Task Manager

Einleitung

Das Ziel des Taskmanagers besteht in der Zuweisung von Tasks (Aufgaben) an die verschiedenen Roboter. Die Taskzuweisung basiert dabei auf den Daten des Observer. Im Taskmanager wird beispielsweise entschieden ob und wohin ein Torschuss erfolgen sollte, welche Gegner verteidigt werden müssen oder wo sich Roboter positionieren sollten. Eine wesentliche Herausforderung besteht darin, die einzelnen Roboter gemeinsam als Team zu koordinieren und dabei auch stets den Gegner und dessen potenziellen Aktionen zu berücksichtigen.

Bangkok 2022

Die erste Version der Strategie wurde für den Robocup in Bangkok entwickelt und dort auch erstmalig getestet. Das Grundprinzip basierte damals auf einem Rule System, durch das alle Roboter zentral gesteuert wurden. Dabei wurde eine Vielzahl an Regeln definiert, die angeben unter welchen Umständen welche Tasks ausgeführt werden sollte. Die Vielzahl an notwendigen Regeln resultierte dabei jedoch in einem unübersichtlichem und schwer erweiterbarem Code, weshalb basierend auf den Erfahrungen eine Architektur mit einer besseren Kapselung und einem mehrschrittigen Entscheidungsprozess entwickelt wurde.

Funktionsweise

Als Architektur für die Entscheidungsfindung wurde ein Rule System verwendet. Dabei handelt es sich um ein System, bei der sequentiell verschiedene Bedingungen geprüft werden. Trifft eine Bedingung zu, so wird eine zugehörige Aktion einem Roboter zugewiesen. Mögliche Bedingungen hierbei ist die Höhe der Torwahrscheinlichkeit, das Team mit Ballbesitz, Anzahl der Gegner in der eigenen Spielhälfte etc.

Probleme und Herausforderungen

Im Wettbewerb und in der Entwicklung zuvor haben sich signifikante Probleme in der Strategie gezeigt. Diese sollen im Folgenden betrachtet werden.

Strategie und Architektur

Das Rule-System hat in der Praxis einige Probleme offenbart. Die hohe Komplexität des Roboterfußballs sowie die Notwendigkeit zu Steuerung eines kompletten Teams resultierten in einem sehr unübersichtlichem und schwer erweiterbarem Code. Dies erschwerte kleine Änderungen erheblich. Da das Rule-System für alle Roboter gleichermaßen galt war der Code zudem schlecht gekapselt, sodass die Entwicklung des Angriffs und der Verteidigung nicht parallel und unabhängig voneinander durchgeführt werden konnte. Stattdessen haben sich die Änderungen am Verteidigungsverhalten auch stets auf den Angriff ausgewirkt und vice versa.

Eine weitere Problematik bestand darin, dass die verschiedenen Rules keine Kentnisse über die Bedürfnisse der nachfolgenden Rules haben. Stattdessen nimmt jede Rule an, dass sie den für sich am besten geeigneten und noch bislang nicht verwendeten Roboter für die ausführung eines Tasks einplanen kann, selbst wenn dieser Roboter einen Task von einer nachfolgenden Rule mit niedriger Priorität deutlich besser ausführen könnte. Dies resultiert in einer oftmals schlechten und ineffizienten Zuweisung der einzelnen Roboter.

Auch problematisch an diesem Ansatz war, dass instabile Zustände auftreten konnten, in der einzelne Rule Bedingungen sich mehrmals pro Sekunde ändern. Dadurch werden auch Roboter immer wieder als verfügbar oder nicht-verfügbar markiert, wodurch auch die nachfolgenden Rules ihre angefragten Roboter möglicherweise mehrmals pro Sekunde ändern. Diese ständig neuen Task-Zuweisungen führen dazu, dass letztenendes keiner der Tasks richtig ausgeführt wird und die Roboter zwischen verschiedenen Orten pendeln, ohne dabei aktiv zum Spielerfolg beizutragen.

Generelle Spielentscheidungen

In den Spielen selbst wurden auch nicht-ideale Entscheidungen getroffen, wodurch Chancen nicht genutzt und Vorteile verloren wurden. Besonders auffällig dabei waren die folgenden Probleme:

- Genereller Mangel an Torschüssen, obwohl die Trefferwahrscheinlichkeit aus menschlicher Sicht gut aussah; Hier werden bessere Heuristiken benötigt.
- Zu viele Fahrten mit Ball; Dies ist im Allgemeinen langsam und insbesondere Drehungen resultieren oftmals in einem unnötigen Ballverlust.
- Schlechte Positionierungen Offensive; Roboter positionieren sich nicht sinnvoll als Passstation; Die Verteidigung der Gegner wurde kaum berücksichtigt, wodurch die Roboter nicht unbedingt anspielbar waren.
- Verstoß gegen das Regelwerk. Vor allem die Double-Touch-Regel, Aimless Kick und Strafraumvermeidung wurden nicht zuverlässig beachtet.
- Defensive war an sich okay aber zu sehr auf Manndeckung ausgelegt. Direkte Torschüsse wurden oft nur vom Torwart verhindert, hier wären Mauern sinnvoll.
- Spieler behindern sich teils gegenseitig und nehmen Mitspielern den Ball weg.
- Keine Berücksichtigung von Gegnerischen Lupfern.

Zielschlüsse und Erkenntnisse

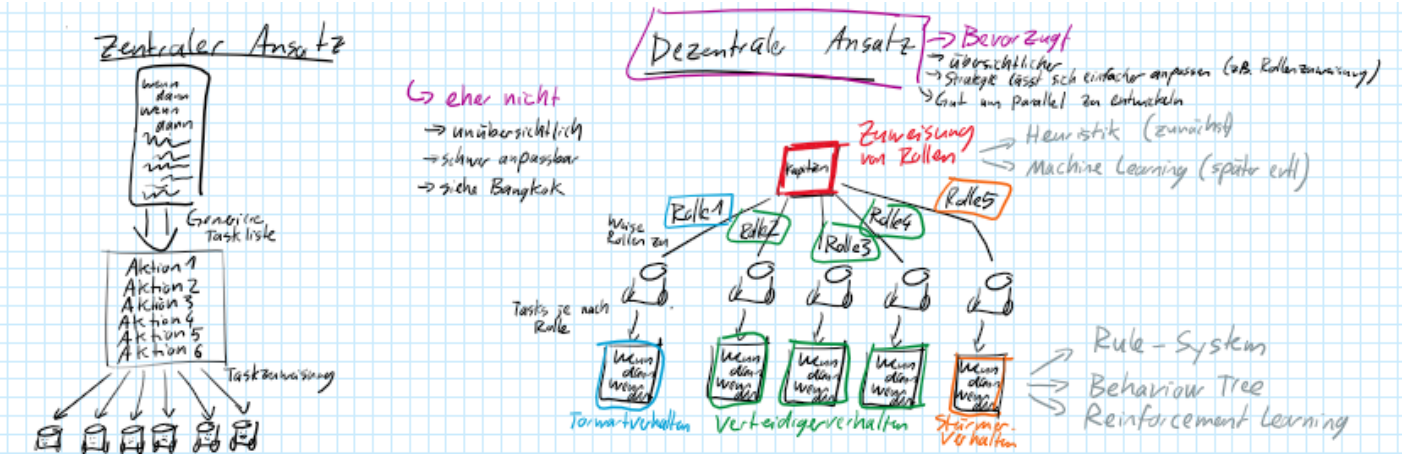
Trotz der beschriebenen Problematiken war die Strategie in Bangkok in der Lage den 3. Platz in der SSL Small Size League Division B zu erreichen. Jedoch haben sich in der Entwicklung und im Wettbewerb die oben beschriebenen Probleme gezeigt, weshalb der Ansatz als nicht zielführend für die folgenden Roboter eingeordnet wurde. Stattdessen wurden aus den Erfahrungen Anforderungen für ein neues System abgeleitet:

- *Entscheidungsfindung sollte in mehrere Schritte aufgeteilt werden*: Da Taskswitching ein großes Problem war, sollte zunächst eine robuste Zuordnung der Roboter zu Offensive/Defensive getroffen werden. Erst im Anschluss sollten die Tasks basierend auf dieser Zuordnung zugewiesen werden.
- *Kapselung der Module*: Um die Entwicklung zu vereinfachen und zu parallelisieren sollten Module unabhängig voneinander bearbeitet werden können.
- *Optimierung der Zuweisung*: Zunächst sollten alle Tasks und deren Zielpunkte ermittelt und erst im Anschluss die Roboter zugewiesen werden. Dadurch kann eine Zuweisung gefunden werden, die die Fahrwege minimiert und Kooperation der Roboter verbessert. Dadurch wird auch die starke Prioritätsabhängigkeit des Rule-Systems vermindert.
- *Aktiveres Spiel*: Der Rein reaktive Spielstil ist nicht ausreichend, um Pässe effektiv zu planen und die gegnerische Defensive zu umspielen. Hier sind Analysen und Planungen notwendig, statt einer reinen Reaktion auf den Ist-Zustand.

Bangkok 2022

Folgen und weitere Entwicklungen

Aus den genannten Anforderungen wurde eine neue Architektur abgeleitet, die die erkannten Probleme beheben soll. Diese werden im nachfolgenden Bild dargestellt. Links ist dabei das Rule-System bei dem anhand einer umfangreichen Taskliste für jeden Roboter ein Task bestimmt wird. Der Ansatz rechts zeigt einen Vorgang, bei dem zunächst Roboter zu verschiedenen Rollen zugewiesen werden, wie z.B. Angreifer, Torwart, Verteidiger. Im Anschluss werden diese von einem kleineren Rule-System oder ähnlichem gesteuert, welches jedoch übersichtlicher ist, da es sich auf das Verhalten einer einzigen Rolle begrenzt. Außerdem sind dabei alle Roboter bekannt, die ebenfalls dieselbe Rolle zugewiesen bekommen haben, sodass die Roboter auch gemeinsam kooperieren können.



Bordeaux 2023

Der für Bordeaux entwickelte Taskmanager stellt die zweite Iteration dar. Anstatt eines einzigen Rule-Systems werden hier mehrere unterschiedliche Module verwendet. Zunächst wird vom Rolemanager jedem Roboter eine Rolle zugewiesen. Für jede Rolle existieren einzelne Verhaltensweisen, wie beispielsweise die Flexwall. Außerdem wurden zur Gestaltung der Offensive auch erstmals MinMax-Bäume eingesetzt, um Pässe zu planen.

Übersicht

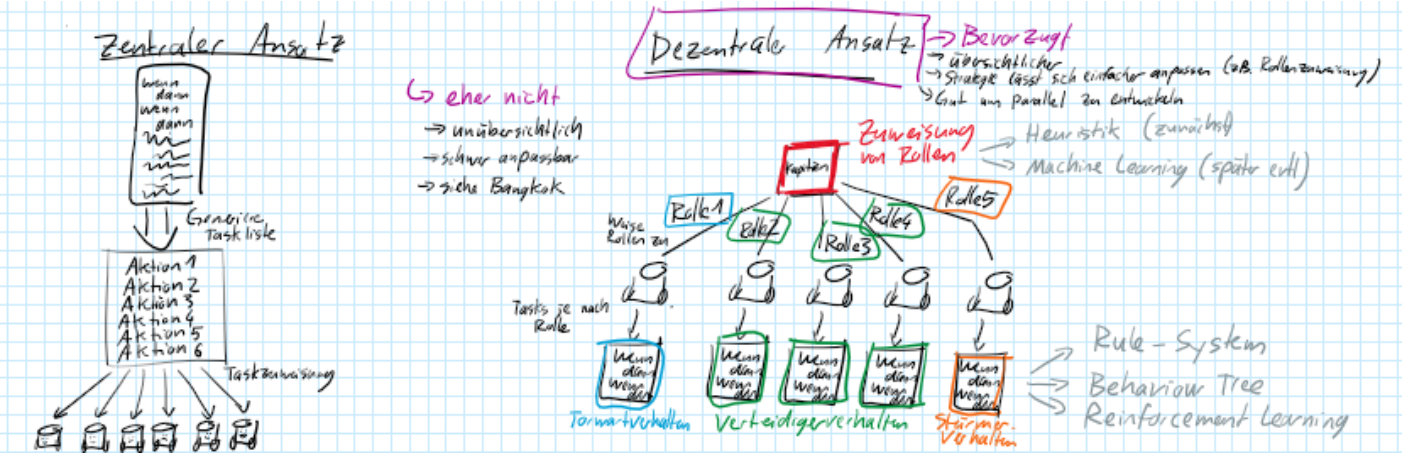
Der Bordeaux-Taskmanagers stellt die zweite Iteration der Strategieentwicklung dar. Der Taskmanager Bangkok wurde aufgrund seiner Probleme hinsichtlich Komplexität und schlechter Erweiterbarkeit nach dem Wettbewerb komplett verworfen. Stattdessen wurden aus den Erfahrungen die Anforderungen für die Entwicklung eines neuen Systems abgeleitet, welches sich vor allem auf Erweiterbarkeit sowie Einfachheit in der Entwicklung fokussiert.

