendete Positionierungsstrategie. Mit Hilfe der A*-Suche kann nun ein Pfad auf dem Graphen zur Zielposition bestimmt werden. Die Zielposition ist dabei durch den Knoten mit dem kleinsten Gewicht gekennzeichnet. Im Gegensatz dazu sollen Knoten mit hohen Gewichten vermieden werden.

In einer Simulation wird das Verfahren auf seine Funktion, sowie der Einfluss der Heuristik der A*-Suche und der Einfluss des Graphens auf den ermittelten Pfad an Hand von Experimenten untersucht. Dazu werden zwei statische Szenarien mit jeweils zwei Spielern betrachtet.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Mit dem Fortschritt im RoboCup wird die Interaktion der Teammitglieder miteinander für den Erfolg im Wettbewerb immer wichtiger. Teamwork wird zum Schlüssel des Erfolgs und ohne Teamwork ist das Ziel, den 2050 amtierenden FIFA Weltmeisters zu besiegen, nicht erreichbar.

Ein Teilproblem des Teamworks ist die Positionierung der Spieler auf dem Feld. Das Problem der strategischen Positionierung beschäftigt sich mit dem Positionierungsverhalten der Spieler, die nicht im Ballbesitz sind. Brüggemann und Albrecht stellen in ihrem Fußballhandbuch [5] das gewünschte Positionierungsverhalten menschlicher Fußballspieler, in Abhängigkeit ihrer Rolle und der Situation auf dem Feld, vor. Dabei stellt sich heraus, dass die strategische Positionierung der Spieler auf dem Feld ein komplexes Problem ist. Der Spieler muss abschätzen können, wo sich seine Mitspieler und die gegnerischen Spieler befinden und wohin sie sich bewegen werden. Auf Basis der Positionsinformationen anderer Spieler, der abgesprochenen Teamstrategie, des gegnerischen Verhaltens und der genutzten Spielerformation, kann ein Spieler eine strategische Position ermitteln und einnehmen.

Im Kontext des RoboCups muss also die Frage beantworten werden, wohin der Roboter soll. Dafür muss aus den bekannten Informationen über die Welt eine strategische Position abgeleitet werden. Außerdem muss beantwortet werden, wie der Roboter die ermittelte Position erreicht.

Um diese Fragen zu beantworten, wird ein Voronoidiagramm mit einem Skalarfeld kombiniert. Das Voronoidiagramm wird genutzt, um das Spielfeld zu diskretisieren. Die Unterteilung des Spielfeldes mit einem Voronoidiagramm ist dahingehend sinnvoll, da die Regionen des Voronoidiagramms konvex sind. Außerdem kann das Feld mit Hilfe eines Voronoidiagramms unterschiedlich fein diskretisiert werden. Zu dem kommt noch hinzu, dass durch das Voronoidiagramm eine Nachbarschaftsbeziehung der Elemente der Diskretisierung definiert wird. Dadurch wird eine effiziente Suche ermöglicht. Das Skalarfeld wird genutzt, um ein Positionierungsstrategie zu beschreiben, also wohin der Roboter soll. Das Skalarfeld wurde als Methode gewählt,

da es schon oft erfolgreich genutzt wurde, um die Positionierungsstrategie zu kodieren, zum Beispiel in [27] und [29]. In diesen Beispielen wurden die Ableitung der Skalarfelder genutzt, auch Gradientenfeld oder Potentialfeld genannt. Diese beiden Methoden werden kombiniert, um mit einer geeigneten Inferenzmethode Richtungskommandos abzuleiten und den Spieler so zu der gewünschten, strategischen Position zu steuern.

Die Implementierung einer Referenzlösung ist Teil der Arbeit. Dabei wird auf dem NaoTH-Framework des RoboCup-Teams Nao Team Humboldt aufgebaut. In Hinblick auf die Anwendung im RoboCup ist es wichtig, dass das Verfahren stabil ist. Außerdem soll es berücksichtigen können, was konkret nicht sein soll, z.B. dass kein Spieler den eigenen Strafraum betritt. Um zukünftiges Hinzufügen von Spielern zu erleichtern, sollte das Verfahren skalierbar sein.

1.1 RoboCup

Seit 1997 werden jährlich die RoboCup Weltmeisterschaften ausgetragen. Das ursprüngliche Ziel ist es, den führenden FIFA Weltmeister 2050 zu besiegen. Im Laufe der Zeit wurde der RoboCup um weitere Ligen, wie Rescue, @Home und Junior ergänzt, die sich nicht mit dem Problem des Fußballspielens beschäftigen. Im RoboCup Soccer gibt es die Humanoid League, Small und Middle Size League, Simulation League und die Standard Platform League (SPL) [10]. Die Standard Platform League zeichnet sich dadurch aus, dass alle teilnehmenden Teams identische Roboter nutzen [18]. Damit unterscheiden sich die Roboter hauptsächlich in ihrer Programmierung. Der Fokus der Forschung liegt auf den Algorithmen. Seit 2008 wird in dieser Liga der Roboter Nao von Aldebaran genutzt. Zuvor wurde der Roboterhund Aibo von Sony verwendet, weshalb die Liga damals Four Legged League hieß.

1.2 Struktur der Arbeit

Die weitere Arbeit ist wie folgt aufgebaut: In Kapitel 2 werden bereits existierende Lösungen, die im Rahmen des RoboCups entwickelt wurden, für das Problem der strategischen Positionierung vorgestellt. Im anschließendem Kapitel werden die mathematischen Grundlagen erläutert, um das in Kapitel 4 vorgestellte Verfahren Voronoi Based Situation Map zu verstehen. In Kapitel 5 werden die Eigenschaften des Verfahrens untersucht. Das letzten Kapitel bietet eine Zusammenfassung und einen Ausblick.

Kapitel 2

Verwandte Arbeiten

Das Problem der strategischen Positionierung wurde schon früher im Rahmen des RoboCups untersucht und Lösungen entwickelt. Im Folgenden werden einige dieser Lösungen der unterschiedlichen Ligen vorgestellt. Dabei spiegelt sich der Charakter der einzelnen Ligen in den Lösungen wider. So sind zum Beispiel die Lösungen der Simulationsliga im Allgemeinen elaborierter, da diese essentiell für den Erfolg sind. Des weiteren besitzen die dort verwendeten Rechner mehr Rechenleistung und Speicher [1], sodass das Problem der Resourcenknappheit nicht so eine große Rolle spielt, wie beispielsweise in der Standard Platform League. Des Weiteren haben die Teams in den Simulationsligen nicht mit den Unzulänglichkeiten der Hardware zu kämpfen.

In vielen Arbeiten wurde das Rollenkonzept aus dem menschlichen Fußball übernommen. Oft werden die Rollen Defender, Striker und Supporter aufgeführt. Der Defender ist ein verteidigender Spieler. Der Striker ist der Spieler, der zum Ball geht und versucht diesen in Richtung Tor zu bewegen. Dem Striker wird der Supporter zur Seite gestellt. Der Supporter soll den Striker beim Angriff unterstützen beziehungsweise absichern. Einige Arbeiten übernehmen auch die Prinzipien von Taktik und Strategie.

Die hier vorgestellten Arbeiten lassen sich, unter Berücksichtigung der verwendeten Methoden, grob in fünf Kategorien unterteilen. Es werden Potentialfelder oder Diagramme genutzt, die Positionierung als Optimierungsproblem betrachtet, die Positionen analytisch bestimmt oder in Form von Plänen und Szenarien definiert. Dabei lassen sich manche Arbeiten durchaus mehreren Kategorien zuordnen.

2.1 Lösungen auf der Basis von Potentialfeldern

Potentialfelder werden genutzt, um die Situation auf dem Feld zu kodieren. Dabei werden Potentialfelder so mit einander kombiniert, dass der Roboter an der

gewünschten, für strategisch sinnvoll erachtete Position gelangt, indem er dem Potentialfeld folgt. Potentialfelder werden im Abschnitt 3.5 näher erläutert.

So nutzen Work, Chown, Hermans, Butterfield und McGranaghan in [29] Potentialfelder und übernehmen gleichzeitig die Konzepte der Strategie, Formation und Rollen, um die Spieler zu positionieren. Eine Strategie wird durch eine Menge von Formationen definiert und kann sich durch eine offensivere oder defensivere Spielweise auszeichnen. Die Formationen sind abhängig von der Ballposition auf dem Feld. Jede Formation definiert eine Menge von Rollen, die von den Spielern eingenommen werden können. Eine Rolle unterteilt sich in Sub-Rollen, die aktiv sind, falls der Ball in einem bestimmten Bereich des Feldes ist. Als Beispiel wird die Rolle des Defenders aufgeführt. Der Defender unterteilt sich in die Sub-Rollen Stopper, Sweeper und Defensive Midfield. Der Defender soll im Allgemeinen weder die Mittellinie übertreten, noch den Strafraum betreten, sowie immer zwischen Ball und Tor stehen. Er wird zum Stopper, falls der Ball in der Mitte des Feldes ist und positioniert sich ungefähr 1m vom Ball entfernt auf der Linie zwischen Ball und Tor. Sollte der Ball in der gegnerischen Hälfte sein, wird er zum Defensive Midfield und bewegt sich auf der Mittellinie entlang. Wenn der Ball sehr nah am eigenen Tor ist, wird der Defender zum Sweeper und steht an statischen Koordinaten vor dem Tor.

Das RoboCup Team B-Human [27] nutzt während der Spiele drei Formationen. Bei der offensiven Formation stehen zwei Supporter links und rechts vom Striker, um ihm beim Angriff zu unterstützen. Bei der normalen Formation gibt es einen Supporter und einen Verteidiger und bei der defensiven Formation zwei Verteidiger vor dem Tor. Der Supporter positioniert sich ungefähr $1,1$ m seitlich und $0,3$ m hinter dem Striker. So kann er den Ball von Zeit zu Zeit sehen. Die genaue Positionierung wird mit Hilfe eines Potentialfeldsystems umgesetzt. Am Ende wird für den Supporter eine Laufrichtung ausgegeben. Das Potentialfeldsystem besteht aus anziehenden und abstoßenden Potentialfeldern. Der Supporter soll den Striker nicht behinder, weshalb abstoßende Potentialfelder vom Striker, der Strecke zwischen Striker und Ball sowie der Strecke zwischen Ball und Tor definiert werden. Supporter wirken aufeinander abstoßend, damit sie einen möglichst großen Bereich abdecken. Anziehend wirkt die gewünschte Zielposition und eine Position hinter dem Striker. Der eigene Strafraum wirkt zusätzlich abstoßend, um zu verhindern, dass der Supporter den Strafraum betritt.

Der Verteidiger steht an einer spezifischen x-Koordinate und kann sich seitlich vor dem Tor hin und her bewegen. Falls kein Torwart im Spiel ist, steht er auf der Linie zwischen Tormitte und Ball, sonst steht er um ca. 50cm seitlich versetzt vor dem Tor. Die genaue Positionierung wird mit einem Potentialfeldsystem, ähnlich dem obigen, realisiert.

Auf die Ergebnisse B-Humans baut das RoboCup Team NTURoboPAL [28] auf und positioniert ihren Supporter wie folgt: Es gibt einen offensiven Supporter in der gegnerischen Hälfte und einen defensiven in der eigenen. Der defensive Supporter

steht dabei auf der Linie zwischen Ball und Tormittelpunkt, während der offensive Supporter sich in der Nähe der Linie zwischen Ball und gegnerischem Tor positioniert. Damit kann der offensive Supporter schneller, nach einem Schuss des Strikers, zum Striker werden, den Ball, das Tor und den Striker beobachten und behindert den Striker nicht. Der Striker benachrichtigt den Supporter nach einem Schuss, damit dieser den Ball aktiv verfolgt, um den Rollentausch zu erleichtern. Ein Problem der Potentialfelder ist es, dass der Roboter in ein lokales Minimum laufen kann. Dieses Problem wird in 3.5 näher beschrieben.

2.2 Darstellung als Optimierungsproblem

Eine weitere Möglichkeit ist es, das Positionierungsproblem als ein Optimierungsproblem zu betrachten. Dabei wird die optimale Spielerposition anhand von sich teils widersprechenden Kriterien bestimmt.

Diesen Ansatz nutzen Vadim Kyrylov und Serguei Razykov [16]. Dabei werden die möglichen Positionen der offensiven Spieler nach bestimmten Kriterien bewertet und anschließend das Pareto-Optimum gesucht. Um die mögliche Positionen zu ermitteln, wird das Feld mit Hilfe eines Rasters diskretisiert. Anschließend werden nur die Positionen betrachtet, die der Roboter innerhalb eines bestimmten Zeitraums τ erreichen kann. Dieser Zeitraum τ entspricht der Zeit, die der Ball auf dem Feld frei rollt. Die nächste Position für einen offensiven Spieler sollte möglichst freistehend sein, um einen Pass annehmen zu können. Außerdem sollte der Abstand zum nächsten Gegner maximal und die neue Position so nah wie möglich an der Position sein, die von der Formation vorgegeben wird. Durch eine freie Schussbahn auf das gegnerische Tor und einer Position nahe der Abseitslinie, wird die gegnerische Verteidigung unter Druck gesetzt.

Für defensive Spieler haben Vadim Kyrylov und Eddie Hou [15] folgende Lösung entwickelt: Es wird jedem Angreifer eine Bedrohung in Abhängigkeit von der Distanz zum Tor sowie zum Ball und dem Winkel zum Tor zugeordnet. Anschließend wird eine Verteilung der Verteidiger auf die Angreifer gesucht, die die verhinderte Bedrohung maximiert und die benötigte Zeit minimiert. Dafür wird der gefährlichste Angreifer als erstes mit dem Verteidiger gedeckt, welcher am schnellsten die Bedrohung reduzieren kann. Mit den übrigen Angreifern wird ebenso verfahren, bis keine Verteidiger mehr zur Verfügung stehen. Damit sich Unterschiede im Weltmodell nicht so stark auswirken, übernimmt genau ein Roboter die Verteilung der Verteidiger. Dafür bietet sich der Goali an, da er wahrscheinlich das beste Weltmodell, auf Grund seiner Position, hat.

Beide Ansätze haben das Problem, dass genaue Daten benötigt werden. So müssen die Positionen aller Spieler bekannt sein und das Verhalten des Balls und der

Spieler modelliert werden, um die Länge des Zeitraums τ genau genug bestimmen zu können.

2.3 Lösungen mit Hilfe von Diagrammen

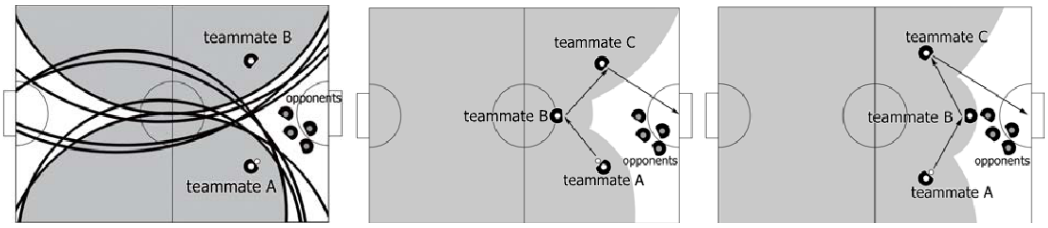
Andere Arbeiten nutzen zur Lösung des Positionierungsproblems Graphen und Diagramme. So stellten Hidehisa Akiyama und Itsuki Noda 2008 eine Lösung in [3] vor, die die Delaunay Triangulation und den Gouraud Shading Algorithmus verwendet. Gegeben wird dabei eine Menge von Ballpositionen und die jeweiligen gewünschten Spielerpositionen zu den Ballpositionen (Trainingsdaten). Auf der Menge der Ballpositionen wird vor dem Spielen die Delaunay Triangulation angewendet.

