Slide 1

Il mio progetto si inquadra nel path planning che consiste nel trovare una curva, con certe caratteristiche desiderate, da un punto ad un altro punto nello spazio, evitando intersezioni con degli ostacoli. Nel mio lavoro mi sono concentrato nello spazio tridimensionale, usando uno specifico tipo di curva: le curve B-spline.

Per capire i metodi implementati è necessario affrontare prima due argomenti.

Slide 2

I diagrammi di Voronoi sono ricavati a partire da un insieme di punti di input nel piano (o nello spazio come nel mio progetto) chiamati siti. Successivamente viene creata una partizione dello spazio in regioni tale che ogni regione è costituita dall'insieme di punti più vicini ad un certo sito di input rispetto agli altri. L'insieme di queste regioni si chiama diagramma di Voronoi. Notare che alcune celle hanno superficie infinita.

Un altro concetto importante è quello di curva B-spline.

Slide 3

Esse sono curve parametriche polinomiali a tratti, definite in un certo dominio parametrico $[a, b]$, per un certo grado m, usando una specifica base detta base delle B-spline. Ogni curva è identificata da un insieme di punti detti vertici di controllo, i quali formano il poligono di controllo.

Queste curve hanno una regolarità prescritta che dipende dal grado, seguono circa il percorso del poligono di controllo e con certi accorgimenti possono interpolare gli estremi del poligono di controllo.

i diagrammi di Voronoi possono essere usati per ottenere una struttura su cui identificare un percorso libero da ostacoli.

Slide 4

Partendo dalla scena vuota si distribuiscono uniformemente dei punti sulle superfici degli ostacoli, e anche sulla superficie di una bounding box che racchiude la scena. In seguito si usano tali punti come siti di input per ricavare il diagramma di Voronoi, e si interpreta questi come un grafo dove i nodi sono i vertici delle celle di Voronoi e gli archi sono i lati delle celle di Voronoi pesati con la lunghezza geometrica dei lati. I lati infiniti delle celle infinite vengono ignorati. Infine c'è una fase di pruning dove vengono eliminati tutti gli archi che attraversano un ostacolo. Si ottiene una struttura che abbraccia gli ostacoli come una specie di ragnatela, e su questa struttura è possibile attaccare con qualche regola i punti di partenza e di arrivo desiderati e poi usare un algoritmo di routing come quello di Dijkstra per trovare un cammino ammissibile tra questi due punti.

Con questa operazione si ottiene un cammino che è costituito da una poligonale spezzata.

Slide 5

Però è più interessante avere una curva smooth. L'idea è stata quella di usare una curva B-spline in modo tale che essa interpoli il punto di partenza e il punto di arrivo e che usi la poligonale spezzata come poligono di controllo. La scelta è ricaduta sulle B-spline in quanto sono uno standard affermato nel mondo del CAD.

In questo modo però sorge un problema.

Slide 6

Infatti il poligono di controllo è libero da ostacoli per costruzione, perché abbiamo fatto il pruning di tutti gli archi che intersecano gli ostacole. Però non è detto che la curva ottenuta sia anche essa libera da ostacoli, come si può vedere nel semplice esempio in due dimensioni.

Per risolvere questo problema ho quindi pensato ad una soluzione che sfrutta una proprietà delle B-spline.

Slide 7

Una curva B-spline di grado m è contenuta dentro l'unione dei convex hull formati da $m + 1$ vertici di controllo consecutivi nel poligono di controllo. Ad esempio per una B-spline quadratica l'area in cui è contenuta la curva è costituita dall'unione di tutti i triangoli di vertici consecutivi sul poligono di controllo. Notare che questo vale anche nello spazio.

Quindi per risolvere il problema della collisione della curva possiamo usare una B-spline quadratica e far si che l'area definita da questa regola sia libera da ostacoli. Per questo ultimo punto in particolare ho sviluppato due soluzioni.

Slide 8

La prima soluzione consiste nell'applicare una trasformazione al grafo ottenuto dal diagramma di Voronoi. Chiamando il grafo originale G e il grafo trasformato G_t, si ha che ogni tripla (a, b, c) di nodi adiacenti in G costituisce un nodo di G_t, e c'è un arco tra due nodi in G_t se condividono un arco in G ad esempio come nel caso di (a, b, c) e (b, c, d). Tale arco è pesato con la distanza tra i primi due nodi in G, nell'esempio a e b. Notare che mentre G non è direzionato, G_t lo è, in generale se c'è un arco tra (a, b, c) e (b, c, d) non c'è un arco tra (b, c, d) e (a, b, c).