Grundprinzip und Architektur

Unter Taskmanager Bangkok wurde der ursprüngliche Ansatz beschrieben, die Strategie über ein einziges *Rule-System* zu realisieren. Aus den dort beschriebenen Problemen wurden Anforderungen für eine neue Architektur abgeleitet, die die Entwicklung, Anpassbarkeit und Erweiterung des Systems vereinfachen sollen. Die Ziele lassen sich folgendermaßen klassifizieren:

- *Mehrschrittiger Entscheidungsprozess*; Hiermit soll eine bessere Kapselung der Module erreicht werden, wodurch diese unabhängig voneinander entwickelt werden können. Außerdem soll nach dem "Teile und Herrsche"-Prinzip das große Problem der Strategie in mehrere Teilprobleme aufgeteilt werden, um die Lösbarkeit des Problems zu vereinfachen. Der wesentliche Ablauf der Task-Zuweisung lässt sich folgendermaßen beschreiben:
  1. Analyse des Spielfeldes
  2. Festlegung der Rollenverteilung (Anzahl Angreifer/Verteidiger...)
  3. Zuweisung der Rollen je nach gewünschter Rollenverteilung und Roboterposition
  4. Festlegung der auszuführenden Tasks in Abhängigkeit der Rollenverteilung
  5. Zuweisung der Tasks zu den verfügbaren Robotern in Abhängigkeit der Rollenverteilung
- *Kapselung der Module*; Roboter die an derselben Aufgabe arbeiten sollten besser kooperieren, während Roboter die andere Aufgaben ausführen ignoriert und nicht in die Entscheidungsfindung einbezogen werden sollten. Dies vereinfacht einerseits die Entwicklung, da nicht stets das Gesamtsystem betrachtet werden muss, um Änderungen an kleineren Teilaufgaben vorzunehmen und verbessert gleichzeitig die Kooperation, da Roboter innerhalb einer Gruppe sich besser abstimmen können. Die Gruppen werden dabei durch den Role Manager definiert.
- *Optimierung der Zuweisung*; Damit die Roboter als Team spielen, ist es notwendig dass sie gemeinsam spielen und ihre Mitspieler bei den Entscheidungen berücksichtigen. Da prinzipiell alle Roboter hardwaretechnisch gleich aufgebaut sind, kann jeder Roboter jeden Task gleich gut ausführen. Dies ermöglicht auch eine optimierte Zuweisung, bei der zunächst die auszuführenden Tasks ermittelt werden und die tatsächliche Zuweisung dieser zu den Robotern erst im nachfolgenden Schritt erfolgt. Dadurch kann der Gesamtweg des Teams minimiert werden, was in kürzeren Fahrtwegen und somit einer höheren Spieldynamik resultiert.

Die grobe Architekturänderung ist in der nachfolgenden Abbildung gezeigt. Links befindet sich das zuvor genutzte Rule-System, bei der ein komplexes Rule-System die Aufgaben für alle Roboter generiert. Rechts ist der für Bordeaux 23 dargestellte Ansatz, bei der die Roboter Rollen erhalten. Roboter mit derselben Rolle befinden sich in derselben Gruppe und arbeiten gemeinsam an einer Aufgabe, wie der Verteidigung des Strafraums.



Eine weitere Wesentliche Neuerung ist, dass die *Rule-Systeme* zur Steuerung der einzelenen Rollen durch *Zustandsautomaten* ersetzte wurden. Ein wesentlicher Vorteil dieses Schrittes besteht darin, dass einerseits besser kontrolliert werden kann, welche Zustände in welche anderen Zustände übergehen dürfen und andererseits, dass *Hysteresen* verwendet werden können. Diese Schritte stabilisieren das System und verhindern so den ständigen Wechsel von Tasks, falls ein instabiler Zustand (zb. durch Gegnerbewegungen/Rauschen/schlechter Datenqualität/...) eintreten kann, bei denen die Roboter sehr schnell zwischen Zuständen wechseln und daher letztlich keine der Aktionen ausführen.

> **Rule-System**:
> Ein Rule-System ist eine Architektur zur Steuerung von Systemen. Hierbei werden verschiedene Regeln definiert, die an Aktionen gekoppelt sind. Jede Rule besitzt zudem eine Priorität, wobei die Regeln in absteigender Priorität geprüft werden. Ist die Bedingung einer Regel erfüllt, so wird die zugehörige Aktion ausgeführt. Weitere Infos: Wikipedia, Complexica.

> **Zustandsautomat**:
> Ein Zustandsautomat (engl. State Machine) ist eine Architektur zur Steuerung von Systemen. Dabei werden verschiedene Zustände sowie Transitionen zwischen diesen definiert. Das System verbleibt in einem Zustand, bis die Bedingung an einer dort entspringenden Transition erfüllt ist. Jedem Zustand können Aktionen zugeordnet werden, um so ein Verhalten zu steuern. Weitere Infos: Wikipedia, Medium.

> **Hysterese**:
> Eine Hysterese beschreibt ein Prinzip, mit dem sich Zustandsänderungen kontrollieren und stabilieren lassen können. Beispiel: Wird ein Zustand bei

> gewechselt, wenn x > 2 ist, soll dieser erst wieder verlassen werden wenn x < 1 ist. Flackert der Wert zwischen 1,8 und 2,2 so bleibt der Zustand stabil. Weitere Infos: Max Planck Gesellschaft: Hysterese.

Verwendete Rollen und deren Aufgaben

Die genauere Zuweisung der Rollen wird in Role Manager Bordeaux 2023 beschrieben. Es stehen die folgenden Rollen zur Verfügung, deren Details in eigenen Wiki-Einträgen angesehen werden können:

- *Torwart*: Soll das Tor Verteidigen und Ball im Strafraum manipulieren.
- *Verteidigung*: Soll direkte Schüsse auf das Tor verhindern indem am Strafraum gemauert wird. Die Tasks werden hierbei gemäß des Flexwall-Algorithmus bestimmt und zugewiesen.
- *Support*: Soll Pässe verhindern, indem gefährliche Gegner gedeckt werden. Die Funktionsweise des Algorithmus findet sich hier.
- *Offensive*: Soll den Ball erobern, sich für Pässe anbieten sowie Torschüsse ermöglichen. Dies wird ausführlicher hier beschrieben.

Bordeaux 2023

Support

Der Support stellt die aktive Verteidigung dar. Der Support wurde für Bordeaux im Rahmen des Taskmanager_Bod23 entwickelt und dort erstmalig eingesetzt. Das Hauptziel des Supports besteht in der Manndeckung, durch die Pässe verhindert und abgefangen werden sollen. Neben dem Support besteht die Verteidigung auch noch aus der Flexwall, die den Strafraum vor direkten Torschüssen schützt, falls ein Pass nicht abgefangen werden kann.

> Da sich die Strategie schnell und stetig ändert, ist dieser Artikel als ein Einstieg in die Thematik zu verstehen und kann keinen Anspruch auf Aktualität erheben. Hierzu ist der tatsächliche Code heranzuziehen.

Grundprinzip

Das Ziel ist die Manndeckung, bei der verhindert werden soll, dass Pässe zu gefährlichen Gegnern gespielt werden können. Hierzu positionieren sich die Roboter nahe des zu deckenden Spielers auf die Sichtlinie vom Ball zu diesem Spieler.
Um die Entscheidung zu treffen, welche Gegner gedeckt werden sollen, müssen zunächst die Gefahrenlevel (=Threat) der Gegner eingeschätzt werden. Hierzu stellt der Observer Funktionen zur Verfügung, die im Allgmeinen auf der Nähe des Roboters zum Tor und zum Ball basieren. Ausnahmen stellen der Torwart und der Ballführende Spieler dar, die nicht berücksichtigt werden sollten, da eine Deckung dieser im Allgemeinen nicht sinnvoll ist.

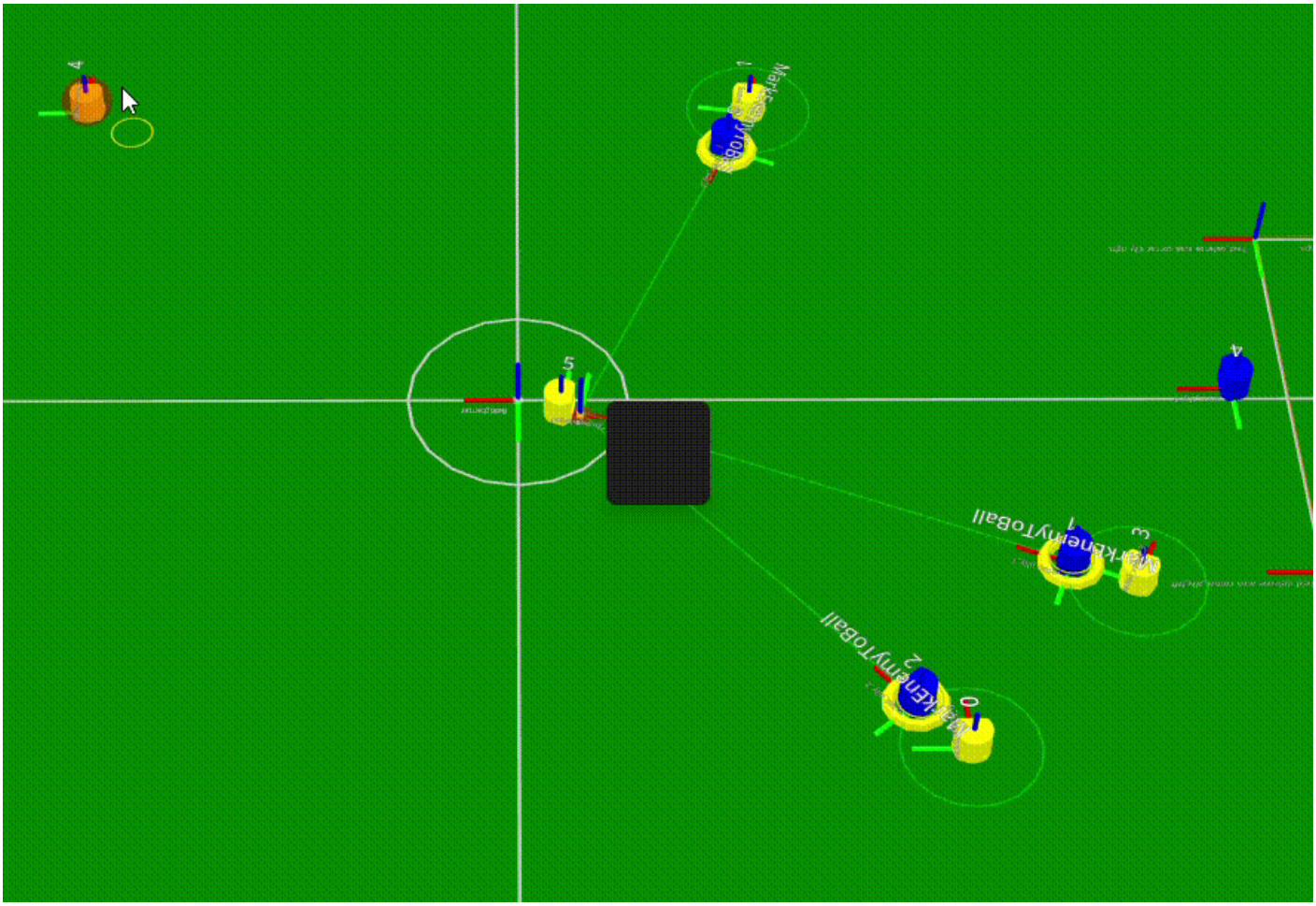
1. Die verbleibenden Gegner werden nach absteigendem Threat geordnet. Im Allgmeinen kann davon ausgegangen werden, dass die Anzahl der Supporter geringer als die der Gegner ist, weshalb nicht alle Gegner gedeckt werden können. Stattdessen werden nur die Gegner mit hohem Threat betrachtet, der Rest verworfen.
2. Es wird für jeden zu verteidigenden Gegner die Zielposition geschätzt; Diese liegt etwa bei dem Gegner, mit einem Offset in Richtung Ball. Dieser Schritt sollte in zukünftigen Softwareversionen direkt von der Pfadplanung übernommen werden.
3. Es wird eine Kostenmatrix aufgestellt, die die Distanzen von jedem Supporter zu jeder Zielposition beinhaltet. Diese ist die Basis für den Munkres Algorithmus, mit dem eine ideale Zuordnung der Roboter zu den Zielen gefunden wird. Hier hat sich als Kostenmaß die euklidische Distanz bewährt, da Roboter im Allgemeinen versuchen, bei ihrem aktuellen Ziel zu verbleiben, auch wenn sich die zu deckenden Gegner ändern.