In einem Spiel wird nun die Position der Spieler wie folgt bestimmt: Stimmt die aktuelle Ballposition mit einer Position aus den Trainingsdaten überein, so versuchen die Roboter sich zu den Spielerpositionen zu bewegen, welche in den Trainingsdaten zu der Ballposition gehören. Anderenfalls liegt der Ball innerhalb eines Dreiecks und die Positionen der Spieler wird mit Hilfe des Gouraud Shading Algorithmus aus den gegebenen Positionen der Eckpunkte des Dreiecks (Trainingsdaten) interpoliert. Sollte sich während eines Spiels eine Positionierung aus der Trainingsmenge als ungünstig erweisen, so kann dieses Problem während des Spiels nicht behoben werden.

Hesam Addin Dashti et al. nutzen ein Voronoidiagramm, um die Spieler auf dem Feld zu positionieren [7]. Zu jedem Teammitglied wird dessen Voronoizelle bestimmt. Der Spieler "fühlt" sich vom geometrischen Zentrum seiner Voronoizelle angezogen und "spürt" weiter Anziehungs- oder Abstoßungskräfte von wichtigen Objekten in der Welt wie Tore, gegnerischen Spielern und dem Ball. Das Voronoidiagramm wird dazu genutzt um die Spieler auf dem Feld zu verteilen. In [6] wird das Verfahren dahingehend ergänzt, dass gegnerische Spieler berücksichtigt werden. Da gegnerische Spieler auch eine Voronoizelle erhalten, tendieren die eigenen Spieler dazu, sich im freien Raum zwischen gegenerischen Spielern zu positionieren. Außerdem wird eine Möglichkeit vorgestellt, mit Hilfe eines additiv gewichteten Voronoidiagramms die Ausdauer (Batterieladung) der Agenten zu berücksichtigen.

In der Small Size Liga werden dominante Regionendiagramme genutzt. Die dominante Region eines Spielers ist der Bereich auf dem Feld, der alle Punkte enthält, die der Spieler vor allen anderen Spielern erreichen kann. Die Diagramme können genutzt werden, um sich frei zu laufen [22] oder zu entscheiden, ob gepasst wird [21]. Falls der Pass durch eine Region führt, die nicht vom eigenen Team dominiert wird, dann kann in bestimmten Fällen ein Spieler so positioniert werden, dass die Lücke geschlossen wird (Abbildung 2.1). Das dominante Regionendiagramm benötigt genauso wie die Optimierungsansätze aus [15] und [16] genaue Informationen über die Mitspieler, damit das Diagramm erstellt werden kann. Für die Verwendung

in der SPL bedeutet das, dass andere Spieler und ihre Orientierung auf dem Feld erkannt werden müssen. Die Orientierung ist wichtig, da der Nao nicht in alle Richtungen gleich schnell laufen kann. Anschließend müsste noch abgeschätzt werden, wie schnell die einzelnen Spieler laufen.



(a) Ausgangsproblem (b) Lösungsmöglichkeit 1 (c) Lösungsmöglichkeit 2

Abbildung 2.1: Beispiel zur Verwendung der Dominanten Regionen, grau sind die dominanten Regionen des eigenen Teams. Das linke Bild stellt eine Situation dar, in der ein Pass von Teammitglied A zu Teammitglied B ungünstig ist, da der Ball durch die dominante Region des gegnerischen Teams müsste. Durch die Positionierung eines dritten Spielers kann das Problem gelöst werden (mittleres und rechtes Bild) [21].

2.4 Lösungen mit Hilfe von Plänen und Szenarien

Eine weitere Kategorie von Lösungen nutzt Pläne oder Szenarien. Sie haben gemein, dass versucht wird für jede Spielsituation eine zuvor festgelegte Reaktion des Teams zu haben. Die Spielsituation wird dabei relativ allgemein beschrieben.

Colin McMillen und Manuela Veloso stellen in [19] ein Verfahren vor, dass auf Teampläne (Play) aufbaut. Diese stellen eine Menge von Rollen bereit. Eine Rolle besteht aus einer Region, für die der Roboter verantwortlich ist, einem Verhalten, falls der Ball in der Region, beziehungsweise nicht in der Region ist und einem Verhalten, falls der Roboter nicht weiß wo der Ball ist.

Erfüllt die Spielsituation die Vorrausetzungen eines Plays, so kann es angewendet werden. Die Voraussetzungen basieren auf Eigenschaften der Umwelt, wie verbleibende Zeit bis zum Spielende, Spieleranzahl und Spielstand.

Jedem Play wird ein Gewicht zugeordnet. Wenn mehrere Plays anwendbar sind, dann wird stets dasjenige ausgewählt, welches das größte Gewicht besitzt. Das zu spielende Play wird vom teamführenden Roboter ermittelt, an die Mitspieler kommuniziert und kontinuierlich auf dessen Anwendbarkeit überprüft. Ein neues Play wird aktiviert, falls das aktuelle Play nicht mehr anwendbar ist oder ein Play mit höherem Gewicht genutzt werden kann. Sollte der teamführende Roboter ausfal-

len oder Probleme mit der Kommunikation der Roboter auftreten, so gibt es ein Default-Play, auf welches in solchen Situtationen zurückgegriffen wird.

Eine weitere auf Szenarien basierende Lösung ist Scenario-Based Teamworking [25]. Bei dieser Lösung handelt es sich um mehr als nur eine Lösung des Positionierungsproblems. Wie der Name vermuten lässt, ist es eine Lösung für das Problem des Teamworkings im Fußball. Da die Positionierung aber eine Grundlage für Teamwork ist, wird diese Lösung auch kurz vorgestellt, außerdem gibt das Szenario die Position vor.

Ein Szenario beschreibt mit Hilfe von Triggern allgemein den Zustand der Welt und der Agenten, sowie die Aktionen, welche ein Agent ausführen soll. Jedes Szenario besitzt ein Ziel, das es zu erreichen gilt, zum Beispiel soll ein Tor geschossen werden oder das Team in Ballbesitz kommen. Des Weiteren besitzen Szenarien Kosten für ihre Ausführung und einen Gewinnwert, falls sie erfolgreich ausgeführt werden. Da ein Szenario nur unter bestimmten Bedingungen sinnvoll ist, kann es abgebrochen werden, falls die Bedingungen nicht mehr erfüllt sind. Außerdem erzeugt ein Szenario Seiteneffekte, d.h. es erhöht das Spieltempo, nimmt Geschwindigkeit aus dem Spiel oder lässt die Spieler sich mehr auf dem Feld verteilen. Ein Szenario besteht aus aufeinanderfolgenden "Sub-Plänen" für jeden Agenten.

Der Erfolg eines Szenarios steht in Abhängigkeit zum vorherigen Szenario. Um ungewollte Kombinationen von Szenarien zu verhindern, wird ein Graph aufgebaut, dessen Knoten einzelne Szenarien sind. Die Kanten sind gewichtet und beschreiben, mit welcher Wahrscheinlichkeit ein Szenario nach einem anderem ausgeführt wird.

Um den Graphen zu erhalten, kann wie folgt vorgegangen werden: Am Anfang wird ein Szenario mit passenden Triggern gewählt und ausgeführt. Nach der Ausführung wird das nächste Szenario entsprechend der Trigger ausgewählt, wobei diese Szenarien mit einer Kante verbunden werden. Nun führt man das ganze solange durch, bis man eine gewünschte Konnektivität der Knoten erreicht hat. Als nächstes wird ein Pfad ausgewählt, der mit einem Clearing-Szenario beginnt und mit einem Scoring-Szenario endet. Diese Sequenz wird ohne Evaluation durchgeführt. Falls ein Szenario erfolgreich war, dann wird das Gewicht der zum Szenario führenden Kante erhöht, sonst wird es verringert. Wenn der ganze Pfad erfolgreich war, wird die Wahrscheinlichkeit des ganzen Pfades erhöht.

Während eines Spiels werden die Szenarien entsprechend der Trigger, der Abbruchkriterien, der Übereinstimmung der Teamziele mit den Szenariozielen und der Übereinstimmung der Team-Strategie mit den Seiteneffekten des Szenarios ausgewählt. Szenarien mit hohen Kosten oder geringem Score sollten nicht genutzt werden.

Ein Coach könnte die gegnerischen Szenarien während eines Spiels modellieren und mit Hilfe von evolutionären Methoden neue Szenarien erstellen.

Ein Problem des szenariobasierten Lösungsansatz ist, dass für jede Situation ein passendes Szenario erstellt werden musste. Da in der SPL keine Coachagent

vorgesehen sind [18], können während des Spiels nur schwer neue Szenarien für neue und unbekannte Situationen entwickelt werden.

2.5 Analytische Lösungen

Die letzte Gruppe der Lösungen leitet die Position der Spieler in Abhängigkeit von der Position wichtiger Objekte, z.B. dem Ball, analytisch ab.

Carlos E. Agüero et al. präsentieren in [2] hauptsächlich ihre Lösung für einen dynamischen Rollentausch, erläutern aber auch kurz die Positionierung des Verteidigers und des Supporters. Der Verteidiger sollte sich auf der Linie zwischen Ball und Tormittelpunkt positionieren, sodass der Ball nicht direkt ins Tor geschossen werden kann (Abbildung 2.2 links). Der Supporter soll während des Angriffs auf dem Mittelpunkt des Rechtecks stehen, welches vom Ball und von der äußeren gegnerischen Ecke aufgespannt wird (Abbildung 2.2 rechts).

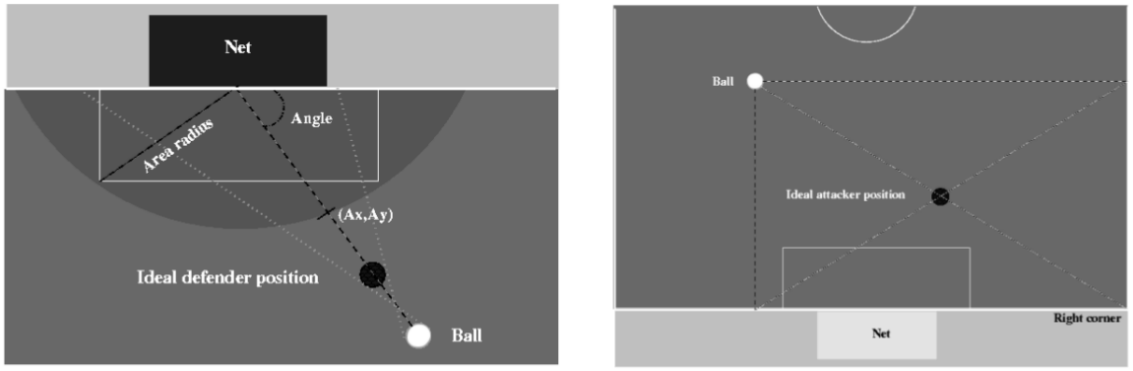


Abbildung 2.2: Links ist die gewünschte Position des Verteidigers und rechts die des Supporters nach [2] zu sehen.

Mike Phillips und Manuela Veloso beschreiben eine Lösung in [24]. Es wird das Feld in eine Offense, eine Goal und eine Defense Region unterteilt (Abbildung 2.3). Das Verhalten des Supporters ist abhängig von der Region, in der sich der Ball befindet. Befindet sich der Ball in der Offense Region, dann bleibt der Supporter auf der x-Höhe des Balls, um einen bestimmten, konstanten y-Wert verschoben. Wenn der Ball innerhalb der Goal Region ist, positioniert sich der Supporter an fixierten Koordinaten in der Nähe der Ecken des Strafraums, um abgeprallte Bälle anzunehmen. Befindet sich der Ball jedoch in der Defense Region, dann bleibt der Supporter in der Offense Region, um den Ball anzunehmen, falls ein verteidigender Spieler den Ball nach vorne spielt. Wenn der Ball dabei eine bestimmte Linie überschreitet und der Supporter droht zwischen Seitenlinie und Striker eingeklemmt zu werden, versucht er hinter dem Striker die Seite zu wechseln.

Abbildung 2.3: Unterteilung des Spielfeldes in Regionen, x zeigt in Richtung des gegnerischen Tors. Wenn der Ball den Pinch Threshold überschreitet, dann startet der Supporter ein Ausweichmanöver und versucht die Seite zu wechseln [24].

Situation Based Strategic Positioning (SBSP) [6, 12] übernimmt wohl am genausten die Konzepte des menschlichen Fußballs. Es werden Teamstrategien, Taktiken, Formationen, Spielzüge (Pläne) sowie Rollen definiert.

Eine Teamstrategie wird durch eine Menge von Taktiken, Aktivierungsregeln für diese Taktiken, Rollen, Gegnermodellierungsstrategien, Teammitgliedermodellierungsstrategien und Kommunikationsprotokollen gegeben. Eine Taktik wird entsprechend ihrer Aktivierungsregeln ausgewählt und besteht aus Formationen, Aktivierungsregeln für diese Formationen und vorgefertigten Plänen. Die Aktivierungsregeln nutzen globale Informationen, vor allem Informationen über den Gegner. Die vorgefertigten Pläne einer Taktik sind definiert durch Aktivierungsregeln, den Positionsentwicklungen, Rollenentwicklungen und Aktionen der Agenten über die Zeit. Jede Formation der Taktik ist durch die Positionierung der Agenten auf dem Feld definiert. Dabei ist die Positionierung des Agenten durch eine Referenzposition, eine Rolle und eine Wichtigkeit gegeben. Die Rolle gibt dabei das Verhalten des Agent vor. In Abhängigkeit der Situation und der Rolle des Agent wird die Referenzposition angepasst. Die ermittelte Position liegt innerhalb eines definierten Bereich des Feldes. SBSP wird teilweise vom CAMBADA Team in der Middel Size League genutzt [17].

K. Petersen, G. Stoll und O. von Stryk [23] positionierten im RoboCup 2009 den Supporter relativ zum Ball, ähnlich der Position des Supporters bei B-Human. Diese Positionierung hat die Vorteile, dass die Position des Strikers als Fehlerquelle ausgeschlossen wird und der Supporter immer noch die Möglichkeit hat, näher an den Ball zu kommen, falls der Striker durch ein Hindernis blockiert werden würde. Dabei kann die Verschiebung in x und y Richtung je nach Situation angepasst werden.

Kapitel 3

Mathematische Grundlagen

In diesem Abschnitt werden die mathematischen Grundlagen erläutert, um das entwickelte Verfahren verstehen zu können. Zunächst wird das Voronoidiagramm und die damit eng in Beziehung stehende Delaunay Triangulation eingeführt, sowie ein Algorithmus zur Berechnung des Voronoidiagramms. Dabei wird sich an die Darstellung aus Computional Geometry: Algorithms and Applications [8] gehalten. Anschließend wird die A*-Suche erläutert. Zuletzt wird das Konzept der Skalar- und Gradientenfelder zur Roboternavigation vorgestellt.

