Università di Pisa
Facoltà di Ingegneria
Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea

Sviluppo e simulazione di algoritmi di coordinamento per velivoli autonomi

Candidato:

Indri Muska
Matricola 426534

Relatori:

Prof. Giorgio Buttazzo
Dott. Mauro Marinoni

Anno Accademico 2012/2013

Ai miei genitori.

Indice

Abstract

Obiettivo della tesi. Il lavoro svolto durante questa tesi di laurea si concentra nel definire una serie di regole pratiche, che permettano ad una squadra di velivoli autonomi o a pilotaggio remoto di attuare una strategia di coordinamento distribuito per raggiungere un obiettivo comune. In particolare, poiché i singoli sistemi inviano e ricevono i messaggi attraverso l'etere, lo scopo principale di quest'analisi è rappresentato proprio dal mantenimento della connettività fra i nodi durante il movimento.

Metodologia utilizzata. Partendo da soluzioni già esistenti in letteratura, è stata concretizzata una tecnica organizzativa, valida per ogni unità della rete, che si sviluppa attraverso lo scambio di un numero relativamente basso di messaggi fra i nodi. Il fattore che diventa preponderante, nonché essenziale durante il movimento, è la conoscenza della posizione dei nodi vicini nella particolare struttura topologica scelta, poiché in base ad essa viene determinata la *zona di movimento possibile*.

Contributi innovativi. Diversamente dallo stato dell'arte analizzato, è stato approfondito il caso d'uso più generale di un'architettura organizzata secondo il paradigma *Leader/Follower* in cui sono presenti più di due leader. Inoltre, è stato realizzato un sistema di *collision detection/collision avoidance*, con lo scopo di rilevare tutti gli oggetti che ostacolano il passaggio dei droni e di modificare la traiettoria degli stessi.

Risultati ottenuti. Le regole così definite hanno portato alla nascita di un simulatore detto *Distributed UAVs Coordinator*, un applicato web-grafico che

visualizza lo stato della rete durante la sua evoluzione temporale. Sebbene sia stato sviluppato in un linguaggio diverso da quello comunemente utilizzato per i *sistemi embedded*, esso rappresenta comunque una piattaforma di lancio ideale per rendere il programma direttamente fruibile su sistemi mobili autonomi. Tale caratteristica potrà essere realizzata anche grazie al *design modulare* del software, nel quale si è cercato di mantenere il più possibile indipendenti i singoli blocchi funzionali.

Struttura della tesi. Nel capitolo 1 vengono descritte le caratteristiche e i vantaggi nell'uso dei *sistemi multi-robot* e dei quadricotteri in generale. Vengono anche presentate le linee generali del problema affrontato, seguite dallo studio dello stato dell'arte e da una piccola analisi della soluzione proposta.

Nel capitolo 2 sono presentate le specifiche dell'architettura di rete adottata, insieme ad una breve introduzione generale sulle *mappe auto-organizzanti* da cui si è preso spunto. Vengono altresì delineate le classi generali in cui sono suddivisi i robot ed il compito che ognuno di essi deve attuare, concludendo con un'analisi sul modello cinematico discreto utilizzato.

Nel capitolo 3 si definiscono precisamente tutte le fasi dell'algoritmo di coordinamento proposto: *inizializzazione*, *scelta dei master*, *avvio dei master* e *strategia di movimento*. Viene anche illustrata la composizione esatta dei messaggi scambiati fra i nodi e delle variabili di stato che identificano univocamente ciascun velivolo.

Nel capitolo 4 si analizza il problema del rilevamento degli ostacoli e della loro elusione. Nel modello proposto si fa uso di una dotazione sensoriale aggiuntiva composta da sensori di prossimità ultrasonici; tale modello, tuttavia, può essere perfettamente sostituito con altri sistemi a scelta dell'utente, grazie alla facilità di integrazione permessa dalla strategia di coordinamento proposta.

Nel capitolo 5 viene presentato l'ambiente di simulazione e di testing *Distributed UAVs Coordinator* (DUC) ed illustrate tutte le possibili funzionalità di cui è dotato. All'inizio del capitolo si descrive anche il modello concettuale che riproduce le caratteristiche di un punto sulla superficie terrestre ed i metodi necessari a

determinare alcuni parametri geografici fondamentali (come la *distanza*, la *rotta* e il *punto medio*).

L'ultimo capitolo ripercorre brevemente le fasi della ricerca dell'algoritmo di coordinamento, prendendo in esame i limiti della strategia e del simulatore DUC, ed aprendo lo scenario ad una serie di possibili implementazione future.

Capitolo 1

Introduzione

All'interno della robotica e dell'automazione moderna, il concetto di *coordinamento* fra unità mobili indipendenti ha da sempre occupato una posizione di rilievo e di grande importanza. Il motivo è semplice ed è legato alle prestazioni, alla robustezza e all'affidabilità che la cooperazione assicura in qualunque ambito la si applichi.