4. Die Roboter werden gemäß der Ergebnisse des Munkres-Algorithmus auf die verschiedenen Gegner verteilt.

Durch diesen Vorgang werden die *n* gefährlichsten Gegner von *n* Supportern gedeckt. Die Deckung ist dabei jedoch nur als eine Vorpositionierung zu verstehen. Sollte der Gegner dennoch einen Pass-Versuch unternehmen, so wird dieser oft in den Lauf erfolgen, wodurch an dem Supporter vorbeigespielt wird. Daher sollten die Supporter Intercept-Mannöver durchführen, sobald ein Schuss in Richtung des Roboters detektiert wird, um den Ball aktiv abzufangen.

Support in der Praxis

Im folgenden Video wird die Funktion des Supports in einem Simulationsszenario demonstriert:



Erfahrungen und Kritik

- Das Verfahren hat sich in der Vergangenheit allgemein als sinnvoll erwiesen. Das Abfangen von Pässen ist absolut kritisch, da so Torschüsse frühzeitig verhindert werden.

Kann der Gegner ungestört passen, so gibt ihm das die Freiheit Lücken in der Verteidigung zu erspielen und auszunutzen. Der Support behindert dabei die Freiheiten des Gegners.
- Da der Suppport im Spielfeld und meist fern vom Strafraum agiert, können die Spieler auch schnell in die Offensive wechseln, wenn ein Ball erfolgreich abgefangen wird, um so einen Konter durchzuführen.
- Der Support versucht stets die aktuelle Position des Gegners zu decken. Dies berücksichtigt dessen Fahrt nicht und führt dazu, dass wir leicht hinterherhinken.
- Wenn sich Gegner "verklumpen", so führt dies dazu, dass ggfs. einige Support versuchen die Gegner zu decken und dabei die Klumpenbildung verstärken. Dies ist nicht sinnvoll, da sich die Roboter dabei gegenseitig behindern und dies keinen signifikanten Mehrwert hat.
- Es hat sich als sinnvoll erwiesen, neben Torwart und Ballführer auch die gegnerischen Verteidiger auszuschließen, bzw. die Threat-Berechnung so zu gestalten, dass diese nicht berücksichtigt werden. Hier sollten alle Spieler ausgeschlossen werden, die sich im oder um den gegnerischen Strafraum befinden.

Weiterentwicklungen

Aktuell bleibt das Konzept für Eindhoven unverändert.

Ideen für zukünftige Weiterentwicklungen

- *Berücksichtigung der Gegnergeschwindigkeit*: Es sollte abgeschätzt werden, wo der Gegner in den nächsten x ms sein wird und diese Position gedeckt werden. Dies sollte das Abfangen von Pässen in den Lauf verbessern.
- *Verbesserung des Threat-Levels*: Gegner die sehr nah am Ball sind, müssen ggfs. nicht gedeckt werden, da sie keine große zusätzliche Gefahr darstellen.
- *Verbesserung des Intercept-Verhaltens*: Der ideale Intercept Punkt sollte sinnvoller bestimmt werden, sodass der Roboter schnellstmöglich den Ball erhält. Außerdem wäre es sinnvoll, den Punkt so zu wählen, dass er weit weg von Gegnern ist, damit diese den folgenden Konter nicht so einfach behindern können.
- *Clusterverteidigung*: ähnlich wie beim Flexwall-Algorithmus können ggfs. Gegner zusammengefasst und gemeinsam verteidigt werden. Dies reduziert die Anzahl an notwendigen Support-Spielern.

Bordeaux 2023

Offensive

Die Offensive ist Teil des Taskmanager. In diesem Artikel wird vor allem auf den in im Rahmen des Taskmanager_Bod23 entwickelten Algorithmus eingegangen und das zugrundeliegende Konzept vorgestellt. Außerdem werden die Erfahrungen sowie Ideen zur Weiterentwicklung dokumentiert.

> Da sich die Strategie schnell und stetig ändert, ist dieser Artikel als ein Einstieg in die Thematik zu verstehen und kann keinen Anspruch auf Aktualität erheben. Hierzu ist der tatsächliche Code heranzuziehen.

Die Positionierung von offensiven Spielern war das Kernthema unseres eTDP 2024 (siehe Anhang links).

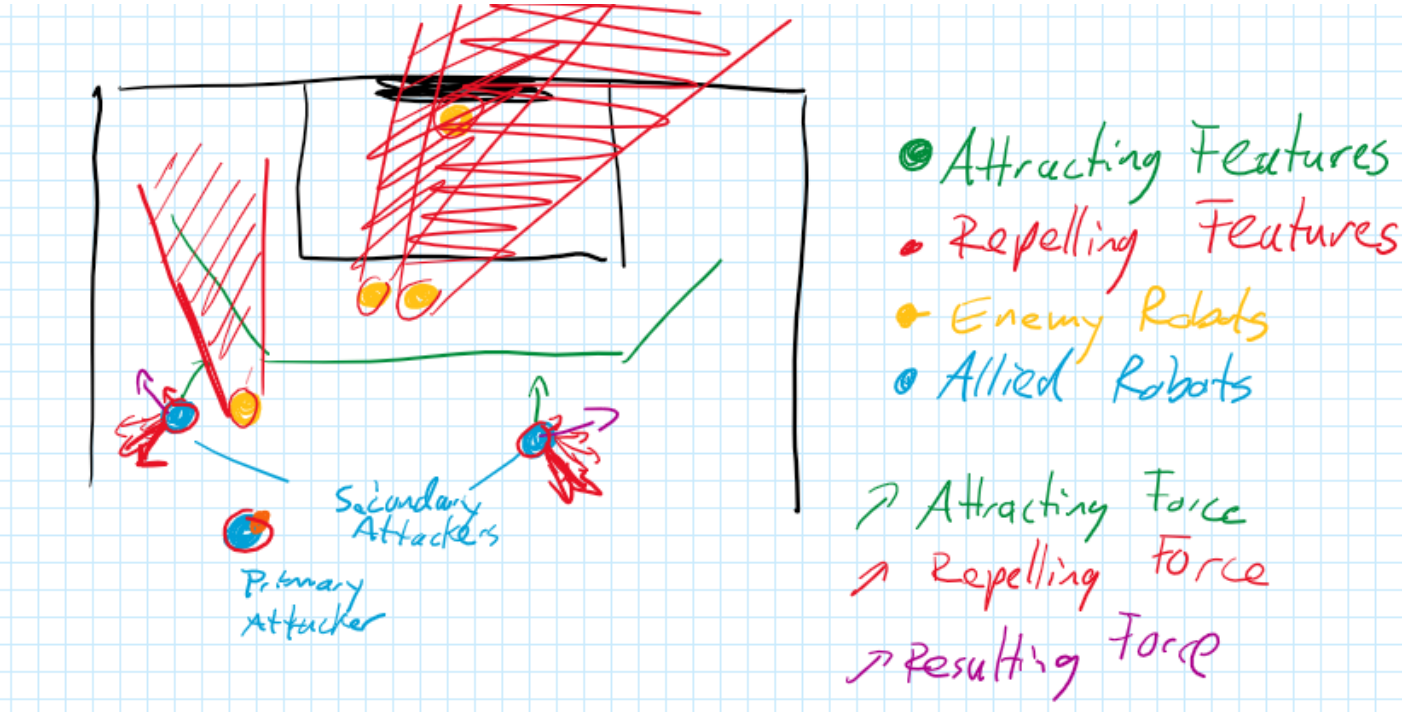
Motivation und Hintergrund

Das Ziel der Offensive besteht vor allem in der Eroberung und Kontrolle des Balles, sowie den Vorbereitungen für einen erfolgreichen Torschuss. Da stets davon ausgegangen werden kann, dass der Gegner versuchen wird, den eigenen Torschuss zu verhindern, müssen die Bewegungen und Möglichkeiten der gegnerischen Roboter zu jedem Zeitpunkt berücksichtigt werden.
Anders als die Verteidigung kann eine Offensive nicht rein reaktiv funktionieren. Eine rein reaktive Offensive kann schon an einer sehr simplen Verteidigung scheitern. Die aktive Vorausplanung erfordert jedoch deutlich komplexere und umfangreiche Algorithmen, bei denen die Verteidiger aktiv berücksichtigt und umgangen werden. Hier können Konzepte aus der Spieltheorie angewandt und herangezogen werden, wie beispielsweise Spielbäume, auf die im Weiteren auch eingegangen werden wird.

Offensive Bangkok 2022

Die erste Iteration der Offensive wurde im Rahmen des Taskmanager Bangkok 2022 entwickelt und getestet. Damals wurde ein zentrales Rule-System für die gesamte Strategie verwendet, was den Code schwer anpassbar gemacht hat und sich auch unmittelbar auf die Verteidigung etc. ausgewirkt hat.
In diesem Rule-System waren verschiedene Offensive Rules, die nach ihrer Priorität abgearbeitet wurden:

1. Torschuss: Wenn ein Roboter den Ball hat und die Torwahrscheinlichkeit hoch ist, soll ein Torschuss unternommen werden
2. Passen: Wenn kein Torschuss ist und ein Pass durchgeführt werden kann, soll ein Pass ausgeführt werden
3. Mit Ball fahren: Sind weder Torschuss noch Pass möglich, soll der Roboter mit Ball fahren
4. Emergency Shot: Sind weder Torschuss noch Pass noch Dribbling möglich (z.B. weil die maximale Fahrdistanz mit Ball erreicht wurde), wird der Ball weggeschossen
5. Get Ball: Ist unser Team aktuell nicht im Ballbesitz, soll der nächste Roboter den Ball holen/stehlen
6. Offensive Movement: Roboter (die nicht den Ball haben) sollen sich freilaufen, wobei sie von dem gegnerischen Strafraum angezogen und von Gegnern sowie deren "Schatten" ausgehend vom Ball abgestoßen werden. Dies wird in der nachfolgenden Abbildung gezeigt:

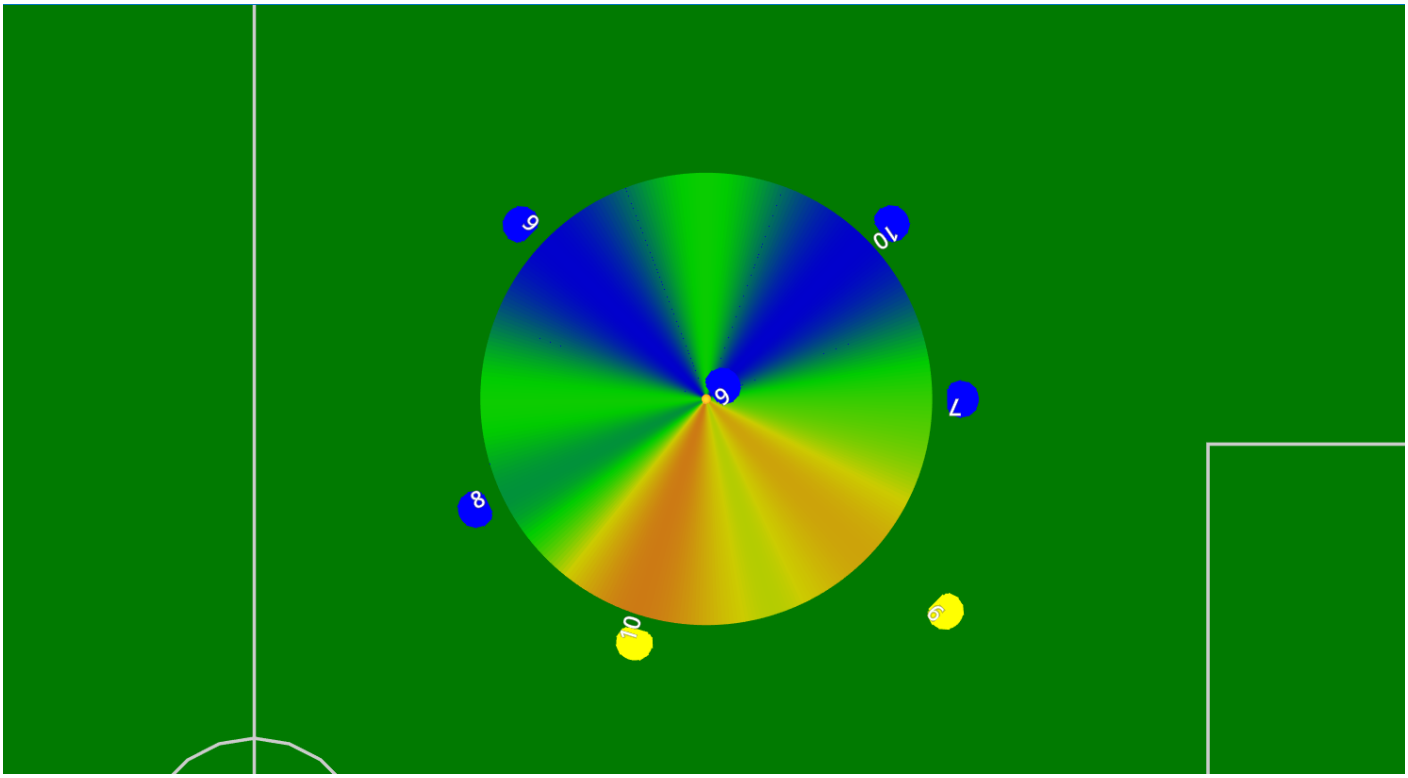


Dieses System stellt eine reine Reaktion auf den Ist-Zustand dar. Gegner werden zwar in der Entscheidung ob ein Schuss durchgeführt werden soll berücksichtigt, ebenso bei der Positionierung im Offensiven Movement. Allerdings sind dies passive Entscheidungen, die das Verhalten und die Möglichkeiten der Gegnern nicht direkt berücksichtigen und um diese herumplanen. Außerdem hat sich gezeigt, dass das Offensive Movement oftmals in nicht idealen Positionen resultiert hat, da die Roboter zu weit weg waren, wenn der Ball in unserer eigenen Hälfte war, außerdem fahren die Roboter zwar von Gegnern weg, jedoch nicht zwingend auf eine Weise, die sie besser anspielbar macht.

