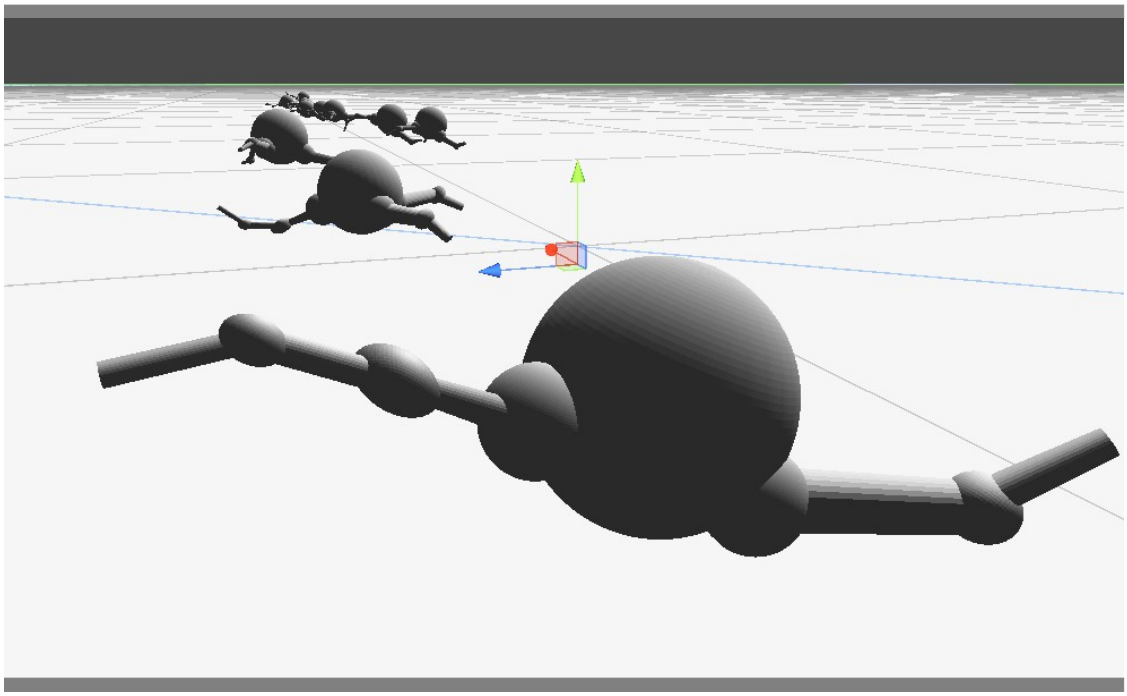

Genetische Algorithmen & geometrische Algebra

Bericht

Projekt

Studiengang:	Informatik
Autor:	Christian Ritz, Christian Märki
Betreuer:	Stämpfli Marx
Datum:	11.12.2014

Abstract

In dieser Arbeit wird untersucht ob es möglich ist eine Fortbewegungsart eines dreibeinigen Roboters mithilfe von geometrischer Algebra zu beschreiben und mit Genetischen Algorithmen zu optimieren. Es wird im ersten Teil beschreiben wie sich ein dreibeiniger Roboter bewegen könnte und wie diese Bewegung simuliert werden kann. Dabei wird gezeigt in welcher Art sich die einzelnen Gelenke biegen müssen. Weiter wird beschreiben wie die Geometrische Algebra angewandt werden kann, um die Rotationen der Beine zu berechnen. Mit Hilfe einer Fitnessfunktion die bestimmt wie gut sich der berechnete Roboter bewegt kann danach der Genetische Algorithmus diese Bewegung verbessern. Zum Schluss wird gezeigt welche Form der Visualisierung verwendet wurde um die Roboter darzustellen.

Inhaltsverzeichnis

1 Einleitung

Die Grundidee des Projektes ist es die geometrische Algebra zu verwenden um die Bewegung eines virtuellen, mehrgliedrigen Roboters zu beschreiben. Es soll versucht werden einen dreibeinigen Roboter wie unten dargestellt mithilfe von genetischen Algorithmen eine möglichst gut funktionierende Art von Vorwärtsbewegung beizubringen.