3.1 Voronoidiagramm

Das Voronoidiagramm findet beispielsweise in der Meteorologie, Kristallographie, Festkörperphysik und Informatik Anwendung. Es konnte auch in der Natur beobachtet werden. Für genauere Informationen und weitere Anwendungsbeispiele siehe [4] und [9]. In dieser Arbeit wird das Voronoidiagramm im zweidimensionalen euklidischen Raum verwendet. Es sei $P \subset \mathbb{R}^2$ eine endliche Menge. Das Voronoidiagramm der Menge P unterteilt den $\mathbb{R}^2$ in $|P|$ Zellen, dass heißt, zu jedem $p \in P$ gibt es eine Zelle

$$\mathcal{V}(p) = \{x \in \mathbb{R}^2 | \forall q \in P \text{ mit } q \neq p : |x - p| < |x - q|\} \tag{3.1}$$

(als Beispiel dient Abbildung 3.1). Mit anderen Worten: $\mathcal{V}(p)$ enthält alle Punkte des $\mathbb{R}^2$, die näher an p als an allen anderen Punkten aus P sind. $\mathcal{V}(p)$ heißt Voronoizelle von p. Die Punkte $p \in P$ werden im weiteren Verlauf des Textes Voronoipunkte genannt.

Bei $\mathcal{V}(p)$ handelt es sich um ein möglicherweise unbegrenztes, offenes, konvexes Polygon. Dies ist leichter erkennbar, wenn man $\mathcal{V}(p)$ als Schnitt von Halbebenen betrachtet. Es gilt

$$\mathcal{V}(p) = \bigcap_{q \in P, q \neq p} h(p, q) \tag{3.2}$$

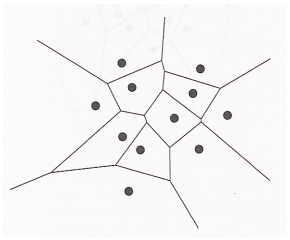


Abbildung 3.1: Beispielvoronoidiagramm aus [8].

wobei $h(p,q)$ die Halbebene ist, die alle Punkte $x \in \mathbb{R}^2$ enthält, für die $|x-p| < |x-q|$ gilt (Abbildung 3.2). Die Gerade $g(p,q)$ mit $(p \neq q)$ ist die Menge alle Punkte, für die $|x-p| = |x-q|$ gilt und ist damit die Grenze der Halbebene $h(p,q)$. Damit liegt $g(p,q)$ zwischen den Voronoizellen von p und q. Somit ist ein Liniensegment von $g(p,q)$ oder ein Strahl auf $g(p,q)$ teil des Voronoidiagramms, da $g(p,q)$ die Zelle $\mathcal{V}(p)$ begrenzt.

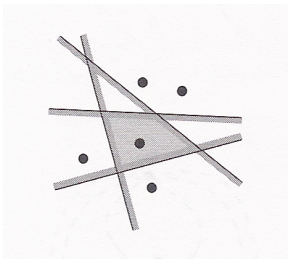


Abbildung 3.2: Veranschaulichung der Definition einer Voronoizelle als Schnitt von Halbebenen [8].

Das Voronoidiagramm kann auch als ein Graph betrachtet werden. Ein Knoten des Voronoidiagramms beziehungsweise des Voronoigraphens ist ein Punkt $c \in \mathbb{R}^2$, der eine offene Kreisfläche $K(c)$ definiert, für die $|\partial K(c) \cap P| \geq 3$ und $K(c) \cap P = \emptyset$ gilt (Abbildung 3.3 rechter Kreis). $\partial K(c)$ ist dabei der Rand der Kreisfläche $K(c)$. Die Gerade $g(p,q)$ definiert eine Kante zwischen p und q falls es einen offene Kreisfläche $K(g)$ mit Mittelpunkt auf $g(p,q)$ gibt, mit $\partial K(g) \cap P = \{p,q\}$ und $K(g) \cap P = \emptyset$ (Abbildung 3.3 linker Kreis). Alle Punkte von $g(p,q)$, die diese Bedingung erfüllen, gehören zur Kante zwischen p und q. Damit enthält eine Kante keine Knotenpunkte und wird höchstens von 2 Knotenpunkten begrenzt.

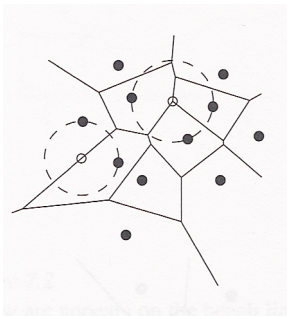


Abbildung 3.3: Veranschaulichung der Definition eines Kantenpunktes und eines Knotenpunktes als Kreismittelpunkte. Der Mittelpunkt des linken Kreises ist ein Kantenpunkt, da zwei Voronoipunkte auf dem Kreis liegen und keine weiteren Voronoipunkte vom Kreis eingeschlossen werden. Der Mittelpunkt des rechten Kreises ist ein Knotenpunkt, da mindestens drei Voronoipunkte auf dem Kreis liegen und keine weiteren Voronoipunkte vom Kreis eingeschlossen werden [8].

3.2 Fortune's Algorithmus

Das Voronoidiagramm einer gegebene Punktemenge, dass heißt dessen Kanten und Knoten, lässt sich mit Hilfe des von Steven Fortune entwickelten Algorithmus in $O(n \log n)$ berechnen [11].

Fortunes Algorithmus ist ein Sweep-Algorithmus, dass heißt, es wird eine Linie (die Sweep-Line) von oben nach unten über die Ebene bewegt. Während die Sweep-Line von oben nach unten bewegt wird, erhält man Informationen über die zu berechnende Struktur. Die Informationen ändern sich nur an bestimmten Punkten, hier Events genannt. Im Allgemeinen hängt die Struktur oberhalb der Sweep-Line nicht von den Informationen unterhalb der Sweep-Line ab.

Die Struktur des Voronoidiagramms kann aber auch von den Informationen unterhalb der Sweep-Line abhängen. Wenn die Sweep-Line den obersten Knoten einer Voronoizelle erreicht, dann liegt der zur Voronoizelle gehörende Voronoipunkt unterhalb der Sweep-Line und es ist nicht möglich den Knoten als solchen zu identifizieren.

Jedoch sind bestimmte Teile des Voronoidiagramms oberhalb der Sweep-Line unabhängig von den Informationen unterhalb der Sweep-Line. Alle Punkte, die über der Sweep-Line und näher an einem bereits bekannten Voronoipunkt als an der Sweep-Line sind, werden nicht mehr von den Informationen unterhalb der Sweep-Line beeinflusst, da alle Voronoipunkte unterhalb der Sweep-Line noch weiter entfernt von den Punkten sind als die Sweep-Line. Die Punktemenge, die näher an einem Voronoipunkt p ist als an der Sweep-Line, wird durch eine Parabel begrenzt. Damit wird die Menge aller Punkte, die näher an irgend einen bekannten Voronoipunkt, als an der Sweep-Line sind, durch eine Sequenz von Parabelbögen begrenzt. Diese Sequenz von Parabelbögen wird Beach-Line genannt.

Der Schnittpunkt zwischen zwei Parabelbögen der Beach-Line liegt genau auf einer Kante des Voronoidiagramms, genauer auf der Kante zwischen den Voronoipunkten, die die Parabelbögen definieren, da dieser Punkt von den beiden Voronoipunkten gleich weit entfernt ist. Die Beach-Line "zeichnet" also das Voronoidiagramm (Abbildung 3.4).

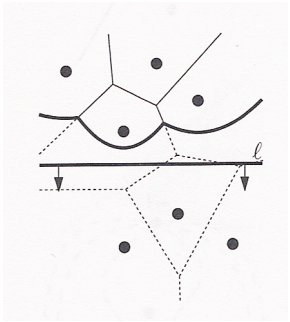


Abbildung 3.4: Die Schnittpunkte der Parabelbögen der Beachline "zeichnen" das Voronoidiagramm. l ist die Sweep-Line [8].

Die Beach-Line ändert sich kontinuierlich, wenn die Sweep-Line von oben nach unten bewegt wird. Jedoch ändert sich ihre Struktur nur dann, wenn ein Parabelbogen hinzugefügt wird (Site Event) oder wenn ein Parabelbogen zu einem Punkt schrumpft und verschwindet (Circle Event).

Ein Site Event passiert, wenn die Sweep-Line auf der Höhe eines neuen Voronoipunktes ist. An der x-Koordinate des Voronoipunktes wird die Beach-Line um eine neue Parabel mit Breite Null (eine senkrechte Strecke) erweitert, da alle Punkte dieser Strecke gleich weit vom neuen Voronoipunkt und von der Sweep-Line entfernt sind. Wenn die Sweep-Line weiter nach unten bewegt wird, weitet sich die Parabel und es wird eine neue Kante "gezeichnet" (Abbildung 3.5).

Die zweite Art von Events ist der Circle Event. Der Circle Event passiert, wenn ein Parabelbogen von seinem linken und rechten Nachbarn "zusammengedrückt" wird und zu einem Punkt "schrumpft" (Abbildung 3.6). Damit dieser Fall eintreten kann, müssen alle drei Voronoipunkte, die die Parabelbögen definieren, unterschiedlich sein. Wenn der mittlere Parabelbogen nun nur noch aus einem Punkt q besteht, dann ist dieser Punkt von allen drei Voronoipunkten und von der Sweep-Line gleich weit entfernt. q ist also der Kreismittelpunkt, der durch die drei Voronoipunkte definiert wird (deshalb Circle Event, Abbildung 3.7). Damit ist q gleichzeitig ein Knoten des Voronoidiagramms und eine neue Kante entsteht, die zwischen den Voronoipunkten der äußeren Parabelbögen liegt.

Die Koordinaten der Site Events sind zu Beginn bekannt, da sie den Koordinaten der Voronoipunkten entsprechen. Die Koordinaten der Circle Events müssen jedoch zur Laufzeit bestimmt werden. Dafür muss nach jedem Hinzufügen eines Parabel-

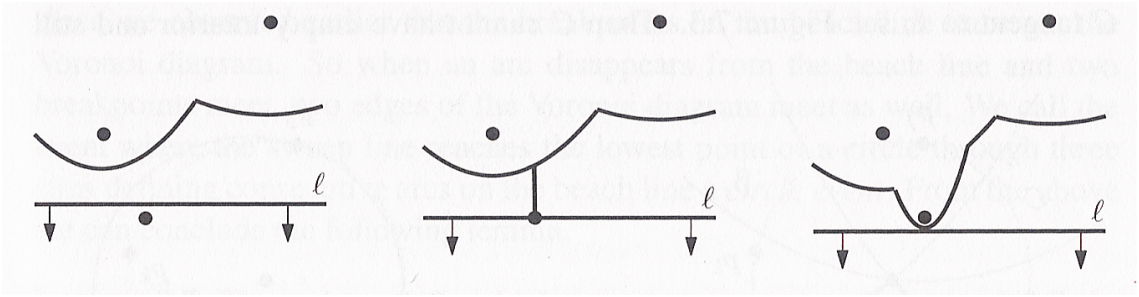


Abbildung 3.5: Veranschaulichung des Verlaufs eines Site Events. Links ist die Sweep-Line kurz vor einem neuen Voronoipunkt. In der Mitte liegt sie genau auf dem Voronoipunkt und eine neuer Parabelbogen wird zur Beach-Line hinzugefügt. Rechts hat die Sweep-Line den Voronoipunkt passiert und der Parabelbogen beginnt sich zu weiten [8].

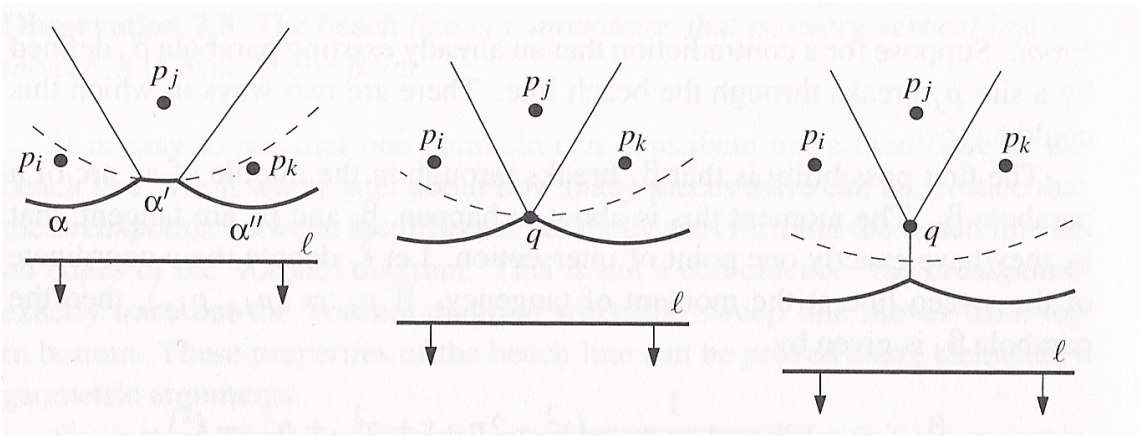


Abbildung 3.6: Der Parabelbogen α' wird von den Parabelbögen α und α'' "zusammengedrückt" und "schrumpft" zu einem Punkt q [8].

bogens (Site Event) überprüft werden, ob die neuen Tripel aufeinanderfolgender Parabelbögen ein Circle Event definieren. Es sei nun (a, b, c) ein solches Tripel, es wurden also a oder c gerade hinzugefügt. (a, b, c) definieren ein Circle Event, falls die Schnittpunkte $x_{a,b}$ und $x_{b,c}$ sich annähern. Wenn jedoch eine neue Parabel eingefügt wird, sodass b aufgeteilt wird, findet b's Circle Event nicht statt, da das Tripel so nicht mehr existiert. Der Circle Event ist dann ein falscher Event und muss ignoriert werden.

Die unterschiedlichen Events werden in einem Heap so sortiert, dass der Heap immer das Event mit der größten y-Koordinate zurück gibt. Dazu muss man sich die falschen Events merken, um sie ignorieren zu können, falls sie das nächste Event wären.

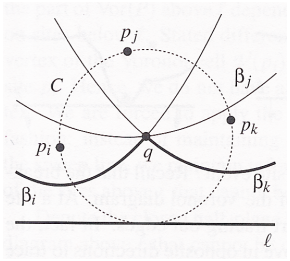


Abbildung 3.7: Der Punkt q ist Kreismittelpunkt des Kreises, der durch p_i, p_j und p_k definiert wird. Damit ist q ein Knotenpunkt des Voronoidiagramms [8].

Die Beach-Line wird durch eine Binärbaum repräsentiert, dessen Blätter die Parabelbögen repräsentieren und die inneren Knoten den Halbkanten entsprechen, die von zwei benachbarten Parabelbögen "gezeichnet" werden. Dabei entspricht das am weitesten links befindliche Blatt dem am weitesten links befindlichen Parabelbogen, das am zweit weitesten links befindliche Blatt, dem am zweit weitesten links befindlichen Parabelbogen und so weiter (Abbildung 3.8).