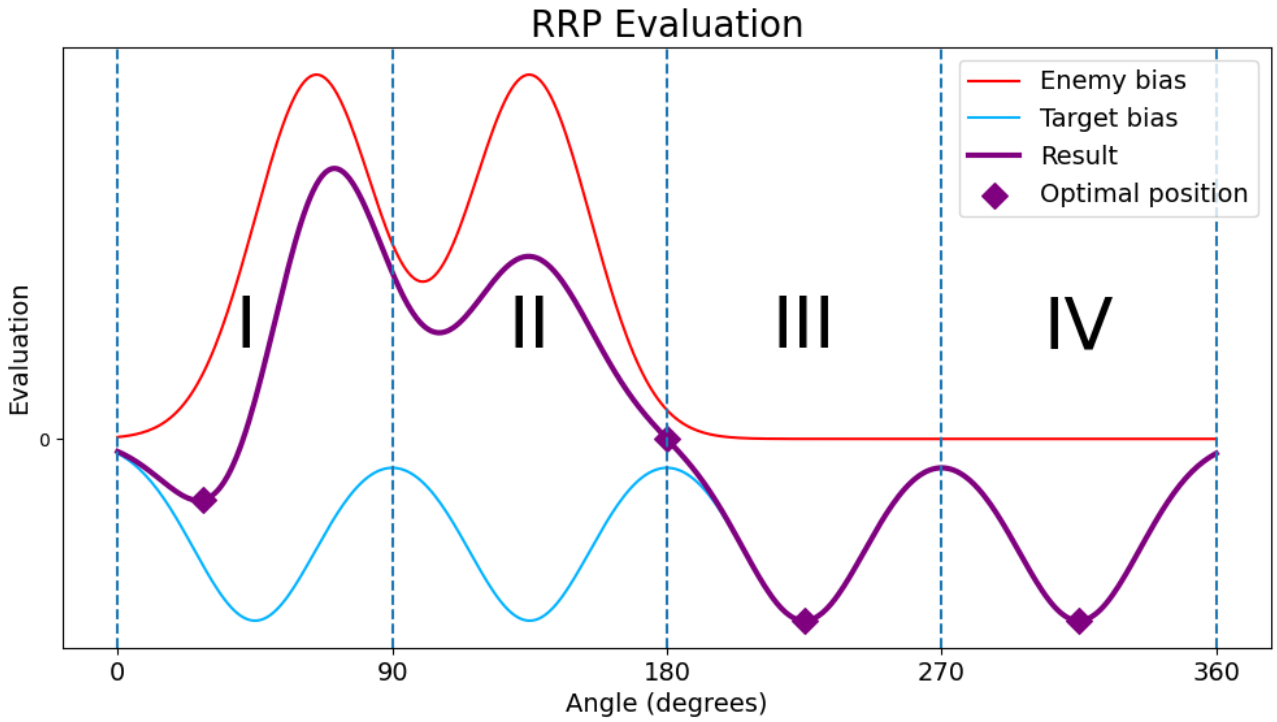
Offensive Crailsheim 2023

Vor dem RoboCup in Bangkok wurde ein Testtunier in Crailsheim gespielt, auf dem erstmalig die zweite Iteration der Offensive getestet wurde. Das wesentliche Rule-System aus Bangkok wurde

von der Logik übernommen, allerdings mit dem Rollensystem, welches durch den Rolemanager_bod23 vorgegeben wird. Dadurch war der Code einfacher und verständlicher, da die Offensive von der Defensive entkoppelt worden konnte. Außerdem wurde auch das offensive Movement überarbeitet.
Anstatt vom Strafraum angezogen zu werden und dabei ggfs. zu weit vom Ball wegzufahren, wurde der Abstand der Roboter auf einen vorgegebenen Abstand festgesetzt. Offensive Spieler konnten sich somit nur noch auf einer Kreisbahn mit etwa zwei Meter Radius um den Ball herum bewegen. Um Gegnern auszuweichen und Soll-Positionen zu definieren, wurden dabei Punkte definiert, von denen die Roboter angezogen bzw. abgestoßen werden.



Das Prinzip folgt einer Optimasuche auf einem Graphen. Der Graph beschreibt dabei die Güte einer Position bezogen auf den Winkel relativ zum Ball. In dieser Überlegung werden Gegner durch eine positive Gausskurve repräsentiert, wodurch im Graphen ein "Berg" an der Gegnerposition entsteht. Die Sollpositionen werden durch negative Gausskurven repräsentiert, die zu "Tälern" führen. Das Ziel der offensiven Robotern ist es, zum Minimum in dem ihnen zugewiesenen Quadranten zu fahren. Dies soll verhindern, dass mehrere Roboter dieselbe Position anfahren sollen. Welche Quadranten verwendet werden sollen, hängt von der Ballposition ab. im Allgemeinen werden dabei Quadranten besetzt, die nahe der Spielfeldmitte sind.



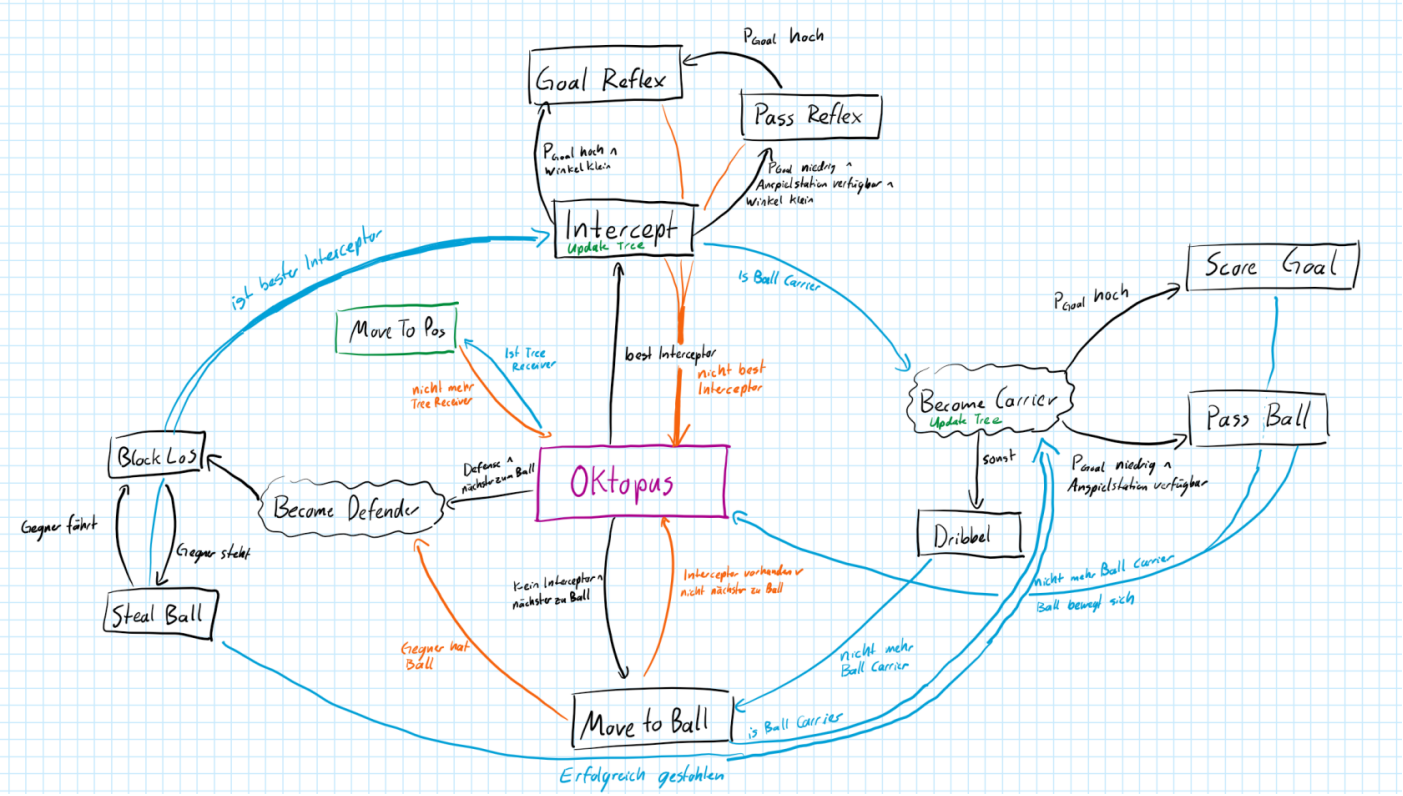
Die Spiele in Crailsheim haben gezeigt, dass es mit diesem Ansatz möglich ist, die Entfernung vom Ball zu stark zu limitieren, dass die Roboter stets anspielbar sind. Außerdem fahren die Roboter nicht nur vor Gegnern weg, sondern weichen dabei auch in eine Richtung aus, die die Anspielbarkeit erhöht.
Auch wenn dieser Ansatz eine Verbesserung darstellt, ist es trotzdem keine hinreichend gute Lösung. Das Hauptproblem dieses Lösungsansatzes ist, dass das System durch die Begrenzung auf den Kreis sehr starr ist. Desweiteren weichen die Roboter zwar Gegnern aus, versuchen dabei aber nicht aktiv eine Gefahr darzustellen. Selbst wenn ein Pass erfolgreich ausgeführt wird, ist dies keine Garantie, dass dadurch ein nennenswerter Fortschritt bezüglich der Torwahrscheinlichkeit entsteht. Die Folge dessen ist der Verwurf dieses Ansatz zu Gunsten einer komplexeren, aktiven Offensive.

Aktive Offensive: Bordeaux 2023

Wie aus dem vorherigen Abschnitt hervorgeht, ist der frühere, passive Ansatz nicht in der Lage, eine wirkliche Bedrohung für den Gegner darzustellen. Selbst schwache Defensiven können nicht aktiv umgangen werden. Daher wurde vor dem RoboCup in Bordeaux ein besonderer Fokus auf die Entwicklung einer aktiven Offensive gesetzt, deren Konzepte im folgenden Erläutert werden.

Grundstruktur

Eine wesentliche Änderung war der in Taskmanager_Bod23 beschriebene Wechsel von einem Rule-System zu einem Zustandsautomaten. Dieser Zustandsautomat steuert alle Spieler mit der offensiven Rolle, wobei die verschiedenen Roboter in unterschiedlichen Zuständen sein können. Generell ist stets ein Zustand aktiv, der sich um die Ballmanipulation (Ball holen, Ball abfangen, schießen, dribbeln...) kümmert und nur von genau einem Roboter ausgeführt werden darf. Diese Beschränkung ist wichtig, da sich Roboter ansonsten gegenseitig den Ball wegnehmen oder behindern würden. Die restlichen Roboter werden verwendet, um sich freizulaufen und für Pässe anzubieten. Der grobe Aufbau des Zustandsautomaten befindet sich in der folgenden Grafik, wobei für die Einfachheit nicht alle Transitionen und Bedingungen eingezeichnet wurden. Die verschiedenen Zustände werden im Folgenden genauer erläutert.



Oktopus

Dies ist der Standardzustand. Roboter, die noch keinem Zustand zugewiesen werden starten hier. Bei Oktopus handelt es sich um einen Algorithmus, bei dem sich die Roboter freilaufen, um sich so als potenzielle Anspielstation anzubieten. Sobald der Roboter sich in einer guten Position befindet und dadurch in eine Passkette eingeplant wird, wechselt er zu *MoveToPos*. Ist das nicht der Fall, sollte der Roboter durch Oktopus weiterhin versuchen, die eigene Position zu verbessern, bis er als Anspielstation herangezogen wird. Die Funktionsweise des Algorithmus wird in einem eigenen Artikel zu Oktopus erläutert.

