Interface gebundene Konstruktion eines Control Sets nach Pivtoraiko and Kelly

Eure Namen fehlen

1.Einleitung

Um Wegfindungsalgorithmen auf Bewegungsprimitiven, unter Beachtung von nichtholonomen Bewegungen, zu verwenden, müssen diese Restriktionen für den Algorithmus definiert werden.

Das zu erzeugende "Control-Set" stellt die Menge aller Bewegungen dar,die das nichtholonomische Konstrukt ausführen kann.

Alle weiteren Wegfindungsalgorithmen können mithilfe des Setsdie Nichtholonomintät des Konstruktes einbeziehen.

Der Report beschäftigt sich mit der Implementation des Codes mit dem solch ein Control Set, erstellt werden kann.

Im folgenden Paper erläutern wir erst die erfolderlichen Modellen und ihrer Umsetzung (2.0) um dann die Implementation des im Paper [1] vorgeschlagenn Algorithmus zu beschäftigen (3.0).

Dabei gehen wir jeweils zunächst auf die Modelleriung ein und erläutern die Verwendung im Bezug zum Control-Set.

Anlschießend befassen wir uns mit den implementationstechnischen Feinheiten dieser Modelle.

2. Datenstrukturen

2.1 ControlSet

Ein Control Set ist eine Menge von allen möglichen Bewegungen, die Minimalisiert und Zerteilt wurden.

Dadurch sind die Bewegungen nicht subsituierbar durch andere Bewegungen im Set.

Das Control Set wird von suchbasierten Algorithmen genutzt.

Ein Grund für die Nutzung des Control Sets ist, dass an Zeit und Rechenaufwand für die lokale Plannung gespart wird, weil das Erstellen eines Control Sets nur einmalig

notwendig ist und dieses wiederverwendbar ist.

Weitere Nutzen des Control Sets werden im Paper [1] erwähnt.

Im Code wird das Control Set als eine Map von Positionen und Bewegungen implementiert.

Bei der Erzeugung eines Control Sets wird auf den Bewegungserzeuger und einem Intervall, den man selbst bestimmt, geachtet.

Bei der Nutzung des Control Sets ist es Vorteilhaft, dass die Bewegungen in dieser Art abgespeichert werden, weil man sich schnell und gezielt Bewegungen in Abhängigkeit der Position aus dem Control Set abrufen kann.

Auch ist das Control Set abhängig von den Bewegungserzeugern, was dem Nutzer die Freiheit gibt, den Code anzupassen.

Zum Beispiel kann der Benutzer zwei Control Set generieren lassen mit unterschiedlichen Bewegungserzeugern und den selben Intervall und diese miteinander vergleichen.

2.2 Bewegung

Die Bewegung stellt eine abstrahierte "Bewegung" wie sie auch natursprachlich verwendet wird dar. In unserem Modell wird jede Bewegung von etwas erzeugt (siehe Bewegungserzeuger) und kann beliebig oft reproduziert werden.

Außerdem hat sie eine variable Startposition und einen davon abhängigen Verlauf.

Für ein Control-Set ist eine Bewegung die elementare Datenstruktur

In unserer Implementation hat eine Bewegung einen relativen Verlauf, einen Erzeuger samt Erzeuger-Argumente und eine Start Position.

Mithilfe des Erzeugers kann die Bewegung tatsächlich von einem Agenten "ausgeführt" werden.

Der relative Verlauf ist eine Liste, in der jeder Eintrag einem "Zwischenstopp" entspricht.

Jeder Eintrag besteht aus einer Berechnung für jede Dimension in der sich etwas ändert.

(Das Konzept der Dimensionen in underem Modell wird 2.4 näher erläutert)

Mit dieser Abstraktion kann der Verlauf von einer Bewegung mit einem beliebigen Startpunkt berechnet werden.

2.3 Bewegungserzeuger

Unter dem Konzept der Bewegungserzeuger versuchen wir in unserem Modell alle Möglichkeiten eines Agenten eine Bewegung zu erzeugen zu modellieren.

Ein Bewegungserzeuger zeichnet sich dadurch aus, dass er eine endliche Menge von Bewegungen erzeugen kann.

Hierbei haben wir die Implementation bewusst frei gelassen, um eine maximale Anpassung zu erlauben.

In unserem Konzept definiert man für einen Agenten einen sinnvollen Bewegungserzeuger (z.B. einen Motor) und kann danach nur noch die allgemeinere Klassen "Bewegung" benutzen.

Insbesondere bei Austausch des Antriebssystem kann die komplette Plannung (solange sie auf Control-Sets beruht) beibehalten werden. Es muss lediglich ein anderer Bewegungserzeuger benutzt werden.