La cooperazione si definisce come il processo mediante il quale le componenti di un sistema lavorano insieme per portare a termine un obiettivo comune. Le singole unità operano autonomamente per creare un organismo più complesso ed intelligente. In natura, si trovano esempi di coordinamento praticamente ovunque, dalle formiche che lavorano insieme per trascinare nel formicaio oggetti più grandi di loro, alle api che si coordinano e cooperano per edificare l'alveare, fino al coordinamento umano, anche fra culture diverse, per favorire l'arte, lo scambio di beni, o semplicemente per portare avanti uno spirito solidaristico. Ogni qualvolta gli esseri viventi cooperano in qualche modo, in generale i risultati sono più funzionali e vantaggiosi che nei casi di azioni solitarie.

Nell'ambito dei **sistemi multi-robot**, si cerca di garantire queste prerogative ad *agenti autonomi*[1] in grado di muoversi e di comunicare in maniera indipendente fra loro al fine di portare a termine un particolare compito. La comunicazione,

[1]Un agente intelligente è un'entità autonoma che per mezzo di sensori ed attuatori riesce ad eseguire le operazioni necessarie a raggiungere uno scopo prestabilito attraverso le rilevazioni dell'ambiente che lo circonda [**1**].

in questo caso, assume un ruolo fondamentale e praticamente imprescindibile rispetto ad altri compiti svolti dagli agenti. Ciascuno di essi, infatti, può essere munito di una serie di sensori, di attuatori o di funzionalità non presenti in altri robot, che rende disponibili al **team**, andando ad aumentare le capacità totali del sistema nella sua interezza. Affinché tutto si muova nel verso giusto, però, è necessaria la divulgazione di informazioni relative allo stato del sistema ogni qualvolta questo viene aggiornato.

> *An agent is a computer system that is situated in some environment and is capable of autonomous action in this environment in order to meet its delegated objectives* [**2**].

In breve, i principali vantaggi nell'avere una squadra di agenti sono:

- la risoluzione in parallelo di sotto-problemi più semplici, con una conseguente riduzione del *tempo di completamento* del compito globale del sistema e del *costo computazionale*;
- l'attuazione di una sensoristica distribuita che permette di ottenere una *complessità* più bassa a livello di singolo agente e quindi di definire una serie di robot *più semplici* e *meno costosi* rispetto ad un unico robot polifunzionale complesso;
- un *incremento di flessibilità*;
- la *tolleranza ai guasti* e la *maggior robustezza* del sistema complesso, grazie all'assenza di un singolo centro decisionale.

1.1 Quadricotteri

Gli agenti a cui si fa riferimento nel seguito della trattazione sono per lo più *quadricotteri*, anche se i concetti sono perfettamente validi per qualunque sistema robotico mobile, sia esso volante o meno.

Un quadricottero (o *elicottero quadrirotore*, o semplicemente *quadrirotore*) è un aeromobile più pesante dell'aria, detto anche *giroplano*, nel quale la sostentazione

è ottenuta tramite una o più eliche in autorotazione, disposte con asse pressoché verticale [**3**].

A differenza degli elicotteri convenzionali, i quadricotteri hanno quattro rotori con pale a "passo fisso", il cui angolo d'attacco non varia durante la rotazione. Il controllo dei movimenti avviene attraverso la variazione della velocità relativa di ogni rotore, variando la spinta e la coppia prodotta da ciascuno di essi. In questo modo, si ottengono velivoli che non necessitano di collegamenti meccanici per variare l'angolo di attacco del rotore, semplificando la conformazione del velivolo e riducendo i tempi e i costi di manutenzione.

Figura 1.1: Un esempio di quadricottero: 3DR Iris.

Così come l'elicottero, il quadricottero appartiene alla categoria dei velivoli **VTOL** (***V****ertical* ***T****ake-****O****ff and* ***L****anding*), che a differenza dei comuni aeroplani "ad ala fissa" sono in grado di librarsi, decollare e atterrare in posizione verticale, senza necessità di spinta[2]. In particolare, i velivoli di interesse per questa trattazione sono detti **UAV** (*Unmanned Aerials Vehicle*), comunemente noti come *droni*. I droni sono *aeromobili a pilotaggio remoto* (APR, in inglese RPA) caratterizzati dall'assenza di pilota a bordo e controllati da un computer o da un sistema di

[2]In campo militare, sono presenti anche velivoli ad ala fissa con decollo e atterraggio verticale. In questo caso è più corretto chiamarli **V/STOL** (***V****ertical/****S****hort* ***T****ake-****O****ff and* ***L****anding*) in quanto sono in grado di librarsi sia in modalità convenzionale (***C****onventional* ***TOL***) che in modalità verticale. L'Harrier II, anche detto *jump jet*, è probabilmente l'aeromobile più famoso ad utilizzare questo sistema, in grado di atterrare e di decollare in modalità verticale direttamente su portaerei [**4**].

elaborazione interno. Il controllo può essere demandato a un pilota lontano dalla zona di volo o su un altro velivolo (allora si parla di *radiocomando*), oppure affidato a sistemi di elaborazione intelligente che ne definiscono il percorso e le operazioni da intraprendere. Sono noti anche attraverso altri acronimi, come RPA (*Remotely Piloted Aircraft*), RPV (*Remotely Piloted Vehicle*), ROA (*Remotely Operated Aircraft*), UVS (*Unmanned Vehicle System*).