Dopo vi è una fase di pruning dove si eliminano tutte i vertici di G_t i cui triangoli corrispondenti intersecano un ostacolo. Infine viene calcolato il cammino minimo con Dijkstra in G_t, ricavando poi i nodi in G di questo.

Guardando l'esempio schematico si ha che G è costituito dai nodi d, a, b, etc.. uniti come in figura. G_t è costituito dalle triple (d, a, b), (b, a, d), (a, b, c), etc.. e nella fase di pruning vengono eliminate le triple (a, b, c) e (c, b, a) perché intersecano l'ostacolo. Eseguendo Dijkstra da una tripla speciale di inizio collegata alle triple che contengono il nodo iniziale a una speciale di fine collegata a quelle che contengono il nodo finale si ottiene il cammino costituito dalle triple: (d, a, b), (a, b, e), (b, e, c), (e, c, f); da cui si può ricavare il cammino in G costituito dai nodi: d, a, b, e, c, f.

Applicando una B-spline quadratica ad un cammino ottenuto in questo modo abbiamo la garanzia che questa non intersechi nessun ostacolo. Ho calcolato anche la complessità di questo metodo,

Slide 9

che possiamo vedere riassunta in questa tabella, dove O è l'insieme degli ostacoli e k è il grado massimo in G. Escludendo casi limite, ci sono delle regole dei diagrammi di Voronoi che permettono di considerare il grado massimo costante e quindi è possibile semplificare la complessità.

Dalla tabella si evince che il costo maggiore si ha per le fasi di pruning, dovuto al controllo delle collisioni, inoltre è possibile dividere l'algoritmo in due parti: una fase di creazione della scena con complessità quadratica, e poi usare la stessa scena per calcolare curve diverse con una complessità ridotta a $\mathcal{O}(n \log n)$ che è la stessa di Dijkstra in un grafo con grado massimo costante.

Slide 10

Questa implementazione è interessante da un punto di vista algoritmico, però ha il difetto di scartare diversi percorsi, con addirittura la possibilità di rendere G_t non connesso. Quindi ho realizzato una seconda soluzione al problema in cui si calcola il cammino con Dijkstra direttamente in G. In seguito c'è una fase di controllo delle collisioni sul cammino trovato, e se una tripla del cammino interseca un ostacolo si aggiungono vertici allineati in modo da rendere tutti i triangoli, degenerati e no, liberi da collisioni. Inoltre le coordinate di questi vertici allineati sono facilmente ottenute una volta fatto il controllo di collisione. Anche di questa soluzione ho calcolato la complessità,

Slide 11

riassunta in questa tabella. Oltre all'insieme di ostacoli e al grado massimo è presente l'insieme P dei vertici del poligono di controllo, che nel caso pessimo ha dimensione pari a $\mathcal{O}(|O|)$.

Anche per questo metodo è possibile considerare una fase di costruzione della scena con complessità quadratica, e una fase di esecuzione sulla scena con complessità maggiore rispetto alla soluzione precedente, dovuta al termine $|P|\ |O|$ per il controllo delle collisioni nel cammino trovato su G.

Slide 12

L'uso di B-spline quadratiche ha l'inconveniente che globalmente la curva ha continuità C^1, ossia è continua fino alla derivata prima. In alcune applicazioni questo non è sufficiente, se ad esempio vogliamo usare la curva come supporto per il moto di un oggetto vogliamo continuità almeno fino alla derivata seconda.

Se manteniamo il poligono di controllo invariato e aumentiamo il grado della curva si ha il problema che la proprietà di convex hull non vale più come prima, l'area in cui si può trovare la curva non è più unione di triangoli ma unione di tetraedri o comunque figure solide. In questo modo non possiamo più fare il controllo di collisione con i metodi descritti prima. La soluzione che ho trovato consiste nell'aggiungere vertici allineati nel poligono di controllo in modo da non dover modificare i metodi visti prima. Per descrivere il modo in cui vengono aggiunti questi nuovi vertici consideriamo un esempio.