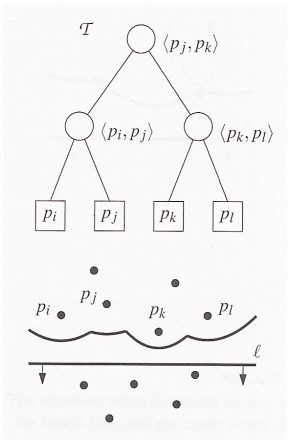


Abbildung 3.8: $\mathcal{T}$ repräsentiert die darunterliegende Beach-Line. Die in-order Traversierung ergibt die Struktur der Beach-Line von links nach rechts [8].

Die Kanten des Voronoidiagramms werden am Ende des Algorithmus aus den zusammengehörenden Halbkanten zusammengesetzt.

3.3 Delaunay Triangulation

Bei einer Triangulation der Punktemenge $P = \{p_1, \cdots, p_n\}$ mit $p_i \in \mathbb{R}^2$ werden die Punkte $p_1, \cdots, p_n$ so untereinander mit Kanten (Liniensegmenten von Geraden) verbunden, dass keine Kanten sich schneiden und keine weiteren Kanten hinzugefügt

werden können, ohne dass eine andere Kante geschnitten wird. Es ist offensichtlich, dass sich die so entstandene Unterteilung des $\mathbb{R}^2$ aus Dreiecken zusammen setzt (für $|P| \geq 3$). Eine Delaunay Triangulation ist eine Triangulation, die den kleinsten Innenwinkel dieser Dreiecke maximiert.

Die Punkte aus P definieren als Voronoipunkte gleichzeitig ein Voronoidiagramm V. Wenn man die Punkte aus P miteinander verbindet, deren Voronoizellen benachbart sind, erhält man den Delaunay Graph der Punktemenge. Der Delaunay Graph ist der duale Graph des Voronoidiagramms, da für jede Fläche (Voronoizelle) des Voronoidiagramms ein Knoten im Delaunay Graphen existiert (der jeweilige Voronoipunkt) und für jede Kante des Voronoidiagramms eine Kante im Delaunay Graphen existiert, der die Voronoipunkte verbindet, die die jeweilige Kante des Voronoidiagramms definieren. Damit existiert aber auch für jeden Knoten des Voronoidiagramms eine Fläche im Delaunay Graphen.

Ist bei allen Knoten des Voronoidiagramms der Grad m kleiner oder gleich Drei, so ist der Delaunay Graph eine Delaunay Triangulation. Für Knoten deren Grad m größer als Drei ist, ist die korrespondierende Fläche des Delaunay Graphen ein konvexes m-Eck. Das heißt aus einem Delaunay Graphen lässt sich leicht eine Delaunay Triangulation erzeugen, indem man die Kanten so zum Graphen hinzufügt, dass die Delaunayeigenschaft erfüllt wird.

3.4 A*-Suche

Diese Beschreibung der A*-Suche lehnt sich an [26] Kapitel 3 und 4 an. Die A*-Suche ist eine heuristische Suche. Der Algorithmus wird genutzt, um den kürzesten Pfade zwischen zwei Knoten in einem Graphen mit positiven Kantengewichten, im weiteren Kosten genannt, zu bestimmen. Die Idee der A*-Suche ist es, abzuschätzen wie hoch die Kosten eines Pfades zum Zielknoten unter Verwendung eines Knotens mindestens sind.

Der Algorithmus unterteilt dafür die Knoten in drei Klassen. Die erste Klasse sind die unbekannten Knoten. Zu diesen Knoten ist noch kein Weg bekannt. Zu Beginn sind alle Knoten unbekannt, außer der Startknoten. Die zweite Klasse von Knoten bilden die Knoten zu denen ein Pfad bekannt ist. Jedem dieser Knoten k wird ein Wert $f(k) = g(k) + h(k)$ zugewiesen. Dabei sind $g(k)$ die Kosten des Pfads vom Startknoten bis zum Knoten k und $h(k)$ ist die Schätzfunktion, die abschätzt wie hoch die Kosten vom Knoten k bis zum Zielknoten mindestens sind. $f(k)$ gibt also an, wie hoch die Kosten im günstigsten Fall sind, wenn der Pfad vom Start zum Zielknoten über den Knoten k geht. Die letzte Klasse von Knoten sind die Knoten, zu denen der kürzeste Pfad bekannt ist.

Der Algorithmus sucht nun in jedem Durchlauf den Knoten k mit kleinstem $f(k)$ aus der Menge der bekannten Knoten und expandiert ihn. Expandieren heißt, dass

alle unbekannten Nachfolgerknoten l zur Menge der bekannten Knoten hinzugefügt werden und deren Wert $f(l)$ bestimmt wird. Existieren schon bekannte Nachfolgerknoten m, so wird überprüft, ob die Kosten im günstigsten Fall ($f(m)$) über den gerade expandierten Knoten k geringer sind, als die über den, in einem vorherigen Durchlauf, ermittelten Pfad zu m. Zum Schluss wird der Knoten k zu der Menge der Knoten hinzugefügt, zu denen der kürzeste Weg bekannt ist. Dies wird solange wieder holt, bis der Zielknoten gefunden wird oder es keine unbekannten Knoten mehr gibt.

Abhängig von der Graphenstruktur und der verwendten Heuristik ist die A*-Suche optimal. Optimal bedeutet, dass die A*-Suche den kürzesten Pfad findet, falls einer existiert. Handelt es sich bei dem Graphen um einen Baum, dann ist die A*-Suche optimal, falls die Heuristik $h(k)$ optimistisch ist. Optimistisch heißt, dass die geschätzten Kosten um vom Knoten k zum Zielknoten zu gelangen kleiner oder gleich den tatsächlichen Kosten sind. Handelt es sich beim Graphen nicht um einen Baum, dann ist die A*-Suche nur dann optimal, falls die Heuristik auch konsistent ist. Eine Heuristik ist konsistent, falls

$$h(k) \leq c(k, k') + h(k') \tag{3.3}$$

gilt. Dies bedeutet, dass die Kosten eines Pfades von k aus über einen anderen Knoten k' zum Zielknoten mindestens so teuer sind, wie die minimalen Kosten vom Knoten k aus.

3.5 Skalarfelder und Gradientenfelder

Skalarfelder und Gradientenfelder werden beispielsweise zur Roboternavigation genutzt. Die Idee ist, dass auf den Roboter anziehende und abstoßende Kräfte wirken. Dabei wirken Ziele anziehend und Hindernisse abstoßend.

Die Kräfte werden dabei durch ein Skalarfeld definiert. Ein Skalarfeld bildet jeden Punkt des Raumes auf ein Skalar ab, welches Potential genannt wird. Es wird für jedes Objekt im Raum (Ziel oder Hindernis) ein Skalarfeld definiert. Das Skalarfeld eines Ziels wird so definiert, dass das Potential eines Punktes mit wachsender Distanz zum Ziel steigt. Im Gegensatz dazu wird das Skalarfeld eines Hindernis so definiert, dass das Potential mit wachsender Distanz sinkt. Je näher ein Punkt also am Hindernis ist, desto größer wird sein Potential. Es seien nun $U_{att} : \mathbb{R}^n \to \mathbb{R}$ das Skalarfeld eines Ziels und $U_{rep} : \mathbb{R}^n \to \mathbb{R}$ das Skalarfeld eines Hindernis. So mit bilden ∇U_{att} und ∇U_{rep} Vektorfelder, die auch Gradientenfelder genannt werden. Dabei zeigen die Vektoren der Gradientenfelder in Richtung des steilsten Anstieges, dass heißt die Vektoren von ∇U_{att} zeigen vom Ziel weg und die Vektoren von ∇U_{rep} zeigen zum Hindernis hin. Durch das Umkehren des Vorzeichens erhält man die gewünschten Vektorfelder. Es gilt somit $F_{att} = -\nabla U_{att}$ und $F_{rep} = -\nabla U_{rep}$.

F_{att} wird auch Attraktorfeld und F_{rep} auch Repulsorfeld genannt. Die resultierende Gesamtkraft F_G, die auf den Agenten wirkt, ergibt sich nun aus der Summe aller anziehenden Felder F_{att} und aller abstoßenden Felder F_{rep}.

Die Summe der Skalarfelder $\sum U$ bildet bildlich gesprochen eine Art Gebirge. Dabei definieren die Hindernisse die Berge und die Ziele die Täler. Der Agent steht auf einem Hang und erfährt eine Kraft, die ihn in Richtung Tal gehen lässt (F_G). Der Agent entfernt sich also von den Hindernissen, die die Berge definieren, beziehungsweise weicht diesen aus und nährt sich, immer hangabwärtsgehend, dem Ziel an. Problematisch sind dabei lokale Minima von $\sum U$. Der Roboter bleibt im lokalem Minimum "gefangen", da an der Stelle des lokalen Minimums $F_G = 0$ ist.

Der Begriff Potentialfeld wird je nach Kontext mal für das Skalarfeld U und mal für das Vektorfeld $-\nabla U$ genutzt. In [13] gibt es weiter Informationen über Potentialfelder, insbesondere was es noch für Typen außer Attraktor- und Repulsorfelder gibt und wie man das Problem mit den lokalen Minima lösen kann.

Kapitel 4

Voronoi Based Situation Map

In diesem Abschnitt wird das im Rahmen dieser Arbeit entwickelte Verfahren genauer erläutert. Abbildung 4.1 veranschaulicht die allgemeine Funktion des Systems. Zu Beginn stehen perzeptuelle Informationen bereit, zum Beispiel über die Ballposition, die eigene Position und die Positionen anderer Spieler auf dem Feld. Diese Informationen werden im nächsten Schritt genutzt, um ein Voronoidiagramm über dem Feld zu erzeugen, womit eine Diskretisierung des Feldes erreicht wird. Außerdem werden die Informationen genutzt, um die Positionierungsstrategie mit Hilfe eines Skalarfelds zu beschreiben. Das Skalarfeld beantwortet, wohin der Roboter soll und wird zusammen mit dem Voronoidiagramm genutzt, um zu beantworten, wie der Roboter dahin kommt. Das Voronoidiagramm wird mit dem Skalarfeld kombiniert, indem jeder Zelle ein Potential zugewiesen wird. Die so entstandene Voronoi Based Situation Map kann nun genutzt werden, um beispielsweise mit Hilfe der A*-Suche einen Pfad von der aktuellen Position zu der Zielposition zu ermitteln. Die Informationen über die Zielposition stammen vom Skalarfeld und werden von der Positionierungsstrategie vorgebeben. Bei der Suche wird der Pfad nicht im Voronoigraphen gesucht, sondern ein Pfad im Delaunay Graphen, der der Nachbarschaftsbeziehung der Zellen entspricht. Verändert sich die Situation auf dem Spielfeld, genauer die Position der Agenten, dann verändert sich die Struktur des Voronoidiagramms und das Skalarfeld. Damit ändert sich aber auch der Pfad, der von der A*-Suche bestimmt wird. In den nächsten Abschnitten werden die einzelnen Teilschritte näher erläutert.

4.1 Diskretisierung des Spielfelds

Durch die Diskretisierung des Feldes wird eine effiziente Suche des Pfades möglich, da nur eine begrenzte Anzahl von Punkten auf dem Feld untersucht werden müssen, um einen Pfad zu finden. Das Spielfeld wird in zwei Schritten diskretisiert. Als Erstes wird ein regelmäßiges Wabenmuster mit 10 mal 10 Waben über das Feld gelegt

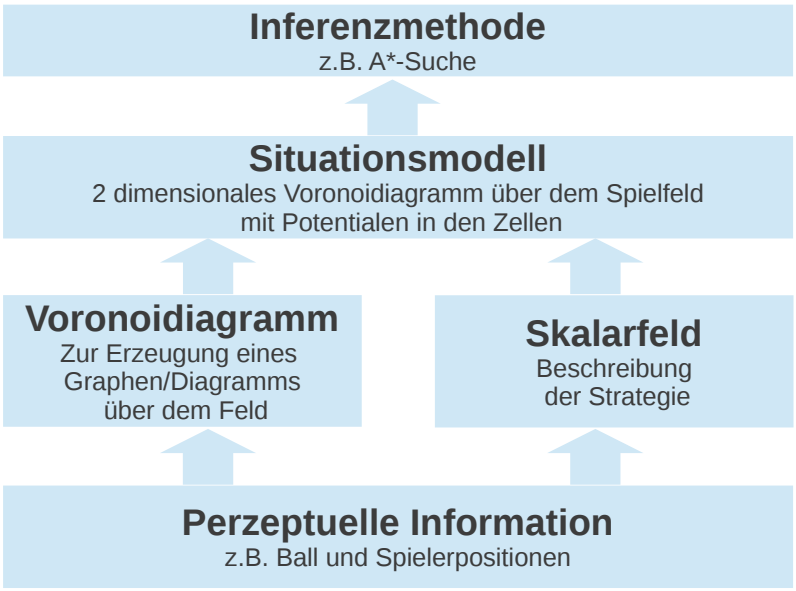


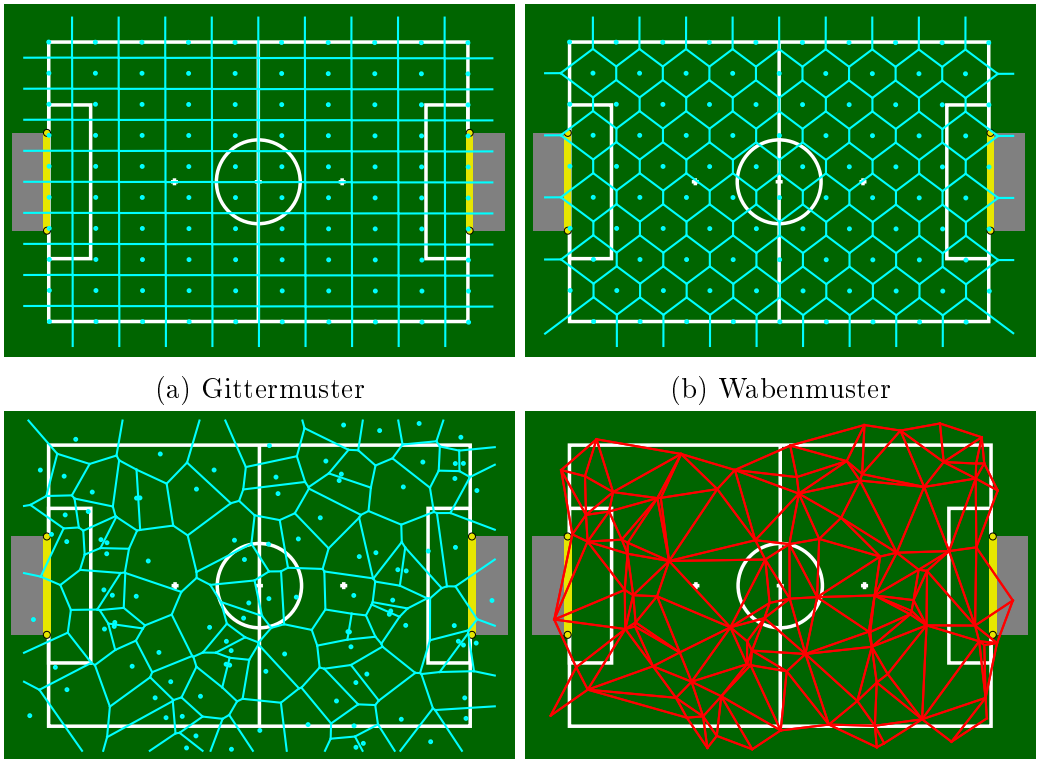
Abbildung 4.1: Übersicht über das System: Perzeptuellen Informationen werden genutzt um Skalarfelder zu erzeugen und ein Voronoidiagramm über dem Feld zu berechnen. Das Voronoidiagramm und die Skalarfelder werden zum Situationsmodell kombiniert, mit dessen Hilfe neue Informationen abgeleitet werden können.