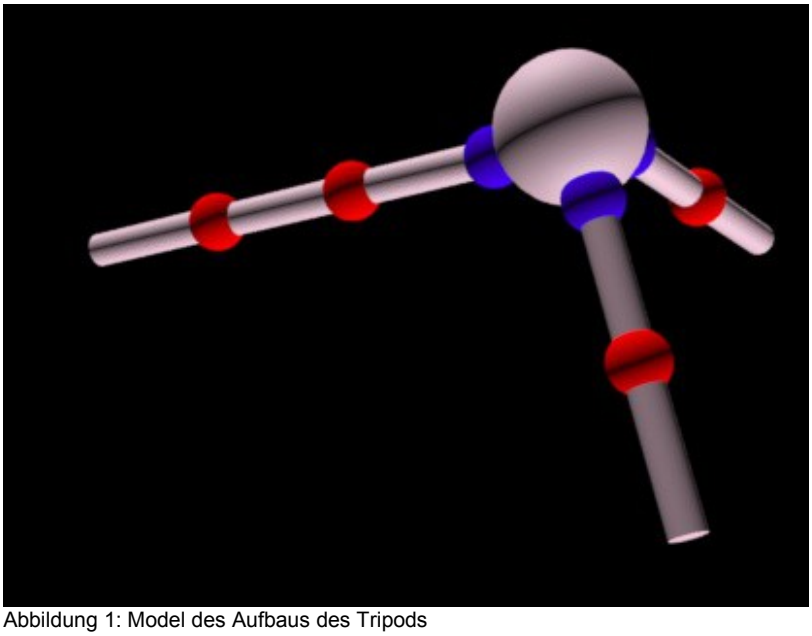

Abbildung 1: Model des Aufbaus des Tripods

Die vereinfachte Darstellung oben soll das Konzept aufzeigen. Rote Kugeln stellen hier einfache Gelenke dar welche nur auf einer Achse bewegt werden können. Blaue Kugeln sind 3d Gelenke. Die Bewegung welche vollzogen wird soll mit geometrischer Algebra beschrieben werden. Es werden physikalische Vereinfachungen vorgenommen.

2 Methodik

2.1 Generelle Idee

Die Bewegung eines Tripod-roboters ist eine komplexe Angelegenheit welche nicht in der Natur beobachtet werden kann und schwierig ist sich vorzustellen. Anders als bei bipedalen Geschöpfen ist es daher auch nicht möglich die Bewegung durch Motion Capture einzufangen um echtes Laufen zu simulieren. Allerdings, so die Hypothese, ist es möglich mit genetischen Algorithmen ein Laufverhalten zu generieren. Insbesondere weil es wenig Constrains hat und damit wenig ungültige Lösungen bietet sich dieser Lösungsansatz an. Damit ein genetischer Algorithmus aber angewandt werden kann muss zuerst die Bewegung beschrieben werden können.