Primary Attacker

Wie bereits erwähnt, sollte stets genau ein Spieler mit der Ballführung beauftragt werden, dieser wird im Folgenden als Primary Attacker bezeichnet. Dies ist im Allgemeinen der Roboter, welcher am nächsten zum Ball ist. Hat dieser aktuell noch nicht den Ball in der Lichtschranke, sollte er *MoveToBall* bzw. *StealBall* Ausführen, je nachdem ob grade ein Gegner bereits den Ball hat oder nicht. Wenn sich der Gegner bewegt, sollte statt *StealBall* eher *BlockLoS* (LoS = LineOfSight) ausgeführt werden, da in diesem Fall das Stehlen des Balles schwierig ist und der Gegner so an einem Pass oder Schuss gehindert werden kann. Sobald der Gegner mit dem Ball zum Stillstand kommt, kann ein Balldiebstahl versucht werden.
Diese Fälle sind sinnvoll für einen sich nicht oder nur langsam bewegenden Ball. Wurde der Ball soeben geschossen, sollte stattdessen versucht werden, dass der Primary Attacker diesen Ball abfängt. Da sich der Ball schnell bewegt, sollte auch nicht der nächste Roboter als Primary Attacker verwendet werden, sondern stattdessen der Roboter, der sich am nächsten zur Balltrajektorie befindet, was durch den *Best Interceptor* aus dem observer vorgegeben wird. Anstatt eines Intercept-Manövers kann auch direkt ein Reflexschuss durchgeführt werden, bei dem der Ball nicht angenommen, sondern direkt weitergeschossen wird, sofern der Winkel dabei klein genug ist und es ein sinnvolles Ziel gibt. Für das Ziel kann einerseits das Gegnertor verwendet werden, sofern die Torwahrscheinlichkeit über dem von Adathresh festgelegtem Schwellwert liegt. Gleichermaßen kann auch ein Reflexpass ausgeführt werden, wobei diese Funktion während des Wettkampfes in Bordeaux aufgrund der Anfälligkeit für Software-Bugs entfernt werden musste.
Pässe und Torschüsse können auch normal ausgeführt werden, wenn der Roboter im Ballbesitz ist. Dabei hat ein Torschuss stets Priorität. Ob ein Torschuss ausgeführt werden sollte, hängt wie schon beim Reflexschuss davon ab, ob die aktuelle Torwahrscheinlichkeit über dem von Adathresh festgelegten Schwellwert liegt. Ist dies der Fall, wird ein Schuss auf den besten Torpunkt ausgeführt, dessen Berechnung ebenso wie die der Torwahrscheinlichkeit hier eingesehen werden kann.
Ist ein Torschuss nicht möglich, wird stattdessen ein Passbaum aufgebaut, der verschiedene Passketten evaluiert. Der Passbaum beinhaltet mögliche Pässe zu jedem anderen offensiven Mitspieler, wobei auch Pässe in den Lauf berücksichtigt werden. Die Pässe werden hinsichtlich der Wahrscheinlichkeit, dass der Ball nicht von einem Gegner abgefangen wird, sowie der Torwahrscheinlichkeit des Empfängers nach Erhalt des Balles evaluiert. Dabei wird auch berücksichtigt, dass der Ball vom Empfänger zu einem weiteren Roboter gepasst werden könnte, wodurch Passketten, Doppelpässe etc. entstehen. Aus dem Baum wird dann die Passkette ausgesucht und gespielt, die einerseits die geringste Abfangwahrscheinlichkeit durch Gegner zulässt und andererseits die resultierende Erfolgschance auf ein Tor maximiert.
Sollten alle Mitspieler gedeckt oder aus anderen Gründen nicht verfühgbar sein, wird als Notlösung ein Dribbling durchgeführt. Das Dribbling besteht dabei aus schwachen Schüssen in Richtung des Gegnerischen Tores, wobei hierbei auch Gegnern ausgewichen wird. Durch das Dribbling sollen die Mitspieler Zeit gewinnen, um sich freilaufen und als Anspielstation anbieten zu können und gleichzeitig ein Fortschritt mit dem Ball zum gegnerischen Tor erzielt werden.

MoveToPos

Dieser Zustand wird von offensiven Spielern eingenommen, die in der aktuellen Passplanung durch den Passbaum als Anspielstation eingeplant sind. In diesem Zustand fahren die Roboter an die Position, an der später der Ball angenommen werden soll. Wird der Roboter in der aktuellen Passplanung nicht berücksichtigt oder ändern sich die Passpläne, sodass er nicht länger berücksichtigt wird, so wechselt der Roboter in den Zustand Oktopus, um sich solange weiter frei zu laufen, bis er erneut in eine Passplanung miteinbezogen wird.

Erfahrungen und Evaluation

Die Offensive wurde erstmalig in Bordeaux 2023 getestet und hat sich schnell als eine deutliche Verbesserung zum vorherigen Stand aus Crailsheim herausgestellt. Durch den Aufbau der Passketten in den Passbäumen war es unserem Team möglich, sehr effizient Pässe zu spielen und so um die gegnerische Verteidigung herumzuspielen. Während die Crailsheimer Offensive zwar in der Lage war, Namec mit stillstehenden Robotern innerhalb von knapp 10 Minuten mit 10:0 zu besiegen, konnte die Bordeaux-Offensive dies in der Hälfte der Zeit schaffen und auch gegen Teams, die leichten Widerstand geleistet haben, indem beispielsweise Mauern am Strafraum gebildet wurden. Das Verfahren an sich hat sich somit als sinnvoll erwiesen, trotzdem sich auch Schwachstellen und Verbesserungsmöglichkeiten aufgefallen, die im folgenden aufgelistet sind:

- *schlechte Dynamik*: Die Roboter benötigen im Allgemeinen zu lange, um sich mit dem Ball zu drehen. Dies ist in erster Linie ein Problem der Hardware und des Plannings, sollte aber in der Planung berücksichtigt werden, da Passketten sonst zu früh unterbrochen werden.
- *schwaches Zweikampfverhalten*: Gegen aggressive Roboter ist es uns allgemein schwer gefallen, Pässe zu spielen und Fortschritt zu erreichen. Bereits ein einzelner Roboter, kann durch eine LineOfSight-Verteidigung sämtliche Passversuche unterbinden. An dieser Stelle wäre einerseits ein Lupfer sinnvoll, um über die Gegner rüber schießen zu können, alternativ wären auch Konzepte denkbar, bei denen andere Offensive Spieler die Verteidiger blockieren, sodass der Primary Attacker einen Schuss ausführen kann. Das Zweikampfverhalten muss jedoch auch in der Defensive verbessert werden; Selbst wenn es uns gelingt, dem Gegner den Ball abzunehmen, sind meist keine sinnvollen Möglichkeiten vorhanden, bevor der Gegner selbst in den Zweikampf geht und den Ball zurückerobert. Auch hier wären Lupfer und unterstützenden offensive Spieler von Vorteil.
- *Lange Rechenzeiten*: Durch die Vielzahl an Möglichkeiten, mit denen Pässe potentiell durchgeführt werden könnten, ist die damit verbundene Rechenzeit sehr hoch. Dies wurde zwar auf unter eine Sekunde reduziert, jedoch auf Kosten der Suchtiefe und -breite. Hier wäre eine effizientere Berechnung sowie evtl. eine Auslagerung der Berechnung auf die Grafikkarte sinnvoll. Andere Teams wie Zjunlict wenden dies bereits an und beschreiben den Vorgang in einem ihrer ETDPs: ZJUNlict 2020.
- *schwache Ballannahme*: Bei den Pässen ist der Ball oft vom Roboter abgeprallt. Dies ist einerseits auf die hardwareseitige Dämpfung zurückzuführen, andererseits sollten jedoch auch die Skills angepasst werden, sodass der Roboter z.B. durch eine bessere Einstellung der Dribbling-rolle und ggfs. durch eine Fahrt zurück den Ball besser annimmt.

- *Instabiler Zustand beim Primären Angreifer* Sind Roboter ähnlich nah am Ball, kann die Zuweisung des Primären Attacker teils mehrmals pro Sekunde wechseln, sodass keine Roboter letztlich an den Ball heranfährt. Hier könnte eine Hystere helfen.

Weiterentwicklungen für Eindhoven

Für Eindhoven soll das Konzept weiterverwendet und weiterentwickelt werden. Diese Änderungen beinhalten:

- Ausgliederung des primary Attacker als eigene Rolle; Dies soll Visualisierung verbessern und den Zustand stabilisieren. Außerdem vereinfacht dies den Zustandsautomat erheblich, da weniger Zustände benötigt werden.
- Bessere und Effizientere Berechnung der möglichen Passpunkte
- Eingliederung in das Eventsystem; Hierdurch sollen Berechnungen seltener und nur wenn sie tatsächlich gebraucht werden durchgeführt werden, um so die allgemeine Performanz zu steigern.

Ideen für Weiterentwicklungen

- *Just in Time Pässe*: Um den Gegner über das Ziel des Passes in Unklaren zu lassen, sollte der Empfänger so zur Position fahren, dass er wenige Millisekunden vor dem Ball dort eintrifft. Möglicherweise könnte er auch erst in die entgegen gesetzte Richtung fahren, um Gegner zusätzlich zu verwirren. Hierbei ist jedoch zu beachten, dass je nachdem wie viele Hindernisse sich auf dem Weg zur Passposition befinden, der Weg ggfs. etwas länger dauern kann.
- *Bessere Prädiktion des Gegnerverhaltens*, Um besser um den Gegner herum zu spielen, wäre es gut, dessen verhaltensweisen besser vorhersehen zu können. Vorsicht jedoch vor Overfitting, der Gegner wird auf unsere veränderte Spielweise ggfs. reagieren.

Flexwall

Die Flexwall ist ein Algorithmus zur Verteidigung des Tores und ein Teil des Taskmanagers. Die Flexwall wurde für den RoboCup 2023 im Rahmen des Taskmanager_Bod23 entwickelt. In diesem Artikel wird das zugrundeliegende Konzept vorgestellt. Außerdem werden die Erfahrungen sowie Ideen zur Weiterentwicklung dokumentiert.

> Da sich die Strategie schnell und stetig ändert, ist dieser Artikel als ein Einstieg in die Thematik zu verstehen und kann keinen Anspruch auf Aktualität erheben. Hierzu ist der tatsächliche Code heranzuziehen.

Motivation und Anforderungen

Wie aus dem Regelwerk der RoboCup Small Size League hervorgeht, ist es nur dem Torwart gestattet, sich im Strafraum aufzuhalten. Dies soll dem Verteidiger die Chance geben, trotz des Kameradelays Schüsse noch abfangen zu können. Im Vergleich zum menschlichen Fußball sind die Roboter und deren Schüsse deutlich schneller und dynamischer im Bezug auf die Spielfeldgröße als ihre menschlichen Vorbilder. In der Praxis bedeutet dies, dass öfter Torschüsse ausgeführt werden, da dieses auch schon von der Mittellinie problemlos anvisiert und getroffen werden kann. Die hohe Dynamik der Gegner macht es schwierig sämtliche Bewegungen vorherzusehen und Pässe abzufangen. Auch die im menschlichen Fußball verbreitete Raumdeckung ist aufgrund der hohen Ballgeschwindigkeit schwer umzusetzen.

Wie der Wettbewerb in Bangkok 2022 gezeigt hat, setzen viele Teams aufgrund dieser Umstände auf eine Vereidigung des Tores durch eine Mauer am Strafraum. Bereits zwei verteidigende Roboter sind hierbei sehr effektiv darin, direkte Torschüsse zu verhindern. Da die Mauer näher am Tor als der angreifende Spieler ist, muss sich die Mauer zudem kaum bewegen, um ein Dribbling des Gegners auszugleichen. Die einzige zuverlässige Möglichkeit mit einer solchen Verteidigung umzugehen besteht für den Angreifer darin, Pässe zu spielen und so den Ball zu einem Spieler zu bekommen, welcher an der Mauer vorbei direkt auf das Tor schießen kann. Dies erfordert jedoch ein hohes Maß an Planung und Koordination, zu welchem nur wenige Teams in der Division B in der Lage sind. Dennoch gibt es insbesondere in der Division A Teams, die aktiv an der Verteidigung vorbei passen und so eine Mauer einfach und effektiv ausspielen. Um hiergegen gewappnet zu sein, wurde für Bordeaux der Flexwall-Algorithmus entwickelt. Das Prinzip basiert auf einer flexiblen Mauer, die nicht nur die Schusslinie des ballführenden Spielers blockt, sondern auch die seiner potentiellen Anspielstationen. Diese Verteidigung ähnelt somit der eines Handball-Spiels, bei der der Strafraum abgeriegelt wird, wodurch der Gegner gezwungen ist, durch eine Vielzahl an

Pässen Lücken zu erspielen und diese auszunutzen.