(Abbildung 4.3a), um eine Grundauflösung des Feldes zu erreichen. Anschließend werden die Positionen der bekannten Spieler und auch die eigene Position als Voronoipunkte hinzugefügt. Um diese Positionen werden weitere Voronoipunkte erzeugt (Abbildung 4.3b). Somit ist die Diskretisierung um Spieler herum feiner. Da das Voronoidiagramm später mit dem Skalarfeld kombiniert wird, indem jeder Zelle ein Potential zugewiesen wird, bedeutet das gleichzeitig, dass das Skalarfeld um einen Spieler genauer aufgelöst wird. Damit wird auch der ermittelte Pfad um dem Spieler herum genauer. Die Punkte um einen Spieler werden wie folgt berechnet:

$$p_j = Pos_{Roboter} + \begin{pmatrix} \cos(j\alpha) & -\sin(j\alpha) \\ \sin(j\alpha) & \cos(j\alpha) \end{pmatrix} \begin{pmatrix} 250 \\ 0 \end{pmatrix} \tag{4.1}$$

mit $\alpha = 22.5°$ und $j = 0, \ldots, 15$. Diese Parameter haben sich in den Experimenten als sinnvoll erwiesen. Um den Spieler herum werden also 16 Voronoipunkte äquidistant auf einem Kreis angeordnet.

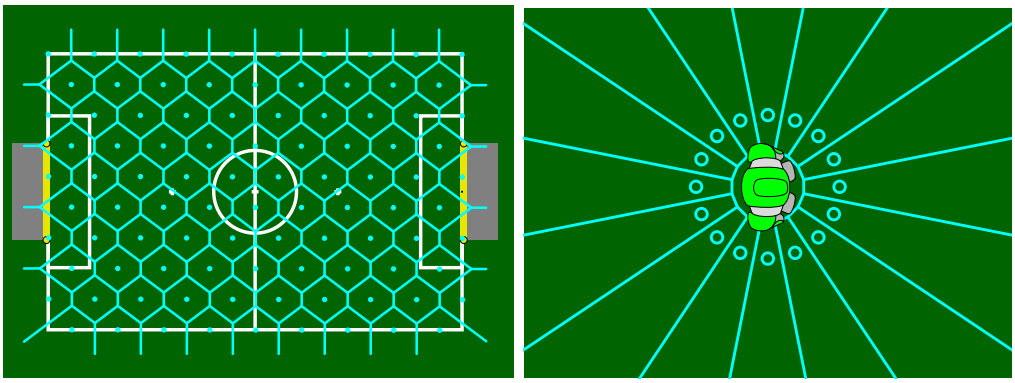
Wie bereits in der Einleitung erwähnt, wurde das Voronoidiagramm gewählt, da es einige angenehme Eigenschaften aufweist. Die Voronoizellen des Diagramms sind konvex und damit eine sinnvolle Diskretisierung des Feldes. Ein weiterer Vorteil des Voronoidigramms ist es, dass man nicht auf regelmäßige, starre Punktemuster begrenzt ist um einen Graphen zu erzeugen (Abbildung 4.2). Insbesondere kann man



(a) Gittermuster (b) Wabenmuster

(c) zufälliges Muster (d) Delaunay-Graph des zufälligen Musters

Abbildung 4.2: Voronoidiagramme, die durch unterschiedliche Voronoipunktemuster erzeugt werden und der Delaunay-Graph des zufälligen Musters.



(a) Das verwendete Hintergrundmuster (b) Das verwendete Agentenmuster

Abbildung 4.3: Das Hintergrundmuster wird mit einem Agentenmuster für jeden Roboter kombiniert.

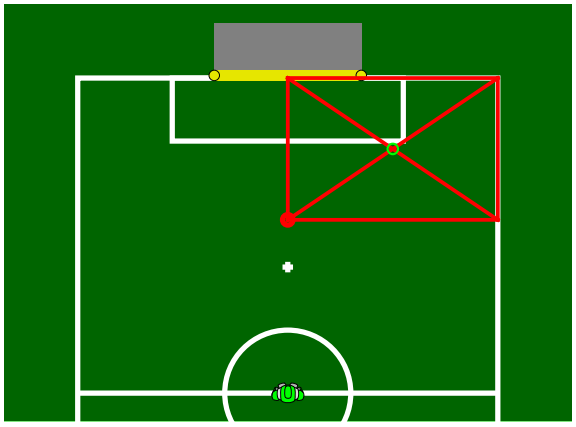


Abbildung 4.4: Die Position, an der sich der Supporter befinden sollte (grüner Kreis), wird durch den Mittelpunkt des roten Rechtecks definiert.

über uninteressanten Bereichen des Feldes wenige Voronoipunkte und über interessanten Bereichen viele Voronoipunkte erzeugen. So erhält man eine Diagramm, welches das Feld über interessanten Stellen feiner und über uninteressanten Stellen grober diskretisiert.

4.2 Positionierungsstrategie

Als Positionierungsstrategie wird die von Carlos E. Agüero et al. in [2] vorgestellte Strategie genutzt. Der Supporter sollte dabei auf dem Mittelpunkt des Rechtecks stehen, das vom Ball und der vom Ball aus gesehen äußeren, gegnerischen Ecke aufgespannt wird (Abbildung 4.4):

$$Pos_{Supporter} = 0.5(Pos_{Ball} + Pos_{Ecke}) \tag{4.2}$$

Befindet sich der Ball auf der Höhe der Linie zwischen den Tormittelpunkten, dann liegt er auf einer Symmetrieachse des Feldes. Da die Lokalisierung des Balls stets fehlerbehaftet ist und die Ballposition somit mal unterhalb und mal oberhalb der Symmetrieachse liegt, wird auch die ermittelte Wunschposition des Supporters mal oberhalb und mal unterhalb liegen. Um dieses springende Verhalten der Zielposition zu verhindern, wird eine Hysterese genutzt. So wechselt der Supporter erst seine Position, falls der Ball auf der Höhe des Pfosten ist, der näher am Supporter steht.

Der von der Positionierungsstrategie ermittelte Punkt ist das globale Minimum eines Skalarfeldes. Andere Spieler, die Pfosten und die Strecke s zwischen Ball und gegnerischem Tor werden als Hindernisse modelliert und als globale Maxima von Skalarfeldern dargestellt (Abbildung 4.5).

Wie schon zuvor erwähnt, wurde das Skalarfeld gewählt, da es schon oft erfolgreich genutzt wurde, um die Positionierungsstrategie zu kodieren. Dies erfolgte zum Beispiel in [27] und [29]. In den genannten Beispielen wurde das Skalarfeld nicht direkt genutzt, sondern dessen Ableitung nach dem Ort. Nichts desto trotz bietet es sich damit als erfolgversprechende Methode an.

Es werden folgende Skalarfelder genutzt:

$$U_{Pos}(p) = \frac{\|p - Pos_{Supporter}\|}{l} \tag{4.3}$$

$$U_X(p) = \begin{cases} e^{\frac{\alpha}{\beta} - \frac{\alpha}{\beta - d_X(p)}} & \text{, für } \beta > d_X(p) \\ 0 & \text{, sonst} \end{cases} \tag{4.4}$$

Dabei ist $\alpha = 800$ und $\beta = 1000$ für $X = Roboter$ und $X = Strecke$. Für $X = Pfosten$ ist $\alpha = 800$ und $\beta = 500$. $d_X(p)$ ist gleich dem minimalen Abstand des Punktes zu X und l gleich der Länge der Felddiagonale.

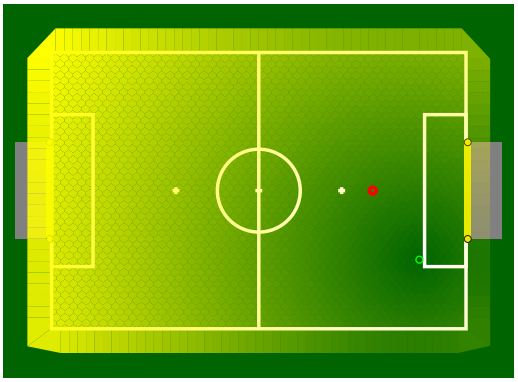
Im Situationsmodell wird nun das Voronoidiagramm mit der Summe der Skalarfelder $\sum U$ kombiniert, indem jeder Voronoizelle $\mathcal{V}(p)$ der aufsummierte Potentialwert $P(p) = \sum U(p)$ ihres Voronoipunktes p zu geordnet wird (Abbildung 4.6).

4.3 Inferenzmethode A*-Suche

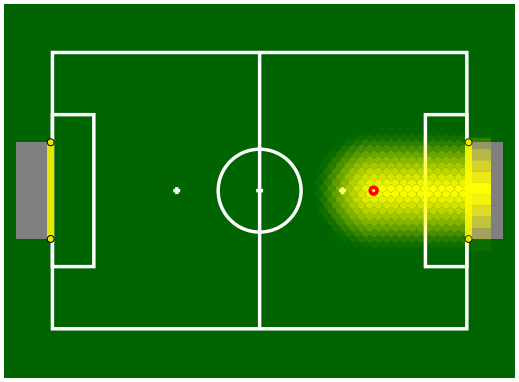
Als Inferenzmethode wird die A*-Suche genutzt. Das Voronoidiagramm teilt das Feld in Regionen auf und der Delaunay-Graph entspricht dem Nachbarschaftsgraphen der Voronoizellen. Wenn der Roboter nun dem Delaunay-Graph folgt, dann wandert er von einer Zelle in die nächste. A* sucht also einen Pfad von der aktuellen Position des Spielers zu der Zielposition auf dem Delanay-Graphen. Die Zielposition ist derjenige Voronoipunkt mit dem kleinsten Potential. Dabei werden die Potentiale in den Zellen als Höheninformation gewertet. Der Delaunay-Graph und die Potentiale werden als ein Gebirge interpretiert, in dem das Potential des jeweiligen Knoten als z-Koordinate im dreidimensionalen euklidischen Raum genutzt wird. Die Kosten c um von einem Knoten k_1 des Delaunay-Graphen mit Koordinaten $p_1 = (x_1, y_1)$ zu einem benachbarten Knoten k_2 mit Koordinaten $p_2 = (x_2, y_2)$ zu gelangen werden definiert als:

$$c(k_1, k_2) = \left\| \begin{pmatrix} x_1 \\ y_1 \\ \alpha P(k_1) \end{pmatrix} - \begin{pmatrix} x_2 \\ y_2 \\ \alpha P(k_2) \end{pmatrix} \right\|_2 \tag{4.5}$$

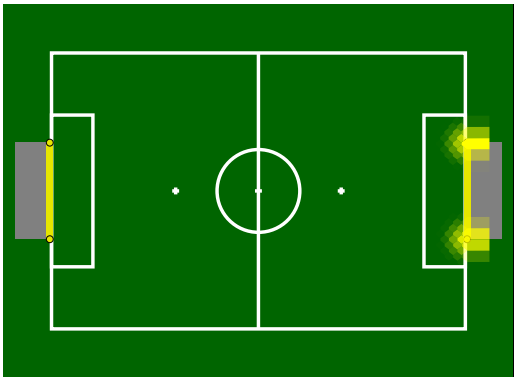
wobei $P(k)$ gleich dem Potential des Knoten k ist und $\alpha \in \mathbb{R}^+$. Anschaulich gesprochen entspricht der euklidische Abstand der Punkte im Gebirge den Kosten. Für die Heuristik wird die gleiche Funktion genutzt, nur dass k_2 den Koordinaten des Zielknotens entspricht.



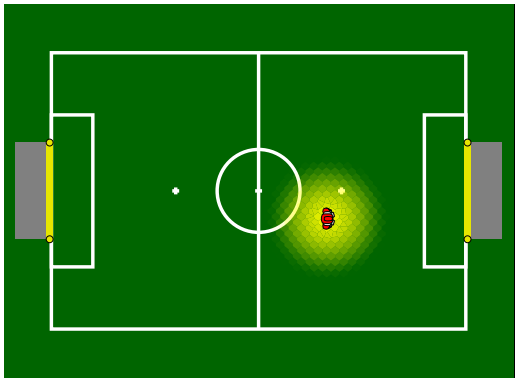
(a) Attraktorfeld der gewünschten Position (hellgrüner Kreis, roter Kreis entspricht dem Ball)



(b) Repulsorfeld der Strecke zwischen Ball und Tormittelpunkt

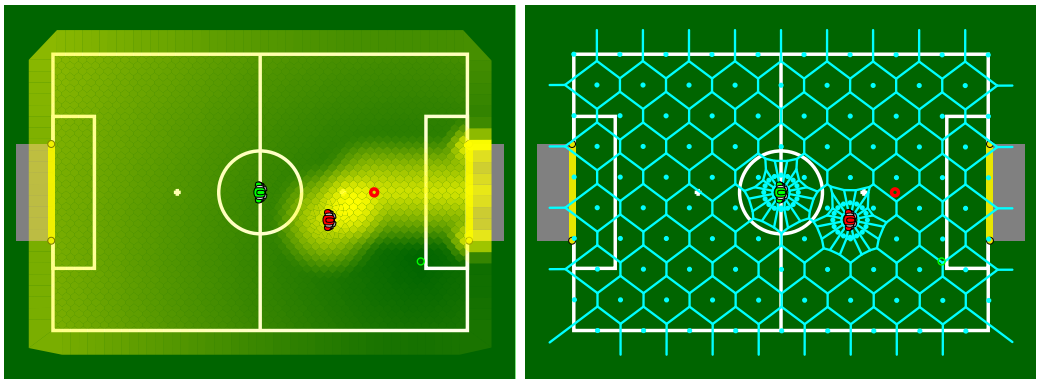


(c) Repulsorfelder der Pfosten



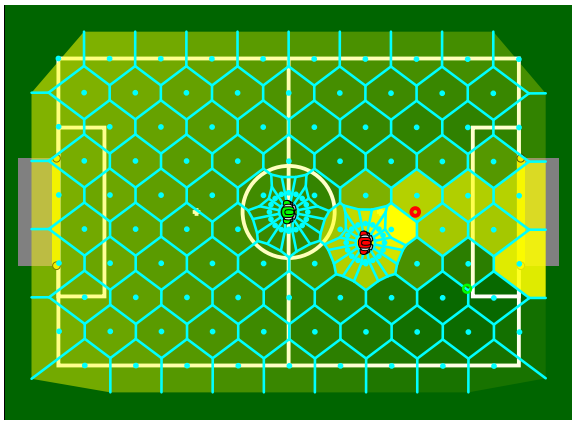
(d) Repulsorfeld eines anderen Spielers

Abbildung 4.5: Darstellung der Skalarfelder, je gelber desto höher ist das Potential.