2.2 Beschreibung der Bewegung

Die Bewegung der Beine des Roboters ist eine Folge von Krümmungen der Gelenke an verschiedenen Zeitpunkten. Um die komplexe Bewegung herunterzubrechen wird zuerst ein einzelnes Gelenk betrachtet. Später kann dann die Gesamtbewegung als Summe mehrerer Einzelbewegungen aller Gelenke definiert werden. Die Krümmung eines Gelenkes wird zu einem Zeitpunkt t0 anfangen und bei einem Zeitpunkt t1 vollendet sein. Es ist durch die Natur des Gelenkes vorgegeben in welchem Bereich sich das Gelenk krümmen kann. Die meisten Gelenke können nicht über 180° gebogen werden. Ein Ellbogen eines Menschen ist hier ein Beispiel. Theoretisch wäre es aber auch möglich Gelenke zu konstruieren welche jegliche Beugung zulassen. Daher handelt es sich bei dieser Regel um eine rational bestimmte Vereinfachung um die Komplexität des Experiments zu minimieren. Eine weitere Vereinfachung ist die Beschleunigung. Im realen wird mit einer veränderten Geschwindigkeit bewegt. Es wurde die Entscheidung getroffen alle Bewegungen eines Gelenkes mit konstanter Geschwindigkeit zu vollziehen. Ein Gelenk kann jedoch für eine Bewegung mehrmals die Richtung ändern. Daher liegt es nahe die Zeit welche für die Bewegung zur Verfügung steht in mehre Bereiche zu segmentieren. Für jedes Segment wird ein Wert zwischen 0-1 angegeben. Dieser Wert bestimmt der Winkel an diesem Zeitpunkt relativ zu der kompletten Krümmung und gemessen vom gerade gestreckten Gelenk. Wenn ein Winkel sich von 0 -180 bewegen kann so würde der Wert 0.5 angeben das das Gelenk an diesem Zeitpunkt 90° geöffnet wird. Mit diesen Daten kann eine Funktion beschrieben werden welche den Zeitpunkt seit Programmstart nimmt und den Winkel zurückgibt welcher die Gelenköffnung an diesem Moment beschreibt. Natürlich kann dieser gegebene Zeitpunkt zwischen zwei Segmentierungen liegen. Daher muss die Funktion linear interpolieren. Die lineare Interpolation weil sie einfach zu berechnen ist und für die gegebene Situation keine genaueren Daten vorhanden waren welche Anlass gaben eine andere Interpolation zu nehmen.
Die Bewegung eines Gelenkes ist damit vollständig beschrieben. Die Gesamtbewegung definiert sich jetzt als periodische Wiederholung der Summe der Einzelbewegungen (Wie oben beschrieben) aller Gelenke. Eine Periode ist die Dauer der Bewegung bis zum Zeitpunkt wo sie anfängt sich wieder zu wiederholen.

2.3 Fitness

Die Berechnung der Fitness ist ein zentrales Element des genetischen Algorithmus. Sie ist mitverantwortlich für die Effizienz und damit für die Dauer einer erfolgreichen Berechnung des Algorithmus. Da es sich bei diesem Projekt jedoch eher um ein Machbarkeitsnachweis handelt statt einer praktischen Implementation ist die Effizienz der Fitnessfunktion sekundär. Zudem scheint es nicht dramatisch zu sein wenn einige Stunden für die Berechnung aufgewendet werden müssen falls es sich um eine einmalige Berechnung handelt und am Ende die vollständige Bewegung digital vorhanden ist.
Es war den Autoren bekannt das es Frameworks gibt welche physikalisch weit fortgeschritten sind währendem sie benutzerfreundliche und vor allem anderen zeiteffiziente Handhabung erlauben. Solche Physikengines übernehmen weitgehend bestimmbare Eigenschaften und erzeugen einen virtuellen Raum in welchem die gegebene Physik Anwendung findet. Es wurde angenommen das wenn ein Model des Roboters in einen solchen Raum gestellt wird und die Laufanimation gestartet wird, das sich der Roboter in einer teilweisen skurrilen Art und Weise fortbewegen wird. Natürlich würde der gewählte Raum die physikalischen Eigenschaften der realen Welt vereinfacht anwenden. Die Fitness kann daher anhand der zurückgelegten Strecke gemessen werden. Aus zeitlichen Gründen ausgeschlossen sind folgende Faktoren:

- Energieverbrauch
    - Fortbewegung mit wenig Energieverbrauch ist vorzuziehen

- Abzug für unerwünschtes Verhalten
  - Wenn zum Beispiel der Körper den Boden berührt ist dies unter Umständen nicht erwünscht
- Dämpfung
  - Muskeln beschleunigen und bremsen. Natürliches Verhalten müsste dies simulieren
- Erschütterung
  - Um potentieller Materialverschleiss vorzubeugen
- Motorische Veränderbarkeit
  - Möglichkeit schnell zum Bremsvorgang zu wechseln oder Kurven durchzuführen

Würden sämtliche Faktoren eingeschlossen wäre es theoretisch möglich natürliches Laufverhalten zu erhalten. Auf dem Web wurde eine Arbeit gefunden welche dies bezeugt (*Geijtenbeek et al., 2013*).