In unserem Modell ist ein Agent nichts weiter als eine Menge von Bewegungserzeugern, mit denen er sich fortbewegt.

Diese Erzeuger stellen die Gesamtmenge an Bewegungen dar.

In dem Algorithmus entsteht aus genau dieser großen Menge nachher das Control-Set.

In unserer Implementation existiert lediglich ein Interface (siehe oben).

Das Zusammenspiel von Bewegung und Bewegungserzeuger enstspricht der Fabrikmethode nach GoF .

2.4 Position

Eine Position ist ein benutzerdefinierte Koordinate. Sie enthält, die vom Nutzer gewählten Werte und Vektoren und ist der kleinste Bestandteil einer Bewegung.

Die Position wird für die Definition einer Bewegung verwendet. Sie beinhaltet Informationen wo der Agent sich befindet und in welchem Zustand.

In unserem Modell lässt sich der Gesamtzustand eines Agenten als Reihe von Werten für Dimensionen beschreiben.

Eine solche "Dimension" kann dabei sowohl eine tatsächliche physikalische Dimension wie eine X-Achse, als auch eine beliebige Variable samt Belegung, wie die Posiition eines Greifarmes sein.

Wir untersscheiden lediglich zwischen Dimensionen die als Strecke dargestellt werden können (wie z.B. Zeit, x-Achse, ...) und sogennaten "User defined Dimensions" (UdFs) welche alles andere abdeckt.

Möchte man z.B. die Entfernung zwischen zwei Positionen wissen, so kann man einfach die Differenz aller Dimensionen die keine UdFs sind entweder als Tupel oder als mit der geometrischen Summenformel verrechneten Gesamtwert angeben.

Technisch ist die Belegung aller Dimensionen ein double, da dies genug Genauigkeit bietet.

Zusätzlich hat jede Dimension eine Toleranz der ebenfalls in Form eines double hinterlegt wird. Dieser Wert beschreibt wie groß die Differenz in dieser Dimension maximal sein darf, so dass der Wert noch als gleich angenommen wird.

In unserem Modell werden alle Bewegungen über ihren Bewegungserzeuger erstellt, so dass jeder Bewegungserzeuger für jede Bewegung individuell Dimensionen und Toleranzen verändern kann.

Somit kann jede denkbare Art einer Bewegung modelliert werden.

2.5 Bewegungstunnel

Wenn man eine Bewegung als einfachen Vektor interpretiert, dann ist ein Bewegungstunnel eine Menge dieser Vektoren.

Diese Tunnel kann man sogar graphisch darstellen, wenn man die Vektorgraphen verbindet.

Als spezielle Eigenschaft haben alle Vektoren ein gemeinsamen Startpunkt und einen gemeinsamen Endpunkt.

Um zu prüfen ob ein Bewegung sich in andere zerlegen lässt (wie in #PAPER beschrieben) sind Bewegungstunnel hilfreich.

Man muss lediglich überprüfen ob der Tunnel mit anderen Bewegungen

"durchquerbar" ist. (siehe 3.0)

Es ist auch leicht möglich die Bewegung zu finden, die möglichst dicht in der Mitte liegt. Hiefür wählt man einfach den Vektor mit der kleinsten geometrischen Länge aus.

3.0 Implementation des generateControlSet

Ein Control-Set besteht aus Bewegungen, die in einer Map abgespeichert sind. Um eine bestimmte Bewegung aufzurufen, wird eine Position angegeben und die Bewegung wird ausgegeben.

Zu erst werden alle Bewegungen erzeugt und als eine Menge von Bewegungen abgespeichert.

Für diese werden, wenn sie eine bestimmter Länge (die entwerder kleiner oder gleich dem bestimmten Intervall) haben, Endpunkte berechnet, die wieder als eine Menge von Bewegungen abgespeichert werden.

Daraus lassen sich Bewegungstunnel finden. Um abzusichern, dass ein Bewegungstunnel b_1 nicht weiter zerteilt werden kann, wird b_1 mit allen anderen Bewegungstunneln verglichen, ob man b_1 mit einer der anderen Bewegungstunneln den Endpunkt von b_1 erreichen kann. Wenn dies der Fall ist, wird b_1 entfernt und der Algorithmus wählt einen weiteren Bewegungstunnel als b_1 aus der Bewegungstunnel Menge und wiederholt den Vorgang.

Nun werden noch die übrigen Tunneln Normalisiert, wobei die kürzesten Bewegunge jeder der Tunneln im Control-Set abgespeichert werden.

4.0 Bewertung/Kritik/Fazit

5.0 Quellen

[1] M. Pivtoraiko, A. Kelly, Generating near minimal spanning control sets for constrained motion planning in discrete state spaces, In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), 2005, pp. 3231–3237.