(a) Summe der Skalarfelder

(b) Kombination des Hintergrundmusters und der Agentenmuster



(c) resultierende Voronoi Based Situation Map

Abbildung 4.6: Das Voronoidiagramm und das Skalarfeld werden mit einander kombiniert und ergeben das Situationsmodell, ein Voronoidiagramm mit Potentialen in den Voronoizellen.

Die Kosten eines Pfades $w = (k_0, k_1, \ldots, k_n)$ werden definiert als:

$$g(w) = \sum_{i=0}^{n-1} c(k_i, k_{i+1}) \tag{4.6}$$

Da es sich bei der Heuristik um den euklidischen Abstand handelt ist sie optimistisch. Da außerdem die Kostenfunktion um von einen Knoten zu einem benachbarten Knoten zu gelangen auch der euklidische Abstand ist, ist die Heuristik auch konsistent, da die Ungleichung 3.4 erfüllt ist. Daraus folgt, dass die A*-Suche den optimalen Pfad findet.

4.4 Umsetzung mit Hilfe des NaoTH-Frameworks

Das Verfahren wurde auf Basis des NaoTH-Frameworks [30, 20] umgesetzt, welches freundlicher Weise vom Nao Team Humboldt zur Verfügung gestellt wurde. Der Vorteil des NaoTH-Frameworks ist, dass mehrere Plattformen unterstützt werden. So läuft das Framework nicht nur auf dem Roboter Nao, sondern auch beispielsweise als Client für den Simulator Simspark.

Das NaoTH-Framework unterteilt sich in zwei Hauptkomponenten, Cognition und Motion. Wie die Namen vermuten lassen, beinhaltet Motion die Komponenten, die für die Ansteuerung der Motorik und Sensorik zuständig sind und Cognition alle Komponenten, die für Informationsauswertung und Verhalten verantwortlich sind. Cognition wiederum setzt sich aus Modulen zusammen, die zur Laufzeit an und ausgeschaltet werden können. Die Module können über Repräsentationen Daten für andere Module bereitstellen.

Die Voronoi Based Situation Map wird im Modul Voronoi Based Situation Map Provider implementiert. Es werden die Repräsentationen PlayersModel, RobotPose, FieldInfo und BallModel genutzt, die vom NaoTH-Framework bereit gestellt werden. PlayersModel stellt Informationen über die Spieler auf dem Feld bereit. Die dort hinterlegten Spielerpositionen werden als Voronoipunkte und zur Erzeugung weiterer Voronoipunkte verwendet. Die Informationen, die PlayersModel bereitstellt, werden vom Modul PlayersLocator ermittelt, indem visuelle Informationen und Informationen, die via TeamComm von den Spielern kommuniziert wurden, kombiniert werden. Die Repräsentation RobotPose stellt Informationen über den Roboter bereit, insbesondere seine Position und Rotation. FieldInfo stellt allgemeine Informationen über das Feld bereit, wie Länge, Breite und Torpositionen. Über BallModel erhält ein Modul Zugriff auf die Information über die Position des Balls in lokalen Koordinaten.

Die c++ Implementierung des Fortune's Algorithmus, um das Voronoidiagramm zu bestimmen, stammt von Ivan Kuckir [14] und wurde im Rahmen der Arbeit erweitert und angepasst.

Kapitel 5

Experimente und Ergebnisse

In diesem Abschnitt wird das Verhalten des Systems in zwei statischen Szenarien mit jeweils zwei Spielern untersucht. Der eine Spieler übernimmt dabei die Rolle des Strikers und der andere die Rolle des Supporters, der die Voronoi Based Situation Map zur Positionierung nutzt. Statisch bedeutet in diesem Fall, dass der Striker und der Ball ihre Position nicht ändern, sondern nur der Supporter. Die ersten beiden Experimente untersuchen den Einfluss des Parameter α der verwendeten Heuristik (Gleichung 4.3) auf den gelaufen Pfad, wenn das Hintergrundmuster nicht geändert wird. Es wird erwartet, dass mit kleine Werten für α der gelaufene Pfad näher am direkten Weg liegt und bei größeren Werten für α der Einfluss des Skalarfeldes steigt und Umwege gelaufen werden. Das dritte Experiment hält α konstant und ändert aber dafür die Auflösung des Hintergrundmusters. Das Hintergrundmuster hat Einfluss auf den gefunden Pfad der A*-Suche. Dieser Einfluss muss untersucht werden.

Die Experimente werden im Simulator Simspark durchgeführt (Abbildung 5.1). Innerhalb Simsparks ist die genaue Position des Agenten bekannt. Der gelaufene Pfad ist die Projektion der Trajektorie des Massenschwerpunktes auf die Ebene des Spielfeldes. Die kleineren sinusartigen Schwingungen des gelaufenen Pfades resultieren aus der Verschiebung des Massenschwerpunktes beim Laufen.

5.1 Experimente

Zu Beginn des ersten Experiments steht der Supporter auf dem Feldmittelpunkt. Der Striker befindet sich näher am Ball, in der gegenerischen Hälfte und zwischen dem Supporter und dessen Zielposition (Abbildung 5.2a). Der Supporter soll vom Feldmittelpunkt aus seine, nach der Positionierungsstrategie ermittelten Position, einnehmen. In der Abbildung 5.3 sind die gelaufenen Pfade für $\alpha = 500, 1000, 2000, 3000$ zu sehen.

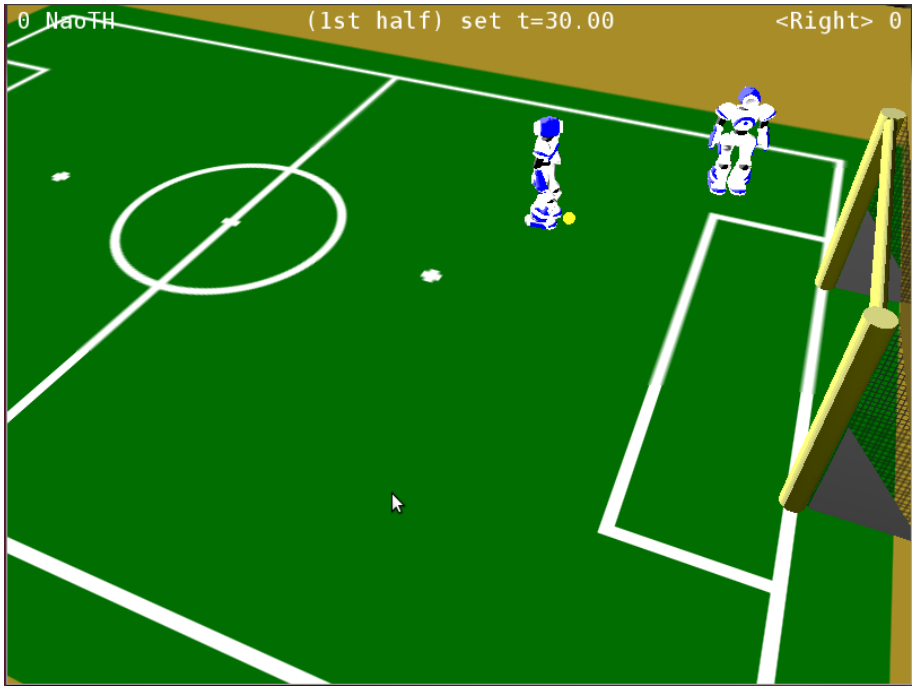


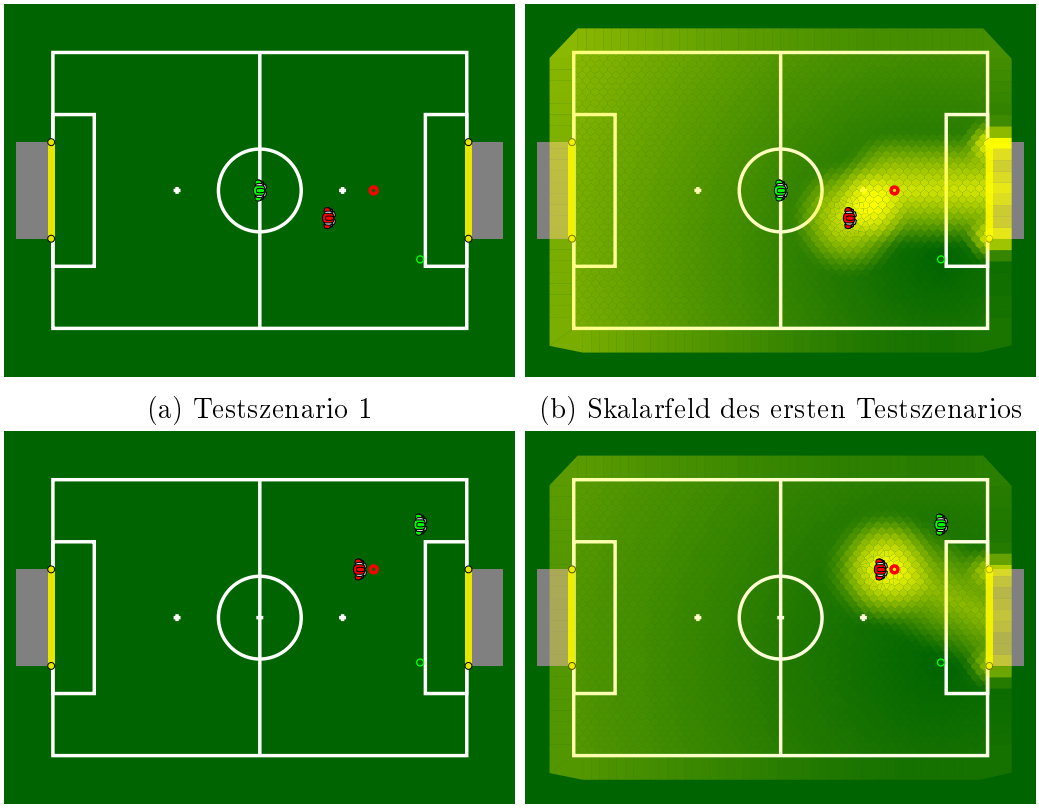
Abbildung 5.1: Die Ausgangssituation des zweiten Testszenarios in Simspark.

Im zweiten Experiment liegt der Ball und steht der Striker auf Höhe des oberen Pfosten. Der Supporter steht auf seiner alten, strategischen Position (Abbildung 5.2c). Da der Ball auf der Höhe des Pfosten ist, liegt seine neue strategische Position auf der anderen Seite des Feldes. In Abbildung 5.4 sind die gelaufenen Pfade für $\alpha = 500, 1000, 2000, 3000$ zu sehen.

Als letztes werden noch unterschiedliche Auflösungen des Hintergrundmusters untersucht. Dabei bleibt $\alpha = 3000$ und nur die Anzahl der Waben ändert sich. Der Supporter soll wie im zweiten Experiment die Seite wechseln. In Abbildung 5.5 sind die gelaufen Pfade des Supporters bei unterschiedlicher Auflösung zu sehen.

5.2 Auswertung und Diskussion

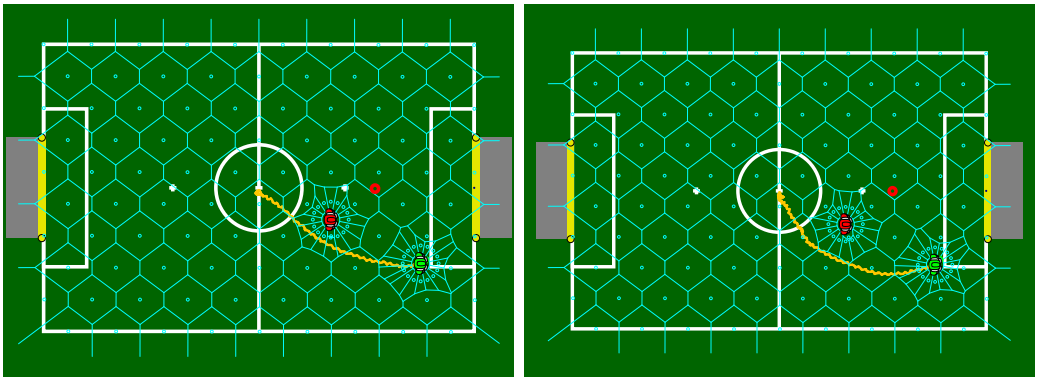
Das erste und zweite Experiment veranschaulichen den Einfluss von α auf den gelaufen Pfad, bei gleichbleibendem Hintergrundmuster. Im ersten Experiment ist zu sehen, dass mit steigendem α der Bogen, den der Supporter um den Striker geht, größer wird. Im zweiten Experiment ist der Einfluss von α noch besser zu sehen als im ersten Experiment. Für $\alpha = 500, 1000, 2000$ wird das Skalarfeld der Linie zwischen Ball und Tor fast ignoriert. Der Supporter wechselt die Seite hinter dem Striker für $\alpha = 3000$.



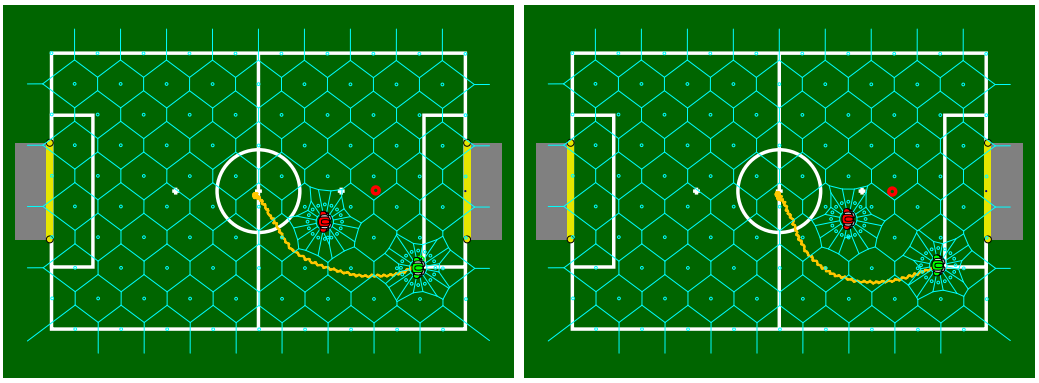
(a) Testszenario 1 (b) Skalarfeld des ersten Testszenarios

(c) Testszenario 2 (d) Skalarfeld des zweiten Testszenarios

Abbildung 5.2: Die zwei Ausgangssituationen der Testszenarien, in denen das statische Verhalten des Systems untersucht wird. Der Supporter soll von seiner Ausgangsposition zur Zielposition (grüner Kreis) laufen. Im ersten Szenario 5.2a sollte der Supporter eine Pfad unterhalb des anderen Spielers wählen. Im zweiten Szenario 5.2c sollte der Supporter hinter dem anderen Spieler die Seite wechseln, also nicht zwischen Ball und Tor laufen. Die Abbildungen 5.2b und 5.2d zeigen die Skalarfelder der Ausgangssituation beider Testszenarien.