Für die Implementierung wurde Unity 3d gewählt. Diese Engine wurde primär für die Gameindustrie kreiert bietet aber für dieses Experiment alle benötigten Werkzeuge und Handhabung. Das Team hatte zudem Vorkenntnisse dieses Programms wodurch lange Einarbeitungszeiten vermieden werden konnten. Zeitgleich übernimmt Unity 3d die visuelle Repräsentation wodurch qualitativ hochwertige Demonstrationen möglich sind. Der gewählte Weg hat jedoch den Nachteil das die Performance leidet und nicht so viele Berechnungen pro Sekunde möglich sind. Für arbeiten welche zeitlich weniger Begrenzt sind wären eigenständige Physikengines kombiniert mit einer selbst entwickelten Ansteuerung besser geeignet. Insbesondere eine Kollisionsberechnung über die Grafikkarte (PhysX) könnte sehr viel bewirken. Es könnte immer noch eine kleinere Grafikengine eingebunden werden um stichprobenartiges Rendering von einzelnen Laufverhalten zu simulieren.

2.4 Geometrische Algebra

Um die geometrische Algebra zu implementieren wurde auf die Bibliothek GAIGEN 2.5 (*GAIGEN 2.5*) zurückgegriffen. Diese funktioniert zwar einwandfrei aber muss für C# Benutzer eigenständig über eine Konsole generiert werden. Der Prozess erscheint am Anfang abschreckend aber bald fällt auf das die meisten Parameter schon vordefiniert sind und nur ein paar Einstellungen abgeändert werden müssen. Eine umfangreiche Anleitung ist auf der GAIGEN Webseite vorhanden (GAIGEN 2.5).

2.4.1 Geometrische Algebra im Allgemeinen

Eine feste Bedingung dieses Projektes zum Zeitpunkt der Projektwahl war es mit geometrischer Algebra ein Projekt zu realisieren. Diese hat jedoch gegenüber der konservativen Matrixberechnungen im euklidischen Raum schwächere Performance (*Dorst, 2007, S.555*). Es wurde daher ein Projekt gesucht welches nicht Echtzeitberechnungen erfordert. So ist es möglich die visuelle Repräsentation von Unity 3d zu deaktivieren und die Fitness rein durch Berechnung zu bestimmen. Wenn der genetische Algorithmus seine Arbeit vollendet hat wird dann für jedes Gelenk das Objekt welches die Bewegungen beschreibt herausgelesen und in ein anderes Programm eingebunden. Ob der Computer jetzt ein paar Minuten oder Stunden rechnet ist im praktischen nicht relevant da es sich um Vorarbeit handelt. Die Implementation konnte dies leider nicht mehr berücksichtigen da die Zeit zu knapp wurde. Die Beschreibung der Gelenke ist universell und kann als Vorlage der Bewegungsbeschreibung eines realen Roboters oder für die Animation eines fiktiven Spiel verwendet werden.

2.4.2 Beschreibung der Gelenkkrümmung

Für die Beschreibung der Bewegung reicht es die Bewegung eines Gelenkes zu definieren. Dazu wurde ein Rotor der geometrischen Algebra genommen. Dieser Rotor beschreibt eine zeitlose Rotation auf einer bestimmten Achse. Wegen einfacherer praktischer Umsetzung wurde das 3d Gelenk als zwei unabhängige Einzelgelenke betrachtet. Die Bewegung des 3d Gelenkes setzt sich demnach aus zwei einfachen Gelenken mit senkrecht zueinanderstehender Achse zusammen. Auf die geometrischer Algebra bezogen ist das die Kombination zweier Rotoren. Die zeitabhängige Veränderung und damit verbunden wann um wie viel das Gelenk gebogen wird wurde der Einfachheit wegen durch Programmierlogik in einer eigenständigen Klasse extrahiert.