Ein wesentlicher Unterschied zum Handball besteht beim Roboterfußball in der Größe des Tores. Dieses ist sehr groß im Vergleich zu einem Roboter, weshalb eine Mauer aus nur einem Spieler pro Gegner nicht sinnvoll ist. Stattdessen sollten gefährliche Gegner im Idealfall durch mehrere Verteidiger blockiert werden, um den Torschuss effektiv zu verhindern. Aufgrund der hohen Dynamik der Roboter und der damit verbundenen immensen Kontergefahr behält der Angreifer auch stets einige Verteidiger zurück. Dadurch hat der Verteidiger in der Regel einen Vorteil durch zahlenmäßige Überlegenheit, die in der Verteidigung ausgenutzt werden kann. Neben dem Torwart und der Strafraumverteidigung sollte es jedoch auch noch Roboter geben, die aggressiv versuchen Pässe zu verhindern und an den Ball zu gelangen. Die übrig gebliebenen Roboter können in der Flexwall mit einbezogen werden.

Aus diesen Überlegungen ergibt sich eine "*m zu n*"-Beziehung bei der *m* Verteidiger die Aufgabe haben, den Strafraum gegen *n* Angreifer zu verteidigen. Dies sollte unabhängig davon sein, wie groß und in welchem Verhältnis *m* und *n* sind. Zwar ist $m>n$ im Allgemeinen wünschenswert, kann jedoch aufgrund von gelben bzw. roten Karten sowie Beschädigungen der Roboter etc. nicht garantiert werden.

Funktion des Algorithmus

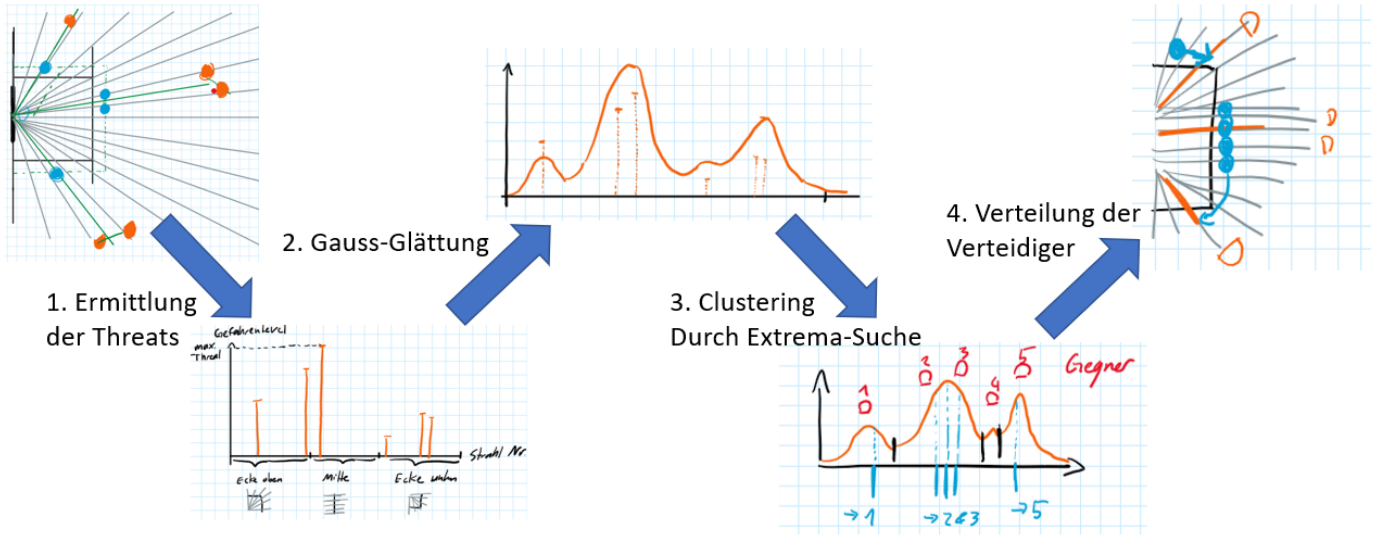
Im Folgenden wird der Algorithmus zum Zeipunkt Bordeaux 2023 beschrieben. Im Fokus steht dabei das grobe Konzept, für die konkrete Umsetzung sollte der Code herangezogen werden.

Flexwall-Algorithmus

Wie im vorherigen Abschnitt beschrieben wurde, muss eine gegebene Anzahl an Verteidigern das Tor gegen eine gegebene und möglicherweise größere Anzahl an Angreifern verteidigen können, weshalb es sinnvoll ist, Verteidiger möglichst effizient zuzuweisen und einzusparen. Dies kann durch sogenanntes *Clustering* realisiert werden, bei dem Gegner zusammengefasst werden, wodurch weniger Verteidiger erforderlich sind, um diese zu verteidigen. Diese so gewonnen Verteidiger können eine zahlmäßige Unterlegenheit teils ausgleichen oder ggf. eine stärkere Verteidigung der direkten Schusslinie ermöglichen.

> “ Clustering beschreibt Verfahren, mit denen Daten auf Ähnlichkeit untersucht und gruppiert werden können. Clustering wird oft eingesetzt, um Daten zu klassifizieren, also in Kategorien zuzuordnen (z.B. Objekterkennung in der Bildverarbeitung). Weitverbreitete Clusteralgorithmen sind z.B. K-Means, Meanshift und viele mehr.

Der allgemeine Algorithmus wird in der folgenden Abbildung visualisiert:



1. **Ermittlung der Threats**
   Im ersten Schritt werden zunächst die Positionen der Gegner erfasst. Hierbei sind jedoch nicht die xy-Positionen aus den Kameradaten relevant, sondern vielmehr der Winkel und Abstand der Roboter zum Tor. Dies liegt daran, dass letztenendes keine Regionen im Spielfeld sondern stattdessen die Schusslinien von den Robotern zum Tor verteidigt werden müssen. Liegen diese von verschiedenen Gegnern näherungsweise übereinander, so können oftmals beide Schusslinien durch nur einen Roboter verteidigt werden. In diesem Schritt wird für die einfachere Handhabung der Daten im Code auch zugleich eine Diskretisierung der Daten vorgenommen. Dies bedeutet, dass die Winkel z.B. auf den nächsten ganzzahligen Wert gerundet werden.
   Neben dem Winkel des Gegners ist auch dessen Threat relevant, der angibt, wie gut seine Schussposition sind. Roboter, die sich Nahe und mittig vor dem Tor befinden haben dabei im Allgemeinenen einen höheren Threat, als Roboter, die weit weg oder in der Ecke des Spielfeldes sind. Gefährliche Gegner erfordern möglicherweise mehrere Verteidiger für eine effektive Verteidigung. Die Berechnung der Threats ist dabei ein Teil des Observers.
2. **Gauss-Filterung des Threat-Arrays**
   In Schritt 1 wurden die Threat-Level für jeden Winkel zum Tor erfasst und als Array abgespeichert. Im nächsten Schritt wird dies durch einen Gauss-Filter (Wikipedia) geglättet. Wie in der Abbildung zu sehen ist, verschmelzen benachbarte Threats dabei zu einem gemeinsamen Berg, welcher zudem an Höhe gewinnt. Weiter entfernte Threats sind durch ein Tal voneinander getrennt. Das Ziel dieses Vorgehens ist es eine Basis für die Entscheidung zu treffen, welche Gegner zusammengefasst werden können. Außerdem kann die Höhe der Berge herangezogen werden, um festzulegen wie viele Verteidiger für diese Sichtlinien benötigt werden.
   Die Formel zur Filterung ist folgendermaßen definiert:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

Diese muss für jeden Balken im Diagramm einzeln angewandt werden, die Resultate werden im Anschluss aufsummiert um den in der Abbildung gezeigten geglätteten Graphen zu erhalten. Das σ in der Formel gibt die Varianz der Kurve an, die wiederum die Breite bestimmt. Je höher σ ist, desto breiter und Flacher werden die Berge. Dadurch wachsen benachbarte Säulen bei der Gauss-Filterung eher zusammen. Hierbei sind durch Tests in verschiedenen Simulationswerte sinnvolle Werte zu ermitteln. Außerdem ist zu beachten, dass σ auch von der Auflösung der Winkelgenauigkeit aus Schritt 1. abhängig ist.

3. **Clustering**
   In diesem Schritt wird der in Schritt 2. erstellte Graph verwendet, um zu bestimmen wie viele Verteidiger welcher Sichtlinie zugeordnet werden sollten. Hierzu werden zunächst alle Extrema, also lokale Maxima und lokale Minima im Graphen gesucht. Alle Gegner deren Winkel zum Tor zwischen zwei Minima im Graphen liegt, können zusammengefasst werden. In dem oben gezeigten Beispiel ergeben sich somit 4 Cluster. Das Maximum gibt an, wie viele Verteidiger zugewiesen werden sollten. Die Anzahl an verfügbaren Verteidigern wird vom Rolemanager Bordeaux 2023 bestimmt. Die Zuweisung erfolgt iterativ, wobei stets das aktuelle globale Maximum im Graphen betrachtet wird. Diesem wird ein Verteidiger zugewiesen und im Anschluss die Höhe des Maximums durch einen als Erosionsfaktor bezeichneten Wert reduziert. Die Berechnung des Erosionsfaktor kann beispielsweise durch die Division des Maximalen Threats durch die Anzahl der Roboter erfolgen. Im Anschluss wird das Verfahren wiederholt, wobei sich durch die Anwendung des Erosionsfaktors nun ggfs. ein neues globales Maximum ergeben hat.
   Dieses Vorgehen führt dazu, dass die Roboter nicht gleichmäßig auf die Cluster verteilt werden, sondern der Threat des Clusters als eine Gewichtung herangezogen wird. Ein Cluster mit doppeltem Threat sollte also auch doppelt so viele Verteiger erhalten. Es ist jedoch wichtig an dieser Stelle die Formel für den Erosionsfaktor sinnvoll zu wählen. Hierbei ist zwingend die Anzahl an verfügbaren Robotern sowie die Höhe des (höchsten) Threats zu berücksichtigen, damit verhindert werden kann, dass alle Verteidiger dem höchsten Threat zugewiesen werden.
4. **Verteilung der Verteidiger**
   In Schritt 3. wurde festgelegt, wie viele Verteidiger für jedes Cluster benötigt werden. Hieraus können nun die Positionen geschätzt werden, an denen sich die Roboter aufstellen müssen, um eine Mauer aufzubauen. In Bordeaux 2023 wurden diese Schätzungen vom Strategie-Code berechnet, da der verwendete Pfadplaner die Zielpositionen nicht im Vornherein augeben konnte, jedoch sollten in zukünftigen Iterationen diese Informationen direkt vom Planner ausgeführt und an das Flexwall-Modul übergeben werden. Sind die Zielpositionen bekannt, so kann eine Distanzmatrix erstellt werden, die den Abstand von jedem Roboter zu jeder Zielposition beinhaltet. Basierend