Slide 13

Il poligono di controllo iniziale è costituito dai vertici in verde, e l'area in cui si può trovare una B-spline quadratica è evidenziata in celeste. Se vogliamo aumentare il grado a 4 è necessario aggiungere due vertici per segmento. In questo modo e possibile aumentare il grado perché l'area in cui si può trovare una curva quartica è costituita dall'unione di convex hull di 5 vertici consecutivi, e in questo caso sono tutti triangoli contenuti nell'area precedente. Se precedentemente avevamo usato uno dei due metodi visti per garantire la non collisione della curva con gli ostacoli, a maggior ragione la curva quartica così ottenuta sarà libera da collisioni.

Oltre all'incremento di grado, ho previsto anche una fase opzionale di post processing.

Slide 14

Con lo scopo di semplificare il poligono di controllo eliminando i vertici inutili, e quindi evitando curve inutili nella B-spline.

Questa fase consiste nel considerare tutte le triple di vertici consecutivi nel poligono di controllo, e, se il triangolo corrispondente non interseca nessun ostacolo, allora si semplifica la tripla eliminando il vertice centrale. Prima di fare questa semplificazione però, è necessario controllare che la modifica non influisca negativamente sulle triple adiacenti.

Infine, oltre agli algoritmi visti, ho implementato anche una soluzione che consiste in un metodo di ottimizzazione stocastico.

Slide 15

Il problema del path planning può essere visto come un problema di ottimizzazione vincolata, dove si deve minimizzare una grandezza di interesse al variare della posizione dei vertici di controllo, con il vincolo che la curva ottenuta sia libera da collisioni con gli ostacoli. La grandezza di interesse è una combinazione di lunghezza della curva, picco di curvatura e picco di torsione.

Per risolvere questo tipo di problemi si può usare la cosiddetta lagrangian relaxation, che consiste nel rilassare il vincolo aumentando la dimensione dello spazio degli stati, e creando una superficie così definita dove gain è la grandezza di interesse e constraint è di quanto la curva interseca gli ostacoli. Su tale superfice si trova il minimo desiderato tra i punti di sella usando un metodo di ottimizzazione come il simulated annealing.

I drawback di questo metodo sono la lentezza di esecuzione e il fatto che sia il calcolo della quantità da minimizzare che del vincolo sono effettuati in modo discreto, quindi lasciando spazio ad errori in casi particolari.

Per sviluppare gli algoritmi visti ho realizzato una applicazione object oriented.

Slide 16

In python 3, usando la libreria SciPy per i calcoli vettoriali, di algebra lineare e il calcolo delle curve B-spline; la libreria NetworkX per la rappresentazione di grafi e le operazioni su essi; e VTK per l'interfaccia grafica.

E ho usato l'applicazione per eseguire diversi test di cui evidenzio

Slide 17

Una esecuzione del primo metodo senza post processing. Una esecuzione del secondo metodo sempre senza post processing in cui si nota che la curva segue un percorso che il primo metodo aveva scartato. Una esecuzione del primo metodo questa volta con il post processing di cui si può apprezzare l'efficacia in termini di semplificazione della curva. Una esecuzione del secondo metodo con il post processing. E infine una esecuzione del metodo di ottimizzazione stocastico dove si può apprezzare la minore lunghezza della curva.

Alcuni possibili sviluppi futuri possono essere:

Slide 18

Cambiare la struttura di base, ad esempio cambiando il modo in cui si attaccano i punti di partenza e arrivo per evitare ganci indesiderati, oppure usare un altro metodo invece di quello basato sui diagrammi di Voronoi, ad esempio usando il visibility graph o gli RRT o altri.

Un altro miglioramento può riguardare l'aumento di grado, l'uso di vertici allineati infatti ha lo svantaggio di forzare la curva a fare dei salti di torsione, una soluzione può essere quella ad esempio di considerare una soluzione tipo il metodo 2 dove però si controllano non più i triangoli nel poligono di controllo, ma i poliedri di 4 o 5 vertici consecutivi.

È possibile anche migliorare la fase di post processing in quanto l'attuale ha lo svantaggio di non essere simmetrica. Se si trova una curva da un punto a ad un punto b in generale sarà diversa dalla curva trovata dal punto b al punto a.

È possibile anche concentrarsi su altri metodi di ottimizzazione, ad esempio usando come stato iniziale del sistema una soluzione ammissibile ottenuta con uno dei due metodi proposti, e facendo evolvere il sistema solo in stati ammissibili.

Slide 19

Con questo ho concluso, se ci sono domande, grazie.