2.4.3 Rotor

Was aber ist ein Rotor? Gegeben sei dieser Rotor der geometrischen Algebra:
R = a + b*e1^e2 + c*e2^e3 + d*e3^e1

a,b,c,d sind numerische Werte. Der Drehwinkel (phi) wird folgender Weise einbezogen:

a= sin(phi)
(b,c,d) = cos(phi)*n,

Ein Vektor würde mit den obigen Parameter einen Vektoren um phi Grad um die Achse n drehen. n repräsentiert den normierten Rotationsvektor welcher senkrecht zur Rotationsebene ist. Weiter hat es e1,e2 und e3 in der Formel. Für Neueinsteiger in die geometrische Algebra ist es am einfachsten diese Standartelemente durch einen analogen Vergleich in der euklidischen Geometrie als Einheitsvektoren der Achsen x,y und z zu betrachten. Verwirrend kann am Anfang auch das Zeichen ^ sein. Dieses wird äusseres Produkt genannt. Mit e1 ^ e2 wird ein Bivektor beschrieben. Ein Bivektor ist generisch ausgedrückt ein Stück Ebene. Er liegt in einer bestimmten Ebene hat aber ein Betrag basierend aus den ausdrückenden Vektoren. Diese entspricht der Fläche des Rhomboiden welcher durch die beiden Vektoren des geometrischen Produktes aufgespannt werden kann. Damit nicht genug hat ein Bivektor auch noch eine Orientierung. Die Orientierung wird durch die Reihenfolge von den Vektoren gegeben welche den Bivektor definieren. Somit hat e1^e2 und e2^e1 entgegengesetzte Orientierung. Zurück zu den Parametern. Durch b,c und d wird die Ebene definiert auf welcher gedreht wird, abhängig davon wie viel von jedem Bivektor genommen wird. Um sich daran zu gewöhnen mag es hilfreich sein sich an die Definition eines Punktes im 3D euklidischen Raum zu erinnern. P = (b*ex,c*ey,d*ez). Analog zu wie viel von b,c und d genommen wird, wird unterschiedlich viel von den Einheitsvektoren ex, ey und ez genommen und damit ein anderer Punkt beschrieben.
Die Anwendung eines Rotors auf einen Vektor ist simpel. Wenn der Vektor v durch den Rotor R gedreht werden soll:
v' = R*v*inverse(R)
v' entspricht dem neuen Vektor welcher durch R gedreht wurde.
Es ist schwierig eine neue Geometrie in wenigen Sätzen zu beschreiben. Es soll hier auf (*Dorst, 2007*) verwiesen werden wo in Rund 600 Seiten diese Geometrie verständlich und mit Beispielen erklärt wird.

2.5 Genetischer Algorithmus

Der genetische Algorithmus soll hier nicht im Fokus stehen und wird demnach nicht erläutert. Für Fachkundige sei gesagt das er die Diversität der Population berücksichtigt und entsprechend die Mutationsrate verändert. Crossover und Mutation findet auf der Genklasse „RotorTimeMutant“ statt. Sie beschreibt wann um wie viel der Rotor angewandt wird. Die Periode der Bewegung wird in x Teile segmentiert und jedem Segment wird eine Zahl zwischen 0 und 1 zugewiesen. Crossover wird irgendwo zwischen den segmentierten Teilen stattfinden. Mutation ist auf jedes Segment bezogen. Das heisst es kann pro Gen nicht mutieren oder gar mehrere zur gleichen Zeit. Für nähere Informationen zu diesem Algorithmus finden Sie den Code auf GitHub.

https://github.com/Nethersoul/Genetische-Algorithmen-geometrische-Algebra.git

3 Ergebnis

Um eine gute Visualisierung umzusetzen, wurde Unity verwendet. Unity hat eine integrierte Physikengine die das Laufverhalten simulieren kann. Daher müssen nur die Rotationen der Beine berechnet werden. Abhängig von der Reibung auf dem Untergrund wird sich der Roboter vorwärts bewegen. Um eine gute Einsicht in das Verhalten des Algorithmus und das Fortbewegungsverhalten zu gewinnen wurden mehrere Roboter in die Szene gesetzt. Alle Roboter waren gleich aufgebaut und führten parallel das selbe Script aus. Durch die unterschiedliche Rotation der Beine verhält sich jeder Roboter anders und legt eine andere Distanz zurück. Durch Optimierung der Rotationen der Gelenke mit Hilfe des Genetischen Algorithmus ändert sich die zurückgelegte Distanz. Die vier Roboter die im vorhergehenden Zyklus die grösste Distanz zurücklegten, werden im nächsten Zyklus zur besseren Überschaubarkeit auf eine Seite der Szene gesetzt (Elitismus).
Die Bewegung der Beine wurde mit Hilfe eines Riggs innerhalb des Roboters umgesetzt. Dadurch konnte die Rotoren direkt auf die Knochen der Beine angewendet werden.
Da in Unity die Rotation von Objekten mit Hilfe von Quaternionen umgesetzt wird, ist es nötig die berechneten Rotoren in Quaternionen umzuwandeln.