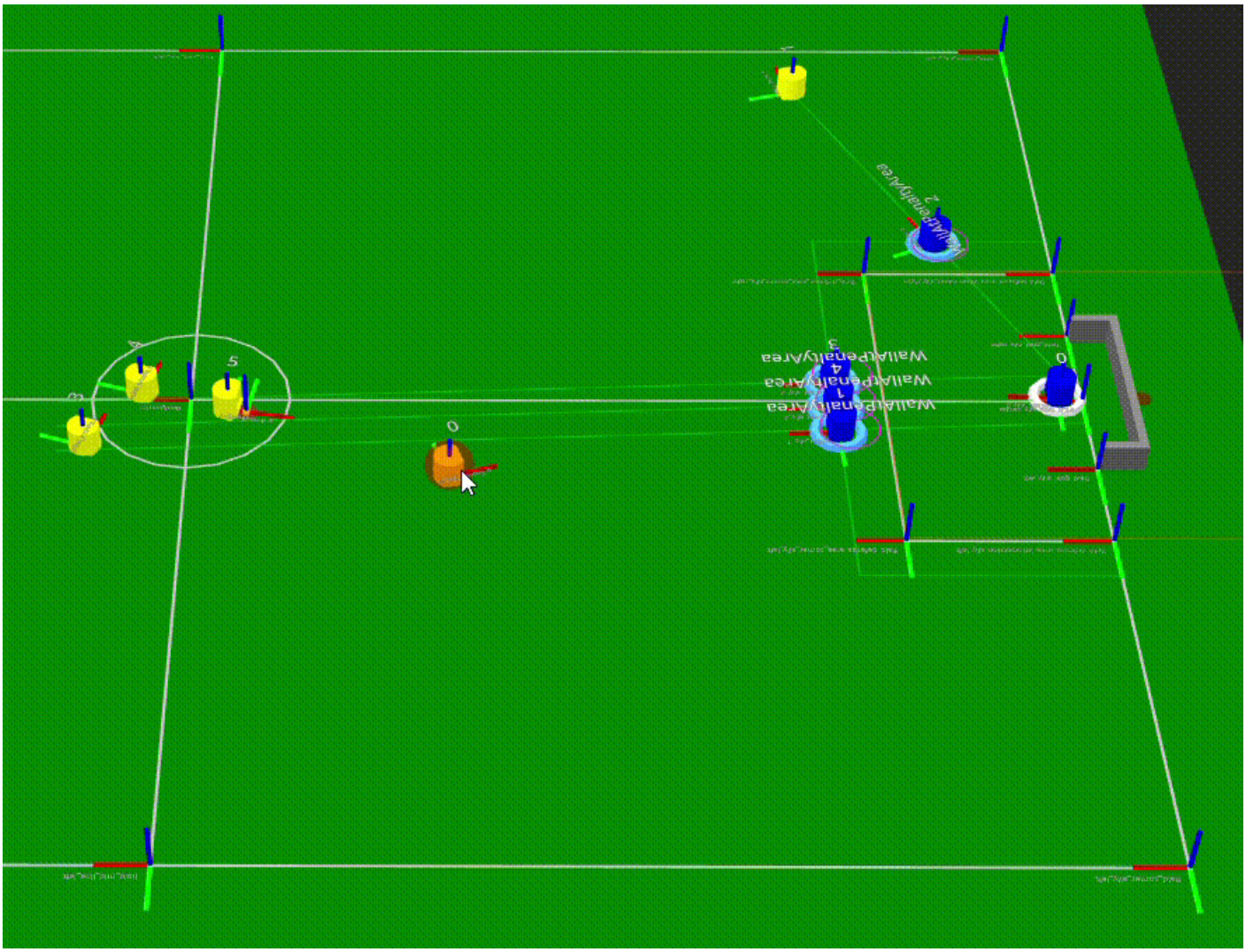
auf dieser Distanzmatrix kann mittels des Munkres-Algorithmus eine Zuordnung der Roboter zu den Positionen vorgenommen werden, sodass der quadratische Gesamtweg minimiert wird.

Das Resultat ähnelt einem Newton Pendel (Youtube); Anstatt dass ein Roboter an der Mauer vorbei kommen muss, reiht er sich beispielsweise links an das Ende der Mauer ein, während sich zeitgleich rechts ein Roboter aus der Mauer herauslöst. Dieses Verhalten minimiert nicht nur die Fahrtzeit der Roboter sondern verhindert auch Kollisionen und Chaos an unserem Strafraum. Um dieses Verhalten zu erreichen ist eine quadrierung der Kosten zwingend erforderlich, da dieses weite Wege besonders stark bestraft und somit verhindert, dass ein Roboter an einer Mauer vorbeigeschickt wird, anstatt den Newton-Pendel-Effekt zu nutzen.

> Munkres Algorithm (Wikipedia):
> Der Munkres Algorithmus (auch bekannt als Munkres-Kuhn-Algorithm, Hungarian Algorithm oder Linear Sum Assignment) ist ein Algorithmus, bei dem die ideale Zuweisung von *n* Agenten zu *n* Tasks gefunden wird, bei der die Gesamtkosten minimiert werden. Dabei wird davon ausgegangen, dass die Agenten (=Roboter) unterschiedliche Kosten (=Fahrtweg) für die verfügbaren Tasks (=Verteidigung von Cluster i) haben. Der Munkres-Algorithmus löst dieses Problem effizienter als ein Brute-Force-Algorithmus, dessen Ausführzeit faktoriell mit der Anzahl an Agenten bzw. Tasks zunehmen würde. Ein ausführlicher Wiki-Eintrag findet sich hier.

Flexwall in Aktion

Im Folgenden wird die Grundfunktion des Algorithmus anhand einer Simulation demonstriert. Die vier Verteidiger (blaue Roboter mit Hellblauem Ring) verteilen sich, um die fünf gelben Angreifer zu verteidigen. Der Ball befindet sich dabei in der Spielfeldmitte, weshalb die Roboter dort als besonders gefährlich eingestuft und durch meherere Verteidiger verteidigt werden. Der Angreifer oben wird durch einen einzelnen Verteidiger verteidigt. Der sich bewegende Angreifer wird teils durch einen einzelnen Verteidiger verteidigt. Fährt dieser durch die Spielfeldmitte, so wird er als ein Teil des mittleren Clusters aufgefasst, welches dann durch eine Mauer aus drei Robotern verteidigt wird. Hierbei zeigt sich auch der durch den Munkres-Algorithmus realisierte Newton-Pendel-Effekt.



Interception-Logik

Im vorherigen Abschnitt wurde auf die Verteilung der Roboter auf verschiedene Gegner-Cluster eingegangen. Im Idealfall ist dies bereits ausreichend, um den Gegner am Torschussversuch zu hindern und so den anderen Mitspielern Zeit und Möglichkeiten geben, den Ball zu erobern. Falls der Gegner jedoch trotzdem einen Torschuss riskiert, so wird dieser versuchen an der Mauer vorbei zu schießen. In diesem Fall kann es bei einer unzureichenden Größe der Mauer notwendig sein, ein Intercept-Manöver durchzuführen, um den Ball abzufangen. Hierzu wird geprüft, ob sich der Ball Momentan etwa Richtung Tor bewegt. Ist dies der Fall, wird der Flexwall-Task des besten Interceptors durch einen Intercept-Task überschrieben. Für die Ermittlung des besten Interceptors stellt der Observer Funktionen zur Verfügung, wobei insbesondere der Abstand des Roboters zur Balltrajektorie herangezogen wird.
Tests haben gezeigt, dass die Interception-Logik redundant ausgelegt werden sollte. Dies bedeutet, dass Verteidiger auch dann ein Intercept-Task zugewiesen bekommen sollten wenn schon ein Roboter mit unterschiedlicher Rolle einen Intercept-Task ausführt. Dies erhöht die Chancen, den Ball auch tatsächlich abzufangen. Es ist allerdings nicht erforderlich, dass mehrere Verteidiger gleichzeitig einen Intercept-Task durchführen, da sie sich sonst ggfs. dabei gegenseitig behindern.

Bewertung und Erfahrungen

- **Allgemein**
  Erstmals wurde die Flexwall in Crailsheim in Q1 2023 eingesetzt und einem Härtetest gegen die Mannheim Tigers und Erforce ausgesetzt. Die Flexwall hat gegen beide Teams trotz deren Erfahrung und Spielstärke im zufriedenstellenden Maße funktioniert und eine Vielzahl an Torschüssen abfangen können. Auch wenn einige Tore nicht verhindert werden konnten, ist von den Beobachtungen auszugehen, dass das Konzepts prinzipiell vielversprechend ist und eine deutliche Verbesserung zu Bangkok darstellt. Auch beim darauffolgenden RoboCup in Bordeaux23 hat sich die Flexwall als sehr effektiv erwiesen. Hier konnten jedoch auch nicht alle Torschüsse verhindert werden, was allerdings auch an der langsamen Beschleunigung der Roboter liegt, die Interceptmanöver erschwert. Außerdem hat sich gezeigt, dass die Pfosten teils nur schlecht verteidigt werden, da die Mauer stets die Tormitte verteidigt.
- **Threat Berechnung**
  Die in Bordeaux verwendete Berechnung des Threat-Levels basierte vor allem auf dem Abstand zum Ball, zu unserem Tor sowie dem Winkel zwischen der Tor- und Schusslinie. Diese Faktoren wurden als unabhängig voneinander angenommen und manuell gewichtet. Dabei wird jedoch nicht berücksichtigt, dass die Variablen abhängig sind, da z.B. ein schlechter Schusswinkel bei geringem Abstand weniger Schlimm ist als bei großen Entfernungen.
  Außerdem ist der Abstand zum Ball nur bedingt relevant. Gegner die nahe am Ballführenden Spieler sind, bieten für den Gegner keinen großen Vorteil, da ihre Schusslinie keine signifikante Verbesserung gegenüber dem Ballführer darstellt. Der verwendete Cluster-Algorithmus berücksichtigt dies teilweise, da nah beieinander liegende Threats zwar "verschmelzen", das daraus existierende Maximum jedoch stets kleiner als die Summe der Teil-Threats ist. Eine Ausnahme dabei ist, wenn die Gegner sich in dem exakt selben Winkel zum Tor befinden. Dies erhöht den Threat-Wert signifikant, obwohl diese Konstallation keine größere Bedrohung als ein einzelner Roboter darstellen sollte. Hier könnte der Cluster-Threat stattdessen durch den maximalen Roboterthreat im Cluster berechnet werden, doch berücksichtigt dies nicht, dass sich Gegner evtl. schnell aus dem Cluster lösen könnten und somit wieder weitere Verteidiger gebraucht werden. Um diese Problematik zu verbessern sollten verschiedene Konzepte erprobt und umfangreich getestet werden.

Weiterentwicklung für Eindhoven 2024

- **Umstellung von Polling auf Event-Basis**
  Anstatt eines regeläßigen Pollings soll die Flexwall auf Events reagieren. Dies ist zum aktuellen Zeitpunkt noch nicht eingebaut/getestet.
- **Überarbeitung der Munkres-Kostenfunktion**
  Der quadratische euklidische Abstand ist für die meisten Anwendungen ein gutes Kostenmaß, nahe des Strafraums ist jedoch eine Verwendung einer quadratischen Manhatten-Distanz sinnvoller, da die Verteidiger nicht durch den Strafraum hindurch fahren und sich somit parallel zu den Strafraumlinien bewegen müssen.

Ideen zur Weiterentwicklung

- **Fokus auf Ballführer**
  Verändert sich der Threat eines Gegners an der Flanke, kann es passieren, dass der Ballführende Spieler kurzzeitig nicht verteidigt wird, da die Roboter ihre Position verlassen, um den neu entstandenen Threat zu verteidigen. In diesem (kurzen) Zeitfenster kann der Gegner einen Torschuss durchführen. Hier könnten harte Beschränkungen eingebaut werden, die verhindern, dass der Ballführer unverteidigt bleibt. Dies resultiert jedoch in möglicherweise langen Fahrtzeiten, bis der neue Threat gedeckt wird sowie Kollisionen zwischen den Verteidigern, da die Mauer nicht zur Seite rücken darf.
- **Einbezug der Blickrichtung**
  Für eine bessere Vorpositionierung könnte sich die Mauer in Abhängigkeit der Blickrichtung des Gegners mitbewegen. Anstatt nur die Tormitte zu verteidigen wird somit antizipiert auf welchen Pfosten der Gegner zielt. Die Mauer sollte sich entsprechend bewegen. Dies verbessert die Vorpositionierung und erhöht die Chancen eines erfolgreichen Intercept-Manövers.
- **Handling von Gegnern nahe des Strafraums**
  Anstatt die Schusslinie von Gegnern zu decken, die sich sehr nah am Strafraum befinden, könnte auch der Passweg versperrt werden. Als besonders gefährlich haben sich in der Vergangenheit auch Gegner erwiesen, die direkt an der seitlichen Strafraumlinie stehen, einen quer über den Strafraum gelupften Ball annehmen und diesen in das Tor reflektieren. In Bordeaux wurde dies nur durch den Torwart verhindert, jedoch sollte bereits der Pass im Idealfall verhindert werden.
- **Verbesserung der Kostenfunktion**
  Der Threat des Clusters könnte sich auf die Kosten für den Munkres-Algorithmus auswirken, um so Prioriäten zu vergeben. Dies resultiert jedoch in längeren Pfadwegen für unwichtige Roboter und ggf. eine komplexere Pfadplanung und Kollisionen an unserem Strafraum.
- **Aktive Prädiktion des Gegnerverhaltens**
  Anstatt nur auf die aktuellen Positionen der Gegner zu reagieren könnte aktiv prädiziert werden, welche Aktionen der Gegner durchführen kann und wird, um die Reaktionszeiten zu verringern. Hierbei ist zu beachten, dass der Gegner auch stets auf uns reagiert und ggfs. seine Entscheidungen aufgrund unserer Prädiktion ändert.

- **Einbau eines Zweikampfverhaltens**
  Die Flexwall hat zum Zeitpunkt Bordeaux 23 noch keine Zweikampflogik eingebaut. Stattdessen wird darauf vertraut, dass der Rolemanager Verteidiger sobald es erforderlich wird zu Angreifern macht, um deren Zweikampflogik zu nutzen. Ein eigenes Zweikampfverhalten könnte hierbei sinnvoll sein, sofern dies nicht die eigentliche Offensive behindert und sinnvoll mit dem Rolemanager abgestimmt wird.

