ANALISI DEL PROBLEMA

Il sistema è composto da due entità:

* Console
* Robot

esse devono interagire tra di loro per garantire le funzionalità richieste.

Date le specifiche tecniche, possiamo dedurre che il sistema è Eterogeneo e Distribuito. Non abbiamo alcun vincolo sulla posizione fisica delle due, serve quindi una infrastruttura di comunicazione che garantisca una interazione stabile e funzionale.

Scelgo quindi di utilizzare il framework QActor sviluppato da AN.

I requisiti richiedono di sviluppare due funzionalità principali:

* Navigazione reattiva da un punto START a un punto GOAL, conoscendo la mappa dell’ambiente;
* Manipolazione dell’ambiente.

NAVIGAZIONE

La fase di navigazione è forse la più problematica a causa della necessità di elaborare un piano ottimale. Per farlo è necessario utilizzare tecniche e algoritmi di intelligenza artificiale che permettano di trovare il miglior percorso tra due punti.

Una scelta saggia è l’algoritmo A\* nella sua versione basilare.

Da requisiti sappiamo che la Console deve essere implementata su sistemi Windows e/o Andorid, mentre il Robot su sistema Rasberry Pi.

La Console avrà sicuramente più capacità di elaborazione rispetto al Robot, di conseguenza scelgo di delegare ad essa il pesante onere di ricercare il percorso migliore.

E comunque necessario inserire la capacità di ricerca anche nel Robot, allo scopo di soddisfare il requisito di reattività della Navigazione.

ASSUNZIONE 0: L’Ambiente di lavoro è pressoché statico.

Di conseguenza è improbabile che avvengano frequenti cambiamenti nell’ambiente, quindi le probabilità che il Robot debba calcolare un percorso sub-ottimo sono basse.

GESTIONE DELLA MAPPA

La Mappa deve essere condivisa tra Console e Robot.

Nel normale funzionamento, l’utente carica la Mappa da file quindi richiede la Navigazione.

Come suddetto è la console a stabilire il percorso ottimale da seguire. Questo deve essere inviato al Robot per poter iniziare la Navigazione.

Non è fondamentale trasmettere al robot la mappa.

I motivi di tale scelta sono 2:

1. Mentre i comandi di navigazione possono essere espressi con delle lettere secondo una certa codifica, per la mappa è necessario mantenere un formalismo più espressivo; Di conseguenza trasferire una mappa di grandi dimensioni richiederebbe un tempo non trascurabile;
2. Il Robot non ha reale necessità di conoscere la mappa. Se il movimento è ben fatto sarà in grado di aggiornare la propria posizione durante la navigazione, di conseguenza tutto ciò di cui ha bisogno è la posizione iniziale relativa.

ASSUNZIONE 1: La Mappa è suddivisa in celle quadrate di dimensione unitaria. La dimensione è definita dalle dimensioni del Robot. Di conseguenza la Navigazione è discretizzata.

ASSUNZIONE 2: La Mappa può essere inclusa o meno in un anello di ostacoli (confini) che definiscono l’area calpestabile. In ogni caso deve essere inclusa in un quadrato o rettangolo.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

X

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

QUESTE SONO MAPPE VALIDE

Y

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

QUESTA MAPPA è INVALIDA

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

MA è FACILMENTE CORREGGIBILE

La Mappa è salvato in un formato semplice e condiviso. Per comodità scelgo di utilizzare il linguaggio PROLOG.

Allo scopo di minimizzare la quantità di informazioni in mappe molto grandi, scelgo di memorizzare solo gli elementi da evitare (ostacoli o bordi) della mappa.

Inoltre Assumo come riferimento l’angolo in basso a sinistra.

L’ultima mappa può quindi essere rappresentata come:

map(10, 10).

element(0,0).

element(1,0).

element(2,0).

element(3,0).

element(4,0).

element(5,0).

element(6,0).

element(7,0).

element(8,0).

element(9,0).

element(10,0).

element(0,1).

element(0,2).

element(0,3).

element(0,4).

element(0,5).

element(0,6).

element(0,7).

element(0,8).

element(0,9).

element(0,10).

element(10,1).

element(10,2).

element(10,3).

element(10,4).

element(10,5).

element(10,6).

element(10,7).

element(10,8).

element(10,9).

element(10,10).

element(9,1).

element(9,2).

element(9,3).

element(9,4).

element(9,5).

element(9,6).

element(9,7).

element(9,8).

element(9,9).

element(9,10).

element(1,10).

element(2,10).

element(3,10).

element(4,10).

element(5,10).

element(6,10).

element(7,10).

element(8,10).

element(9,10).

element(3,4).

element(6,4).

element(6,5).

element(6,6).

element(6,7).

element(3,7).

element(4,7).

element(5,7).

MAPPA DINAMICA

Fino ad ora abbiamo assunto la mappa come statica. Le specifiche ci impongono però la possibile dinamicità.

“Se il robot incontra un ostacolo non previsto durante la navigazione, deve considerarlo come dinamico, attendere un tempo predefinito ed effettuare un nuovo rilevamento. Se l’ostacolo non è più presente può proseguire con il piano normale,

Se l’ostacolo è ancora presente deve assumerlo come statico ed evitarlo”

Dal momento che il raspberry non è molto potente in termini di capacità di calcolo, e soprattutto come suddetto non è a conoscenza della mappa totale, scelgo di delegare ancora una volta il compito di elaborare il piano alternativo.

Quando il Robot rileva l’ostacolo statico comunica alla Console la posizione del nuovo ostacolo e la propria posizione.

La Console si occupa di aggiornare la mappa e calcolare un nuovo piano alternativo verso il Goal.

RILEVARE OSTACOLI

Il problema di quando rilevare un ostacolo non è banale.

Sicuramente il Robot NON sarà sensibile agli ostacoli durante una rotazione, dal momento che si sta muovendo su se stesso (almeno idealmente).

Bisogna discutere il sensing durante il movimento in avanti.

Possiamo assumere il sonar come sempre attivo, un dispositivo che spamma di continuo eventi di oggetto rilevato.

La domanda è quando ascoltarli??

* Possiamo ascoltare gli eventi durante il movimento, a questo punto è sufficiente essere sensibili a una breve distanza ma, se un ostacolo si trova a metà cella, il robot si fermerà e qualsiasi tipo di conoscenza della posizione andrà a puttane (a meno di non tornare indietro di quel tanto che serve);
* Un modo più semplice è porre prima di ogni movimento in avanti un’azione di sensing. La distanza di rilevamento deve coincidere con lo step di movimento del robot, in questo modo potrò sapere se nella prossima cella è presente un ostacolo e nel caso rielaborare.

Personalmente preferisco la seconda possibilità, leggermente pessimista ma molto più precisa e di facile implementazione. Rispecchia inoltre il normale comportamento di un essere senziente. (Non vado avanti se so che dopo un passo devo tornare indietro).