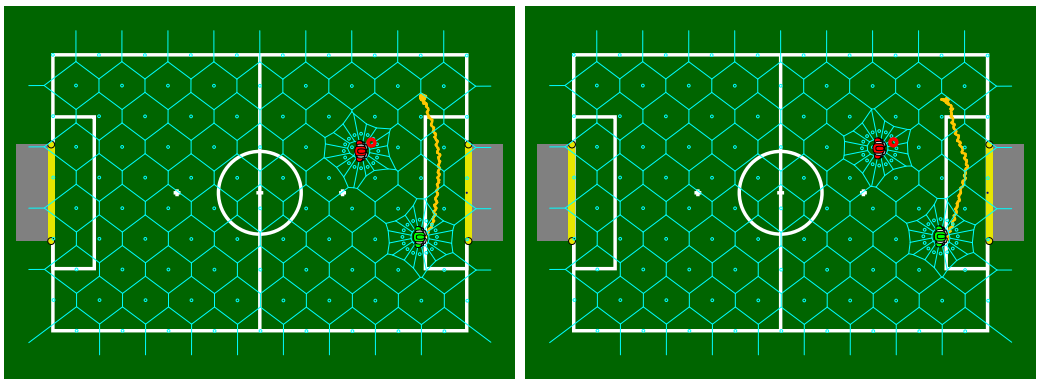


(a) $\alpha = 500$ (b) $\alpha = 1000$

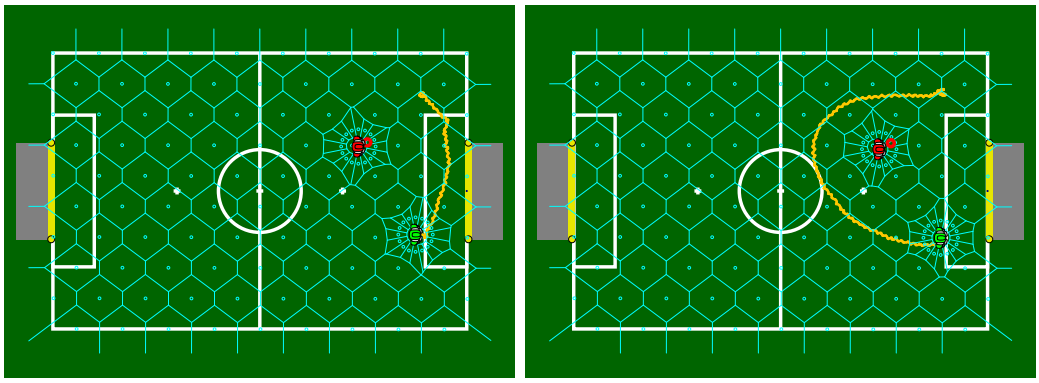


(c) $\alpha = 2000$ (d) $\alpha = 3000$

Abbildung 5.3: Zurückgelegte Pfade des Supporters (orange), um die ermittelte Position einzunehmen, mit $\alpha = 500, 1000, 2000, 3000$.

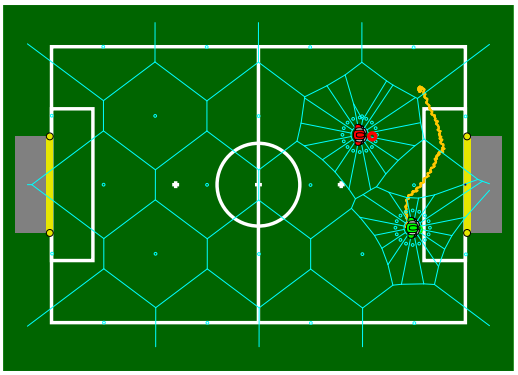


(a) $\alpha = 500$ (b) $\alpha = 1000$

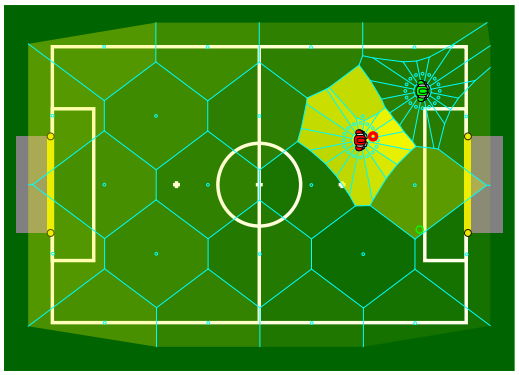


(c) $\alpha = 2000$ (d) $\alpha = 3000$

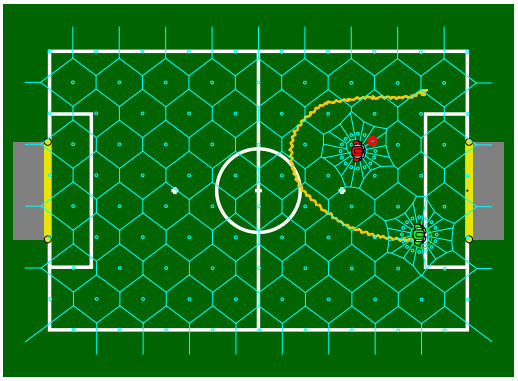
Abbildung 5.4: Zurückgelegte Pfade des Supporters (orange) beim Seitenwechsel mit $\alpha = 500, 1000, 2000, 3000$.



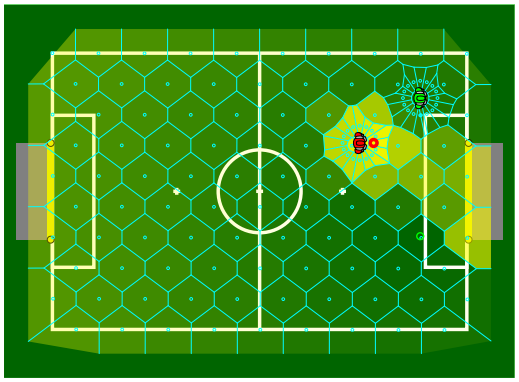
(a) 5 mal 5 Waben



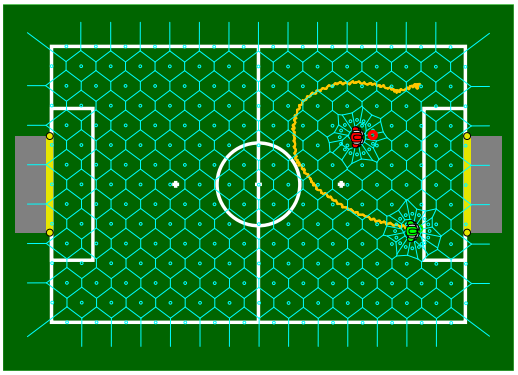
(b) Skalarfeld bei einer Auflösung von 5 mal 5 Waben



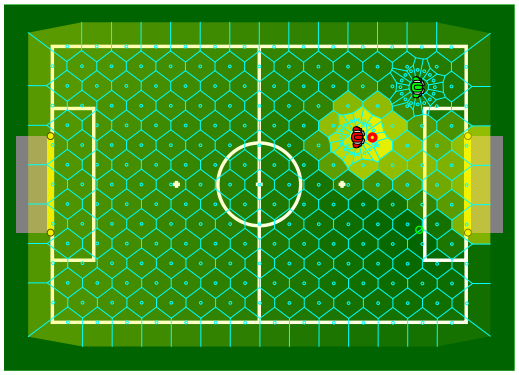
(c) 10 mal 10 Waben



(d) Skalarfeld bei einer Auflösung von 10 mal 10 Waben



(e) 15 mal 15 Waben

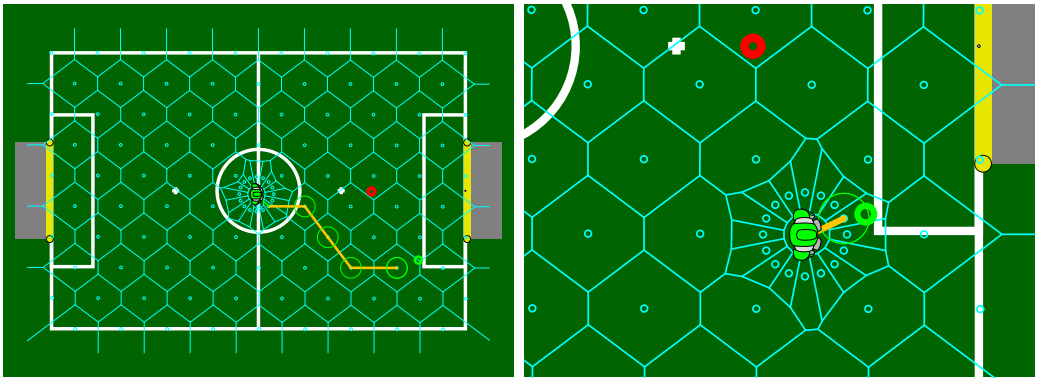


(f) Skalarfeld bei einer Auflösung von 15 mal 15 Waben

Abbildung 5.5: Zurück gelegte Pfade des Supporters (orange) beim Seitenwechsel mit unterschiedlicher Rasterung und $\alpha = 3000$.

Der Parameter α der Heuristik und der Kostenfunktion ist eine Faktor, der anschaulich gesprochen die Höhe der Berge festlegt. Wird α klein gewählt, so kann der Weg um einen anderen Spieler länger erscheinen, als durch den Spieler hindurch zu gehen. Wird jedoch ein großes α gewählt, erscheint der Weg um jenen Spieler kürzer. Im Prinzip bestehen die Kosten und die Heuristik der A*-Suche aus einem konstanten und einem variablen Teil. Der konstante Anteil ist der Abstand der Knoten des Delaunay Graphs in der Ebene. Der variable Anteil wird durch das α dargestellt. Durch die Wahl von α wird der tatsächliche Abstand der Knotenpunkte verzerrt. Schöne Beispiele dafür sind Abbildung 5.3a und 5.3d sowie 5.4a und 5.4d. Bei $\alpha = 500$ ist die Verzerrung des Abstandes noch relativ gering, sodass der Supporter noch vor dem Striker die Seite wechselt (Abbildung 5.4a) beziehungsweise knapp am Supporter vorbei geht (Abbildung 5.3a) und relativ nah am direkten Weg ist. Bei $\alpha = 3000$ wird der Abstand durch α so verzerrt, dass der Bogen, den der Supporter um den Striker geht, größer wird (Abbildung 5.3d). Beim Seitenwechsel verzerrt α den Abstand so stark, dass der Weg um den Striker herum kürzer wirkt, als der Weg, der zwischen Ball und Tor führen würde (Abbidlung 5.4d). Somit beschreibt α mit welchem Anteil das Skalarfeld in die Pfadplanung eingeht. Ist α für ein gegebenes Hintergrundmuster groß genug, so ist der tatsächliche Abstand zweier Knoten nicht relevant für die Pfadplanung und der Roboter orientiert sich nur am Skalarfeld.

Beim experiementieren ist aufgefallen, dass der Agent unter bestimmten Umständen am Ende des Pfades einen kleinen Haken schlägt (zum Beispiel in Abbildung 5.5a). Das Voronoidiagramm wird mit dem Skalarfeld kombiniert, indem jeder Voronoizelle der Potentialwert ihres Voronoipunktes zugewiesen wird. Die Idee des Skalarfeldes ist es, dass die Zielposition als globales Minium des selbigen modelliert wird. Es sei k die Stelle, an der das Skalarfeld ein globales Minimum hat. Mit Hilfe der A*-Suche wird ein Pfad von der aktuelle Position zum Voronoipunkt p mit dem kleinsten Potential gesucht. Da in k das Skalarfeld sein globales Minimum hat, gilt $P(k) \leq P(p)$. Mit anderen Worten, der Zielknoten des ermittelten Pfades muss nicht mit dem Punkt übereinstimmen, der vom Skalarfeld als Zielposition modelliert wird. Ein Beispiel ist in Abbildung 5.6 zu sehen. Das Ende des von der A*-Suche bestimmten Pfads ist der Voronoipunkt mit dem kleinsten Potential, das bekannte globale Minimum. Das tatsächliche globale Minimum des Skalarfeldes stimmt mit dem Bekannten nicht überein und liegt am Rande der Voronoizelle, die vom Zielpunkt des Pfades definiert wird. Der Agent versucht nun das bekannte globale Minimum zu erreichen. Sobald der Agent jenen Punkt erreicht, existiert jedoch schon ein neues bekanntes, globales Minimum, da nun ein Voronoipunkt des Agentmusters näher am tatsächlichen Minimum ist (Abbildung 5.6b). Dieses Verhalten des Systems ist nicht optimal. Jedoch ist es fragwürdig, ob das Hakenlaufen in einem dynamischen Szenario, wie einem Spiel, als problematisch betrachtet werden kann. In einem dynamischen Szenario müsste sich die Supporterposition die meiste Zeit ändern, weil



(a) Das Ende des gefundenen Pfades (orange) ist die Zelle mit dem kleinsten Potential

(b) Da der Zielknoten nicht die Stelle des Minimums ist, gibt es ein Voronoipunkt mit kleinerem Potential, welcher das neue Ziel ist.

Abbildung 5.6: Die Stelle des globalen Minimums (dicker, grüner Kreis) stimmt nicht mit dem Voronoipunkt überein, der das kleinsten Potential hat und somit Zielknoten der A*-Suche ist. Beim Erreichen des Zielknotens besitzt ein Voronoipunkt des Agentenmusters ein kleineres Potential und wird zum neuen Ziel.

sie auf der Position des Balls beruht. Der Supporter müsst also seine strategische Position so gut wie nie erreichen, außer der Ball verharrt für eine Weile an einer Stelle. Dabei handelt es sich hier nur um eine Hypothese, die noch mit dynamischen Experimenten überprüft werden muss.

Der vom Agenten zurückgelegte Pfad ist nicht nur von α abhängig, sondern auch vom zugrundeliegenden Hintergrundmuster, beziehungsweise von dessen Auflösung. Im dritten Experiment wurde α konstant gehalten und die Anzahl der Waben des Hintergrundmusters verändert. Bei einem Hintergrundmuster mit 10 mal 10 oder 15 mal 15 Waben wechselt der Supporter hinter dem Striker die Seite. Wird jedoch ein 5 mal 5 Wabenmuster gewählt, so wechselt der Supporter vor dem Striker die Seite (zwischen Striker und Tor). Das von der Strecke s zwischen Ball und Tormittelpunkt erzeugte Skalarfeld wird nicht aufgelöst und beeinflusst die Pfadbestimmung erst, wenn der Agent nah an der Strecke ist. Zu diesem Zeitpunkt wirkt jedoch der Weg über den von s erzeugten "Berg" kürzer als der Weg um den Striker herum. Es muss also beachtet werden, dass die Auflösung des Hintergrundmusters nicht zu grob gewählt wird. Je kleiner der Wirkungsbereich von Skalarfeldern (insbesondere von repulsiven Skalarfeldern) ist, desto feiner muss das Hintergrundmuster sein. Wird dies, wie in diesem Beispiel, nicht beachtet, so werden diese "wegweisenden" Felder "übersehen" und der Agent geht einen komplett anderen Pfad (Abbildung 5.5a) als bei einer genaueren Auflösung (Abbildung 5.5c und 5.5e). Gleichzeitig gilt jedoch auch, dass eine höherer Auflösung in der praktischen Anwendung nicht unbedingt

ein besseres Resultat liefert. So sind die gelaufenen Pfade für das 10 mal 10 und das 15 mal 15 Wabenmuster sehr ähnlich.