Oktopus

Der Oktopus Algorithmus ist Teil des Taskmanagers und für die Positionierung von Offensiven Spielern verantwortlich. Ziel dabei ist es, dass die sekundären Angreifer sich freilaufen und anspielbar machen, um den Ball zu empfangen und somit einen Fortschritt Richtung Gegnertor schaffen. Der Algorithmus wurde erstmals beim RoboCup 2023 im Rahmen des Taskmanager_Bod23 eingesetzt.

Der Oktopus Algoritmus war Teil des eTDP 2024 (siehe Anhang links).

Motivation

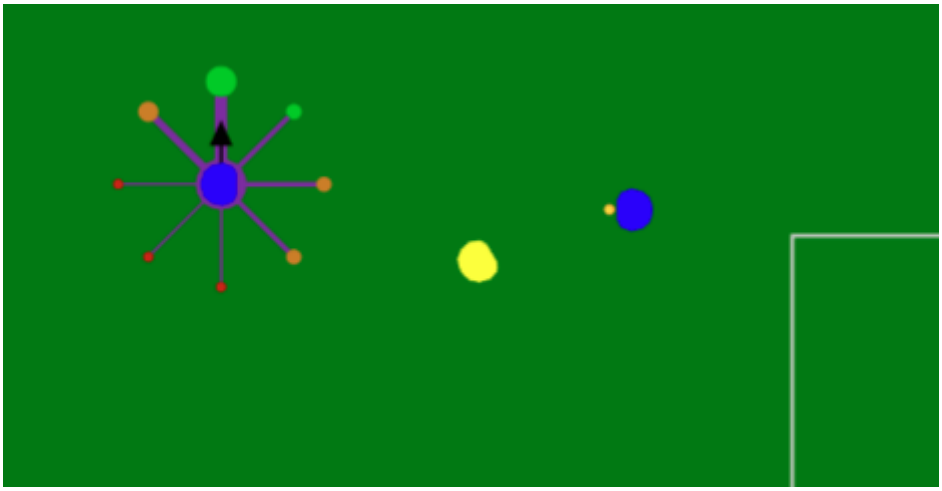
Der Oktopus Algorithmus wurde speziell für die Zusammenarbeit mit Passbäume_Bordeaux_2023 entwickelt. Die Passbäume analysieren das Spielfeld und die Position der sekundären Angreifern, also Robotern, die zwar an der Offensive beteiligt jedoch nicht im Ballbesitz sind. Der Passbaum evaluiert mögliche Pässe, wobei auch die Abfangwahrscheinlichkeit der gegnerischen Roboter berücksichtigt wird. Aus diesen Pässen wird die bestmögliche Passkette ausgewählt, die auch aus mehreren Pässen in Folge bestehen kann. Da der Passbaum nicht nur den Roboter an sich, sondern auch dessen Position bestimmt, sind für die Roboter die in einer Passkette eingeplant sind keine weiteren Schritte zur Positionierung notwendig.
Anders sieht es allerdings mit Robotern aus, die nicht in einer Passkette berücksichtigt werden. Diese Roboter sind an ihrer aktuellen Position nicht gut genug positioniert und sollten sich bewegen, um die Position zu verbessern und nach Möglichkeit in zukünftigen Passketten miteinbezogen werden. Position verbessern heißt dabei, dass die Roboter einerseits sich freilaufen sollten, sodass sich keine Gegner auf der Passlinie befinden und andererseits in eine gute Schussposition finden sollten, sodass nach einem erfolgreichen Pass ein Torschuss unternommen werden kann.

Algorithmus

Der Oktopus-Algorithmus wurde aufgrund seiner Ähnlichkeit in der Visualisierung nach dem gleichnamigen Tier benannt. Wie ein echter Oktopus besitzt ein Oktobot acht Tentakel, mit denen er seine Umgebung abtastet. Dabei wird für jedes Tentakel an der Spitze ein Score berechnet, der sich aus der Tor-&Passwahrscheinlichkeit an dieser Position zusammensetzt. Diese werden miteinander multipliziert, um einen Gesamtscore zu erhalten. Außerdem kann noch ein Bias eingeführt werden, dass der Roboter seine aktuelle Bewegungsrichtung möglichst weiter behält, um einen häufigen Wechsel der Bewegungsrichtung zu verhindern. Dies kann sinnvoll sein, um

einen Hysterese-Effekt einzuführen, der instabile Zustände vermeidet, bei denen der Roboter sehr schnell zwischen zwei Zuständen oszilliert und damit effektiv stecken bleibt.

Nachdem für jedes Tentakel der Score berechnet wurde, kann das beste Tentakel ausgewählt und als Richtung gewählt werden. Da der Oktobot nicht den Score seiner aktuellen Position kennt, sondern nur Scores in seiner Umgebung, ist er dazu gezwungen sich zu bewegen. Dies ist sinnvoll, da der Passbaum evaluiert hat, dass die aktuelle Position nicht ideal ist.



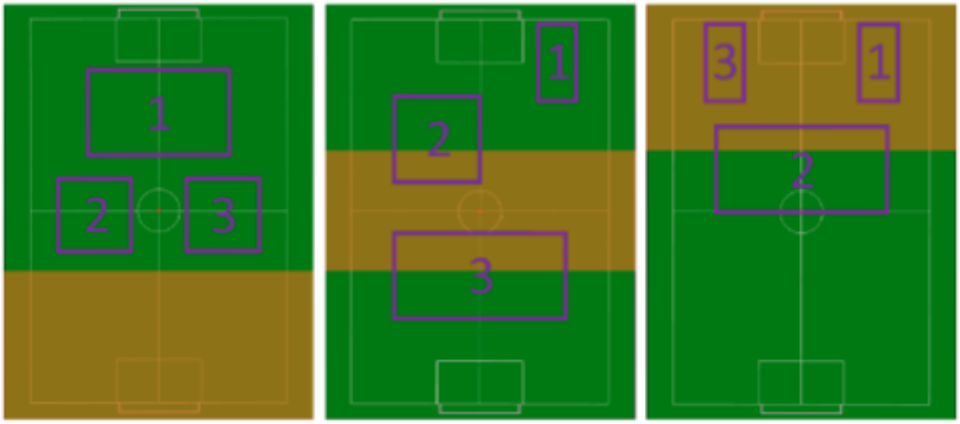
Da der Oktobot stets versucht in eine Richtung zu fahren, die die Pass- und Torwahrscheinlichkeit verbessert, kann er sich prinzipiell komplett selbstständig bewegen und positionieren. Dies hat jedoch auch den Nachteil, dass evtl Positionen angefahren werden, an denen der Roboter seine Anspielbarkeit maximiert, dabei jedoch die Torwahrscheinlichkeit verschlechtert. Außerdem könnte es passieren, dass mehrere Oktobots dasselbe lokale Optimum anfahren. Das ist allgemein zu vermeiden, da sich die Roboter sonst gegenseitig behindern und eine bessere Verteilung der Roboter auch dazu führt, dass der Gegner seine Verteidiger aufteilen muss, wodurch Lücken frei werden, die ausgenutzt werden können.

Daher sollte die Position der Oktobots auf Bereiche beschränkt werden können, die sicherstellen, dass die Position einigermaßen sinnvoll ist. Innerhalb dieser Bereiche sollte der Oktobot jedoch genug Freiheit haben, um seine Position selbst optimieren zu können. Durch eine geschickte Wahl der Bereiche lässt sich außerdem Domänenwissen einbauen, also die menschliche Intuition, wo etwa eine Positionierung sinnvoll wäre. Gleichzeitig hat der Roboter jedoch die Freiheit, seine Position innerhalb des Bereiches zu optimieren, wodurch sich der Algorithmus auch gut auf unbekannte Spielsituationen anwenden lässt, ohne dass man als Programmierer hierfür extra Fälle definieren muss.

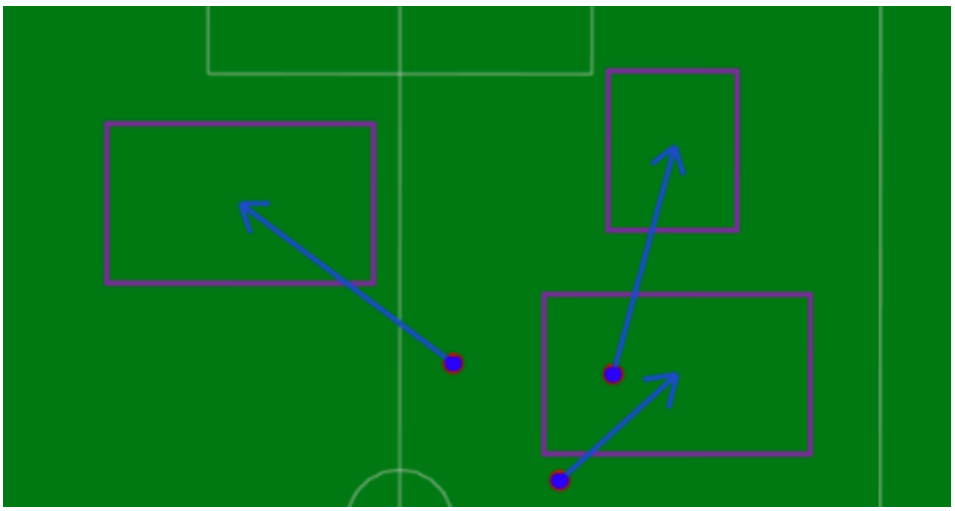
Die angesprochenen Bereiche werden im Allgemeinen als Aquarium bezeichnet. Befindet sich der Oktobot aktuell außerhalb seines zugewiesenen Aquariums, so soll er in dessen Mitte fahren. Sobald er sich im Aquarium befindet, beginnt die Abstastung und damit verbundene Positionsoptimierung. Außerdem lässt sich verhindern dass der Oktopus das Aquarium verlässt, indem für Tentakel außerhalb des Aquariums negative Scores gesetzt werden.

In der folgenden Abbildung wird gezeigt, wie sich die Aquarien definieren lassen. hier werden je nach Ballposition drei unterschiedliche Fälle definiert, die einem defensiven, ausgeglichenen oder

aggressiven Spielstil entsprechen. Jedes Aquarium hat dabei eine Priorität, die angibt, welche Aquarien besetzt werden sollten, falls es nicht genug Roboter gibt. Kleinere Zahlen werden dabei zuerst besetzt. Im defensiven Fall ist die Prioriät, denn Ball schnell nach vorne zu bekommen, um die aktuelle Situation zu entschärfen und die Gefahr eines Gegentores zu verringern. Im Ausgeglichenen und Aggressiven Fall sind Aquarien nahe des gegnerischen Strafraums, um die eigenen Torchancen zu maximieren. Außerdem werden auch Aquarien hinter dem Ball definiert, um Rückpässe zuzulassen. Diese erlauben einerseits die Durchführung von Reflexschüssen, bei denen der Ball ohne Annahme schnell aufs Tor geschossen wird, was schwer zu halten ist. Andererseits lassen sich durch Rückpässe auch die gegnerischen Mauern effektiv umspielen, da der Ball schnell an die andere Seite des Strafraums weitergeleitet werden kann.



Nachdem die Lage der Aquarien bestimmt wurde, müssen die Oktobots diesen zugeordnet werden. Dabei ist das Ziel, dass alle Aquarien so schnell wie möglich besetzt werden. Um dies zu gewährleisten wird der Munkres Algorithmus verwendet. Dabei werden als Kosten die quadratischen euklidischen Abstände verwendet, damit lange Wege besonders bestraft werden. Der Fokus liegt somit darauf, dass Bewegungen parallelisiert werden. Das bedeutet auch, dass bereits besetzte Aquarien wieder verlassen werden, damit andere Oktobots geringere Wege fahren müssen.



Erfahrungen und Kritik

In Bordeaux 2023 hat sich der Oktopus Algorithmus in Kombination mit den Passbäume_Bordeaux_2023 als äußerst effektiv erwiesen. Viele der Spiele konnten vor Ablauf der vollen 10 Spielminuten durch ein 10:0 frühzeitig beendet werden. Insbesondere durch den Einsatz der Passbäume war ein schnelles und effektives Ausspielen der gegnerischen Verteidigung möglich.

Der Oktopus Algorithmus ist dabei besser als andere Ansätze wie die kraftbasierte Positionierung oder die Restricted Radial Positioning, wie sie in Offense_Bod23 beschrieben werden, da die Roboter neben der Anspielbarkeit auch die Torwahrscheinlichkeit maximieren, statt sich nur auf die Anspielbarkeit zu fokussieren. Der Kompromiss aus Beschränkung durch die Aquarien sowie Freiheit ist dabei gut geeignet, um einerseits Domänenwissen miteinfließen zu lassen und gleichzeitig eine ausreichende Abstraktion für unbekannte Szenarien zu erreichen.

Weiterentwicklung

Für Eindhoven 2024 soll der Algorithmus weiterverwendet werden, dabei allerdings auf mehr Roboter ausgelegt sein, um auch in Div A verwendet werden zu können.