**ENGLISH**

Superquadrics-augmented Rapidly-exploring Random Trees

This thesis work investigated the advantages and disadvantages of using superquadrics (SQ) to do the collision-checking part of the Rapidly-exploring Random Trees (RRT) motion planning algorithm for higher Degree of Freedom (DoF) motion planning, comparing it with an established proximity querying method known as the Gilbert-Johnson-Keerthi (GJK) algorithm. In the RRT algorithm, collision detection is the main bottleneck, making this topic interesting to research.

The SQ-based collision detection method was compared to the GJK algorithm both qualitatively and quantitatively, comparing computational speed, memory requirements, as well as the ability to handle arbitrary shapes. Furthermore, how appropriate they are in modelling a 6 DoF arm was analyzed. A qualitative comparison between the RRT algorithm and the A\* algorithm was also been provided, comparing their suitability for searching in higher dimensional spaces.

The SQ-based collision detection method was implemented in two versions: Obverse-SQ (OSQ) and Reverse-SQ (RSQ). When there were no collisions the SQ-based algorithms performed roughly at parity with the GJK algorithm in terms of computational speed. However, when a collision had occurred, the SQ-based algorithms were able to return a positive faster than the GJK algorithm, outperforming it. Furthermore, from a memory point-of-view the SQ-based algorithms required less memory as they could leverage the explicit and implicit representations of the SQ objects, whereas the GJK algorithm requires both objects being checked for collision to be explicitly represented as convex sets of points.

In regards to the ability to handle arbitrary shapes, the SQ-based algorithms have an advantage in that they can allow for non-convex shapes to be checked (including e.g. donut-shaped supertoroids). Conversely, the GJK algorithm is limited to convex shapes. To compensate for this, the GJK algorithm would require more geometric primitives to accurately capture the same non-convex shape, increasing the computational burden. Thus, it can be concluded that the SQ-based method is more suitable for modelling a 6 DoF arm. However, despite its faults, a GJK-based collision detection module would in most cases be a lot more straightforward than the alternative to set up, as it is very simple to collect a set of points.

Finally, both collision detection method types were implemented with the RRT algorithm. Unfortunately, due to the inherently random nature of the RRT algorithm the results of this set of tests could not be used to make any further conclusions beyond showing that it is possible to combine the SQ-based algorithm with the RRT algorithm. Instead, one should see the RRT algorithm as a multiplicative factor applied to the inherent properties of the previously examined collision detection methods.

**SWEDISH**

Raskt-utforskande Slumpmässiga Träd med N:tegradsytor

Detta examensarbete undersökte fördelarna och nackdelarna med att använda n:tegradsytor (NY) för att utföra kollisionsdetektion i rörelseplanneringsalgorithmen Raskt-utforskande Slumpmässiga Träd (RST). RST används typiskt för planeringen av system med relativt många frihetsgrader. En etablerad metod för kollisionsdetektion, Gilbert-Johnson-Keerthi-algoritmen (GJK), implementerades även i jämförelsesyfte. Eftersom att GJK-algoritmens största flaskhals ligger i kollisionsdetektionen är detta ett intressant ämne att efterforska.

Den NY-baserade kollinsdetektionsmetoden jämfördes med den GJK-baserade metoden både kvantitativt och kvalitativt. Kvalitativt jämfördes beräkningshastighet och minnesåtagande, medan de kvalitativt jämfördes i deras förmåga att representera godtyckliga geometriska former. PÅ ett högre plan diskuterades det även hur lämpliga de är för att modellera en robotarm med 6 stycken frihetsgrader. RST-algoritmen jämfördes även med en annan planeringsalgoritm, A\*. Framförallt fokuserade diskussionen kring planering av system med relativt många frihetsgrader.

Två versioner av den NY-baserade metoden implementerades: framvänd-NY och bakvänd-NY. I det fall inga kollisioner fanns presterade GJK-algoritmen ungefär lika bra som ONY och RNY algoritmerna i att fastslå detta, utifrån beräkningshastighet. Men när det kom till att upptäcka existerande kollisioner presterade GJk-algoritmen sämre. Minnesmässigt använder GJK-algoritmen mer minne, då den kräver att båda objekten är explicitrepresenterade (dvs, som ett punktmoln), medan man med en NY-metod endast behöver representera ena objektet explicit och den andra implicit.

Gällande förmågan att representera godtyckliga geometriska former är den NY-baserade metoden överlägsen. Till skillnad från GJK som är begränsad till konvexa mängder, NY kan uppta icke-konvexa former, exempelvis flottyrmunkformade supertoroider. En metod som använder GJK-algoritmen skulle behöva bygga upp icke-konvexa former med flera mindre konvexa komponenter, vilket ökar beräkningsbördan. NY-metoden är därför överlägsen när det kommer till att modellera robotarmar med 6 frihetsgrader. Det är dock i praktiken lättare att implementera GJK-metoden då den endast kräver punktmoln, medan NY kräver parametrar som måste bestämmas eller finjusteras.

RST-algoritmen implementerades sist, utformad så att kollisionsdetektionsmetoderna är utbytbara. Det var dock inte möjligt att dra slutsatser utifrån det testdata som erhölls, ty RST-algoritmens slumpmässiga karaktär. RST-algoritmen kan ses som en multiplikativ faktor som endast förstorar de inneboende egenskaperna hos kollisionsdetektionsmetoderna, vars fördelar och nackdelar redan redovisats. Det visades i alla fall att NY kunde kombineras med RST.