Es ist nötig den Algorithmus einige Zyklen zu durchlaufen bis ein Resultat sichtbar wird. Es zeigt sich aber nach einiger Zeit eindeutig eine Optimierung des Fortbewegungsverhalten. Nach anfänglichen Schwierigkeiten werden die Roboter nach einigen Zyklen eine grössere Distanz zurücklegen. Es deutet deutlich daraufhin, dass die Optimierung durch den Gentischen Algoritmus funktioniert. Ebenso ist deutlich zu sehen dass die Mutation der Rotoren unterschiedliche Geh-bewegungen bewirkt. Allerdings kann es durch die Physikengine von Unity, manchmal zu ungewolltem verhalten der Roboter kommen. Da die Fortbewegung durch die Reibung der Beine auf einer Oberfläche berechnet wird kann es zu unterschiedlichen Distanzen führen und die Voraussetzung der Beständigkeit verletzen. Die gleiche Art von Fortbewegung kann daher in zwei Läufen unterschiedliche Fitnesswerte produzieren

4 Schlussfolgerungen/Fazit

Unity ist für die Umsetzung von Geometrischer Algebra nicht geeignet. Durch die Umrechnung des Rotors in Quaternionen ist wird die Geometrische Algebra überflüssig. Zur direkten Anwendung der Rotoren sollten diese direkt auf direkt auf die Vertices des Meshes angewendet werden. In Unity können die Positionen der Vertices manipuliert werden jedoch wird sich die Weltposition des gesamten Meshes oder Objektes nicht an die Position der Verices anpassen. Die Physikengine kann nur die Position des Objektes berücksichtigen welches zu fehlerhaften Resultaten führen würde.
Die Kombination von Genetischen Algorithmen und Geometrischen Algebra ist eine gute Lösung um dem Roboter das Gehen bei zu bringen. Jedoch ist die Performance des Genetischen Algorithmus ein Problem. Es muss bei der Umsetzung auch darauf geachtet werden dass der Genetische Algorithmus nicht in eine Zufallssuche degeneriert. Dazu muss das Crossover Verfahren und die Mutationsrate angepasst werden. Eine weitere Möglichkeit wäre die Umsetzung eines zweibeinigen, aufrecht gehenden Roboters der die Möglichkeit hat sich aus zu balancieren.
Die Resultate sind zufriedenstellend, jedoch muss der Algorithmus einige Zyklen durchlaufen bevor starke Änderungen bemerkbar werden. Ausserdem sollte vermutlich auf eine stärkere Physikengine gewechselt werden, da Unity noch stark in der Entwicklung ist und die Physikengine noch einige Probleme bereitet. Eine optimierte Berechnung der Kollisionen über die Grafikkarte wäre wünschenswert da dies der Flaschenhals des Systems ist.

5 Abbildungsverzeichnis

6 Literaturverzeichnis

Dorst, 2007
Leo, Dorst, Daniel Fontijne, Stephen Mann, GEOMETRIC ALGEBRA for Computer Science, Morgan Kaufmann, Burlington USA, 2007

Geijtenbeek et al., 2013
Thomas Geijtenbeek, Michiel van de Panne, A.Frank van der Stappen, Flexible Muscle-Based Locomotion for Bipedal Creatures, ACM Transactions on Graphics, 32, 6, 2013
http://goatstream.com/research/papers/SA2013/

GAIGEN 2.5
http://g25.sourceforge.net/

7 Selbständigkeitserklärung

Ich bestätige, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt habe. Sämtliche Textstellen, die nicht von mir stammen, sind als Zitate gekennzeichnet und mit dem genauen Hinweis auf ihre Herkunft versehen.