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 auzugleichen. 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

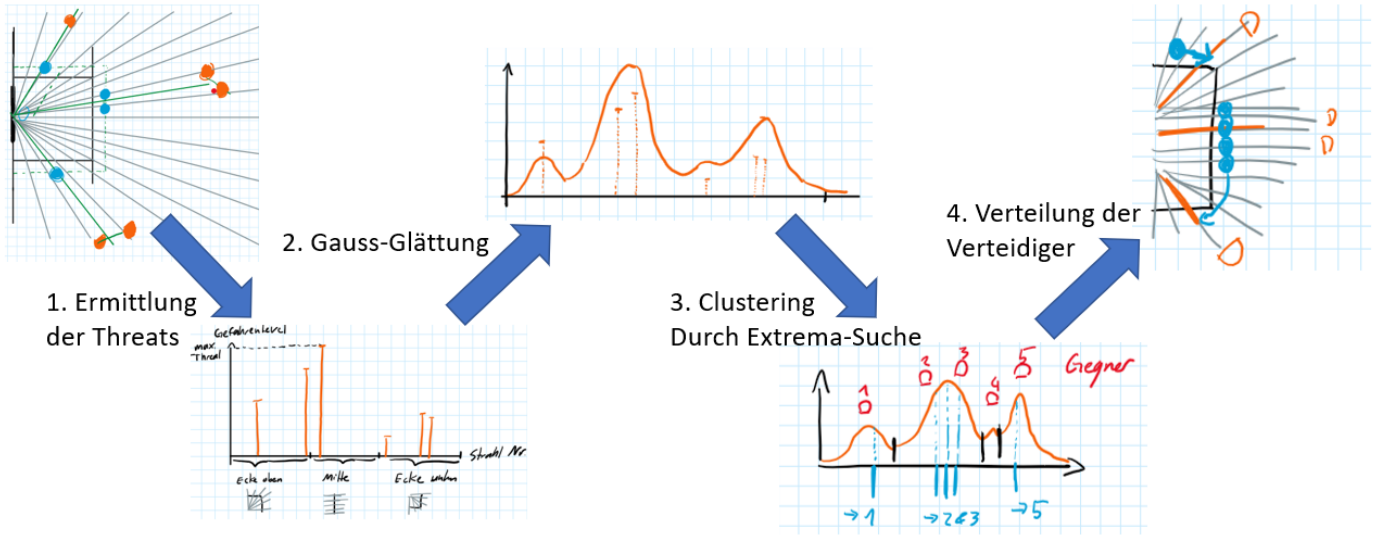
Im Folgenden wird der Algorithmus zum Zeipunkt Bordeaux 2023 beschrieben. Im Fokus steht dabei das grobe Konzept, für die konkrete Umsetzung sollte der Code herangezogen werden.

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

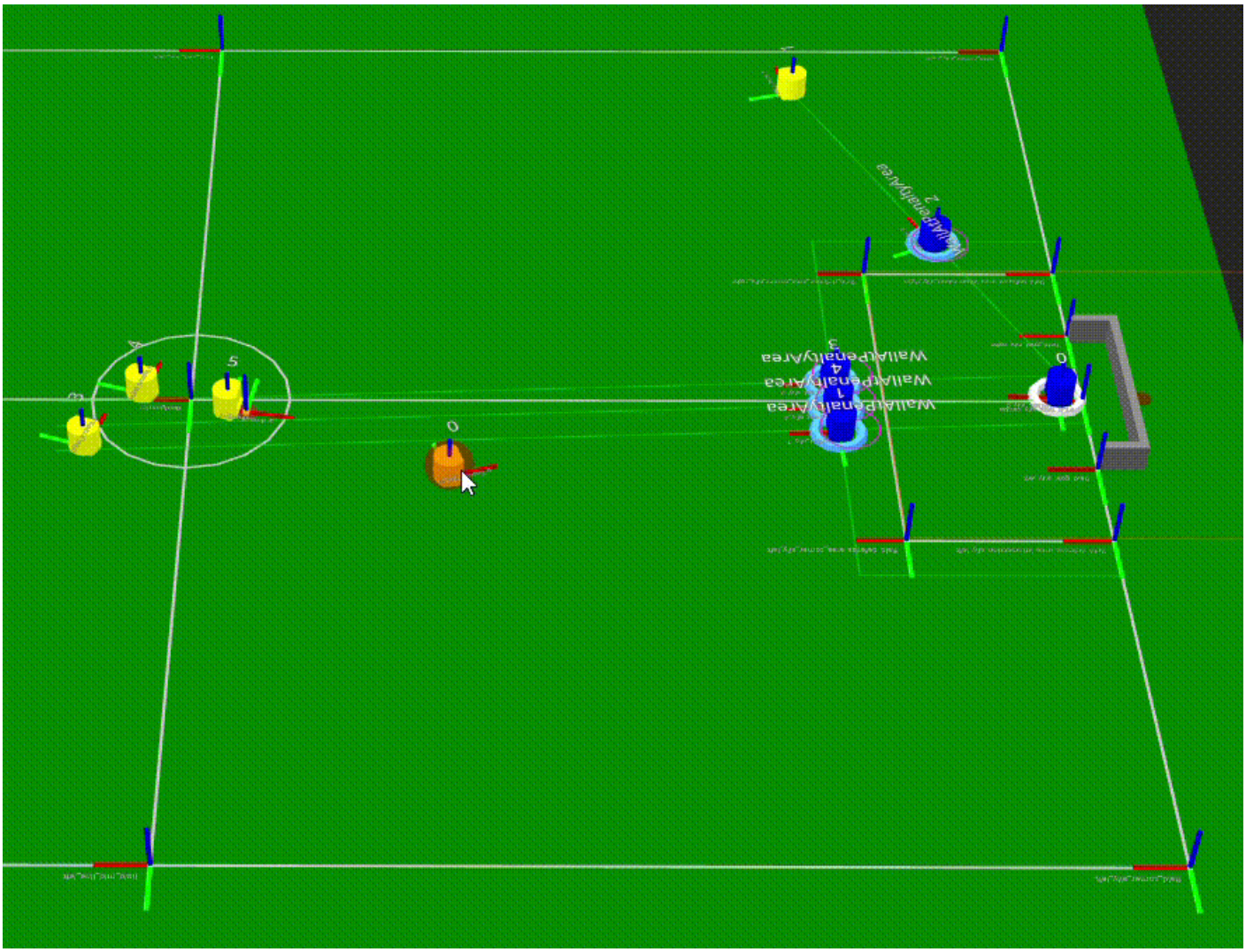
auf dieser Distanzmatrix kann mittels des Munkres-Algorithmus eine Zuordnung der Roboter zu den Positionen vorgenommen werden, sodass der quadratische Gesamtweg minimiert wird.

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.

Revision #5
Created 27 July 2025 18:29:45 by Malte Sparenborg
Updated 7 August 2025 15:09:33 by Malte Sparenborg