Dieses Phänomen lässt sich auch unter einem anderen Gesichtspunkt betrachten. Da jeder Voronoizelle des Potential ihres Voronoipunktes zugewiesen wird, diskretisiert das Voronoidiagramm nicht nur die Ebene des Spielfelds sondern auch das Skalarfeld. Das Skalarfeld wird also mit Hilfe der Voronoipunkte abgetastet. Betrachtet man das Skalarfeld als zweidimensionales Signal, so wird bei einer zu groben Auflösung des Hintergrundmusters, also bei zu wenigen Abtastpunkten, das Nyquist-Shannon-Abtasttheorem verletzt. Das Nyquist-Shannon-Abtasttheorem besagt, dass die Abtastfrequenz mindestens zweimal größer sein muss als die höchste Frequenz des Signals, damit das Signal aus den abgetasteten Werten rekonstruiert werden kann.

Zum Schluss werden noch kurz die der Voronoi Based Situation Map zugrunde liegenden Daten betrachtet. Die Qualität der Daten von PlayersModel, RobotPose und BallModel hängen maßgebend von der Selbstlokalisierung der Roboter ab. Wenn sich ein Roboter falsch lokalisiert, so kann das Situationsmodell diesen Fehler nicht kompensieren. Dies äußert sich zum Beispiel darin, dass sich das Voronoidiagramm sprungartig ändert, falls ein Roboter umfällt, da sich dieser dann nicht korrekt lokalisiert. Ein weiterer Punkt ist, dass nur in der Simulation gegnerische Spieler erkannt werden. Derzeit ist das NaoTH-Framework nicht in der Lage Spieler auf dem Nao zu erkennen und zu identifizieren.

Kapitel 6

Zusammenfassung und Ausblick

In dieser Arbeit wurde ein neues Verfahren zur strategischen Positionierung von Robotern im RoboCup vorgestellt. Es wird ein Voronoidiagramm genutzt, um das Feld zu diskretisieren und eine Graphenstruktur über dem Feld zu erzeugen. Mit Hilfe eines Skalarfeldes wird die Positionierungsstrategie beschrieben. Das Voronoidiagramm und das Skalarfeld werden zur Voronoi Based Situation Map kombiniert. Die Voronoi Based Situation Map ist eine Repräsentation der aktuellen Situation auf dem Feld und ermöglicht eine effiziente Pfadsuche, welche das Skalarfeld berücksichtigt. Das Verfahren wurde in einer Simulation mit statischer Umgebung betrachtet. So wurde untersucht, welchen Einfluss die Heuristik und das Hintergrundmuster auf den resultierenden Pfad haben. Es stellte sich heraus, dass der resultierende Pfad, durch die Gewichtung des Anteils des Skalarfeldes in der definierten Metrik der A*-Heuristik, beziehungweise der Kostenfunktion, gesteuert werden kann. Ist der Anteil des Skalarfeldes gering, so nähert sich der resultierende Pfad der direkten Strecke zum Ziel an. Ist der Anteil des Skalarfeldes groß, so werden die Hindernisse, welche im Skalarfeld als lokale Maxima modelliert werden, mehr beachtet und der Agent weicht diesen aus. Des Weiteren wurde gezeigt, dass das Hintergrundmuster nicht zu grob gewählt werden darf.

Die hier vorgestellt Referenzimplementierung scheint vielversprechend zu sein und es lohnt sich durchaus diesen Ansatz weiter zu verfolgen. Als nächstes muss untersucht werden, wie sich das Verfahren in einer dynamischen Situation verhält. Unter Umständen bietet es sich an, das Verfahren zuvor dahingehend zu optimieren, dass es auf den Roboter Nao genutzt werden kann. So könnte das Verhalten des Verfahrens in dynamischen Situation und gleichzeitig auf die reale Anwendbarkeit untersucht werden.

Wurde die Anwendbarkeit bestätigt, kann das Verfahren weiter verbessert werden. Dabei kann sowohl die Methode zur Erstellung der Graphenstruktur, als auch die verwendete Positionierungstrategie weiterentwickelt werden.

Da es sich um eine Referenzimplementierung handelt, wurde eine relativ einfa-

che Positionierungsstrategie gewählt. Diese Positionierungsstrategie berücksichtigt jedoch gegnerische Spieler nur indirekt durch das Skalarfeld und basiert vollkommen auf der Position des Balls. Es sollte also eine raffinierte Positionierungsstrategie entwickelt beziehungsweise aus anderen Arbeiten übernommen werden, die Mitspieler und Gegner beachtet und unterschiedlich behandelt.

Auch die verwendete Graphenstruktur bietet noch Möglichkeiten zur Verbesserung. So muss der Einfluss des Hintergrund- und des Agentenmusters genauer untersucht werden, insbesondere die Eigenschaften unterschiedlicher Muster. Außerdem sind das Hintergrund- und Agentenmuster entkoppelt, dass heißt, ein Voronoipunkt des Agentenmusters kann beliebig nah an einem Voronoipunkt des Hintergrundmusters liegen. Dies ist nicht optimal und könnte zum Beispiel mit Hilfe eines centroidal Voronoi Diagram wie in [6] gelöst werden.

Ein mögliches Agentenmuster wäre es, die Punkte nicht um die Spielerposition zu erzeugen, sondern entlang des von der A*-Suche bestimmten Pfads. Generell könnten andere Methoden zur Graphenerstellung untersucht werden, insbesondere adaptive Verfahren. Dadurch müsste der Graph nicht in jedem Zyklus komplett neu bestimmt werden.

Adaptive Verfahren würden es ermöglichen in den Zellen Informationen effizient zu speichern. Zum Beispiel könnte in den Zellen die Wahrscheinlichkeit, dass der Ball sich in ihr befindet, gespeichert werden. Da die Zellen einen Bereich auf dem Feld repräsentieren, ist es denkbar mit ihrer Hilfe den freien Raum zwischen den Spielern zu modellieren und in die Positionierungsstrategie einzubeziehen.

Literaturverzeichnis

[1] RoboCup Soccer Simulation League 3D. Robocup soccer simulation league 3d competition rules and setup. `http://hedayat.fedorapeople.org/misc/rc2012rules-v1.0.pdf` (09.01.2013), 2012.

[2] Carlos E. Agüero, Vicente Matellán, José M. Ca^nas, Víctor M. Gómez, and Juan Carlos. Switch! dynamic roles exchange among cooperative robots. In *in Proceedings of the 2nd International Workshop on Multi-Agent Robotic Systems - MARS 2006. INSTICC*, pages 99–105. Press, 2006.

[3] Hidehisa Akiyama and Itsuki Noda. Multi-agent positioning mechanism in the dynamic environment. In Ubbo Visser, Fernando Ribeiro, Takeshi Ohashi, and Frank Dellaert, editors, *RoboCup 2007: Robot Soccer World Cup XI*, volume 5001 of *Lecture Notes in Computer Science*, pages 377–384. Springer Berlin / Heidelberg, 2008.

[4] Franz Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.

[5] D. Brüggemann and D. Albrecht. *Fußball-Handbuch 1. Modernes Fußballtraining: Das systematische Lehrbuch für Trainer, Übungsleiter, Sportlehrer, Sportstudenten und Spieler.* Fußball-Handbuch. Hofmann-Verlag Schorndorf, 4. unveränderte auflage edition, 1998.

[6] Hesam T. Dashti, Shahin Kamali, and Nima Aghaeepour. Positioning in robots soccer. In Pedro Lima, editor, *Robotic Soccer*, pages 29–44. I-Tech Education and Publishing, 2007.

[7] HesamAddin Dashti, Nima Aghaeepour, Sahar Asadi, Meysam Bastani, Zahra Delafkar, Fatemeh Disfani, Serveh Ghaderi, Shahin Kamali, Sepideh Pashami, and Alireza Siahpirani. Dynamic positioning based on voronoi cells (dpvc). In Ansgar Bredenfeld, Adam Jacoff, Itsuki Noda, and Yasutake Takahashi, editors, *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *Lecture Notes in Computer Science*, pages 219–229. Springer Berlin / Heidelberg, 2006.

[8] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer Berlin / Heidelberg, second, revised edition, 2000.

[9] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Rev.*, 41(4):637–676, December 1999.

[10] The RoboCup Federation. Robocup. `www.robocup.org` (09.01.2013).

[11] Steven Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[12] Markus Hannebauer, Jan Wendler, Enrico Pagello, Luís Reis, Nuno Lau, and Eugénio Oliveira. Situation based strategic positioning for coordinating a team of homogeneous agents. In *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, volume 2103 of *Lecture Notes in Computer Science*, pages 175–197. Springer Berlin / Heidelberg, 2001.

[13] T. Hellström. Robot navigation with potential fields. Uminf-11.18, Department of Computing Science, Umeå University, 2011.

[14] Ivan Kuckir. C++ implementation of fortune's algorithm. `http://www.ivank.net/blogspot/vor/voronoi.zip` (15.11.2012).

[15] Vadim Kyrylov and Eddie Hou. Pareto-optimal collaborative defensive player positioning in simulated soccer. In Jacky Baltes, Michail Lagoudakis, Tadashi Naruse, and Saeed Ghidary, editors, *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science*, pages 179–191. Springer Berlin / Heidelberg, 2010.

[16] Vadim Kyrylov and Serguei Razykov. Pareto-optimal offensive player positioning in simulated soccer. In Ubbo Visser, Fernando Ribeiro, Takeshi Ohashi, and Frank Dellaert, editors, *RoboCup 2007: Robot Soccer World Cup XI*, volume 5001 of *Lecture Notes in Computer Science*, pages 228–237. Springer Berlin / Heidelberg, 2008.

[17] Nuno Lau, Luís Seabra Lopes, Nelson Filipe, and Gustavo Corrente. Roles, positionings and set plays to coordinate a robocup msl team. In Luís Lopes, Nuno Lau, Pedro Mariano, and Luís Rocha, editors, *Progress in Artificial Intelligence*, volume 5816 of *Lecture Notes in Computer Science*, pages 323–337. Springer Berlin / Heidelberg, 2009.

[18] RoboCup Soccer Standard Platform League. Robocup standard platform league (nao) rule book. `http://www.tzi.de/spl/pub/Website/Downloads/Rules2012.pdf` (09.01.2013), 2012.

[19] Colin McMillen and Manuela Veloso. Distributed, play-based coordination for robot teams in dynamic environments. In Gerhard Lakemeyer, Elizabeth Sklar, Domenico Sorrenti, and Tomoichi Takahashi, editors, *RoboCup 2006: Robot Soccer World Cup X*, volume 4434 of *Lecture Notes in Computer Science*, pages 483–490. Springer Berlin / Heidelberg, 2007.

[20] Heinrich Mellmann, Yuan Xu, Thomas Krause, and Florian Holzhauer. Naoth software architecture for an autonomous agent. In *Proceedings of the International Workshop on Standards and Common Platforms for Robotics (SCPR 2010)*, Darmstadt, November 2010.

[21] Ryota Nakanishi, James Bruce, Kazuhito Murakami, Tadashi Naruse, and Manuela Veloso. Cooperative 3-robot passing and shooting in the robocup small size league. In Gerhard Lakemeyer, Elizabeth Sklar, Domenico Sorrenti, and Tomoichi Takahashi, editors, *RoboCup 2006: Robot Soccer World Cup X*, volume 4434 of *Lecture Notes in Computer Science*, pages 418–425. Springer Berlin / Heidelberg, 2007.

[22] Ryota Nakanishi, Kazuhito Murakami, and Tadashi Naruse. Dynamic positioning method based on dominant region diagram to realize successful cooperative play. In Ubbo Visser, Fernando Ribeiro, Takeshi Ohashi, and Frank Dellaert, editors, *RoboCup 2007: Robot Soccer World Cup XI*, volume 5001 of *Lecture Notes in Computer Science*, pages 488–495. Springer Berlin / Heidelberg, 2008.

[23] Karen Petersen, Georg Stoll, and Oskar von Stryk. A supporter behavior for soccer playing humanoid robots. In Javier Ruiz-del Solar, Eric Chown, and Paul Plöger, editors, *RoboCup 2010: Robot Soccer World Cup XIV*, volume 6556 of *Lecture Notes in Computer Science*, pages 386–396. Springer Berlin / Heidelberg, 2011.

[24] Mike Phillips and Manuela Veloso. Robust supporting role in coordinated two-robot soccer attack. In Luca Iocchi, Hitoshi Matsubara, Alfredo Weitzenfeld, and Changjiu Zhou, editors, *RoboCup 2008: Robot Soccer World Cup XII*, volume 5399 of *Lecture Notes in Computer Science*, pages 235–246. Springer Berlin / Heidelberg, 2009.

[25] Ali Rad, Navid Qaragozlou, and Maryam Zaheri. Scenario-based teamworking, how to learn, create, and teach complex plans? In Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *Lecture Notes in Computer Science*, pages 137–144. Springer Berlin / Heidelberg, 2004.

[26] Stuart Russell and Peter Norvig. *Künstliche Intelligenz. Ein moderner Ansatz.* Pearson Studium, 2. edition, 2004.

[27] T. Röfer, T. Laue, J. Müller, A. Fabisch, F. Feldpausch, K. Gillmann, C. Graf, T. J. de Haas, A. Härtl, A. Humann, D. Honsel, P. Kastner, T. Kastner, C. Könemann, B. Markowsky, O. J. L. Riemann, and F. Wenk. B-human team report and code release 2011. `http://www.b-human.de/downloads/bhuman11_coderelease.pdf` (26.07.2012), 2011.

[28] Chieh-Chih Wang, Shao-Chen Wang, Chun-Hua Chang, Bo-Wei Wang, Hsin-Cheng Chao, and Chih-Chung Chou. Team report and code release 2011. `http://www.csie.ntu.edu.tw/~bobwang/RoboCupSPL/NTURoboPAL_TeamReport2011.pdf` (26.07.2012), 2011.

[29] Henry Work, Eric Chown, Tucker Hermans, Jesse Butterfield, and Mark McGranaghan. Player positioning in the four-legged league. In Luca Iocchi, Hitoshi Matsubara, Alfredo Weitzenfeld, and Changjiu Zhou, editors, *RoboCup 2008: Robot Soccer World Cup XII*, volume 5399 of *Lecture Notes in Computer Science*, pages 391–402. Springer Berlin / Heidelberg, 2009.

[30] Yuan Xu, Heinrich Mellmann, and Hans-Dieter Burkhard. An approach to close the gap between simulation and real robots. In Noriaki Ando, Stephen Balakirsky, Thomas Hemker, Monica Reggiani, and Oskar Stryk, editors, *Simulation, Modeling, and Programming for Autonomous Robots*, volume 6472 of *Lecture Notes in Computer Science*, pages 533–544. Springer Berlin Heidelberg, 2010.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Weiterhin erkläre ich, eine Bachelorarbeit in diesem Studiengebiet erstmalig einzureichen.

Berlin, den 11. Februar 2013 .....

Statement of authorship

I declare that I completed this thesis on my own and that information which has been directly or indirectly taken from other sources has been noted as such. Neither this nor a similar work has been presented to an examination committee.

Berlin, 11th February 2013 .....