Nonostante vengano comunemente associati ad ambiti militari, gli UAV sono sempre più presenti in applicazioni civili governative, aziendali e private. Fra le più conosciute, si possono elencare quelle di:

- Sicurezza territoriale, delle frontiere e lotta ai narcotrafficanti[3];
- Monitoraggio centrali termoelettriche e impianti industriali;
- Telerilevamento;
- Aerofotogrammetria e rilievo dell'architettura;
- Monitoraggio ambientale e calamità naturali[4];
- Biodiversità e monitoraggio fauna;
- Operazioni di ricerca e soccorso.

1.2 Problema generale affrontato

La progettazione e lo sviluppo di applicazioni robotiche distribuite richiede l'analisi preventiva di alcuni aspetti fondamentali relativi alla comunicazione, affinché un insieme di agenti mobili autonomi riesca a cooperare congiuntamente per conseguire l'obiettivo comune preposto.

Tenendo conto che il mezzo di comunicazione utilizzato per lo scambio di messaggi è generalmente l'etere, bisogna innanzitutto identificare la tipologia di

[3]Negli Stati Uniti le agenzie governative utilizzano UAV come l'MQ-9 Reaper per le sue proprietà esplorative, per pattugliare i confini della nazione, o per individuare latitanti [**5**].

[4]I droni T-Hawk [**6**] e Global Hawk [**7**] sono stati usati per raccogliere informazioni sulla centrale nucleare danneggiata di Fukushima e sulle aree colpite dalla catastrofe della regione di Tōhoku, dallo tsunami del marzo 2011 [**8**]. Gli attivisti contro la caccia alle balene hanno utilizzato l'Osprey [**9**] nel 2012 per monitorare le navi baleniere giapponesi in Antartide.

rete con cui vengono scambiati i messaggi fra i vari agenti. Le reti migliori per questo genere di applicazione sono le MANET (**M**obile **A**d-hoc **Net**work), ovvero un sistema di terminali autonomi con collegamenti senza fili che possono modificare la loro formazione topologica grazie alla mobilità degli apparati. Sono reti in cui tutti i nodi sono liberi di muoversi casualmente e auto-organizzarsi arbitrariamente, ma collaborano per il corretto instradamento dei pacchetti secondo la modalità di forwarding di tipo *multihop*.

L'*Internet Engineering Task Force* (IETF) ha creato un gruppo di lavoro per le MANET, il cui obiettivo è quello di *standardizzare un protocollo di routing unicast intra-dominio che reagisca in modo efficace ai cambiamenti topologici, pur mantenendo il routing efficace.*

> *A "mobile ad-hoc network" is an autonomous system of mobile routers (and associated hosts) connected by wireless links – the union of which form an arbitrary graph. The routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet* [**10**].

Il fatto che i vari nodi della rete si muovano formando un grafo di forma arbitraria rende le MANET ottimali per applicazioni robotiche distribuite, in cui ogni nodo della rete è identificato da un agente che comunica con gli altri.

Ovviamente, una soluzione di questo tipo porta anche ad una serie di inconvenienti. Innanzitutto, in base alla struttura topologica assunta dal *grafo diretto*[5], si ha il problema di definire il numero di messaggi che vengono scambiati fra i vari nodi. I grafi orientati possono essere di tre tipi:

- *Connessi:* dati due nodi qualsiasi esiste almeno un cammino che li collega;
- *Completi:* ogni nodo è connesso con tutti gli altri;
- *Totalmente sconnessi:* non presentano *archi*.

Escludendo i grafi totalmente sconnessi che risultano praticamente inutili ai fini della trattazione, la maggior parte delle applicazioni robotiche distribuite

[5]Un grafo diretto (o *digrafo*, o *grafo orientato*) è un grafo costituito da un insieme di nodi e da collegamenti orientati tra i nodi, detti *archi*.

è costituita da reti con grafi connessi. Questo perché rappresentano il miglior compromesso in termini di estensione superficiale rispetto al numero di messaggi da inviare. Difatti, nei digrafi completi, ogni nodo comunica con tutti gli altri, per cui il vantaggio di una comunicazione diretta fra nodi viene reso disponibile a discapito di una distanza massima di comunicazione pari dal raggio di copertura più piccolo fra tutti i segnali generati dai nodi stessi.

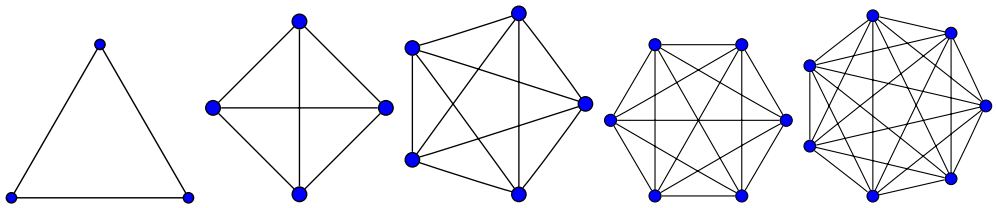


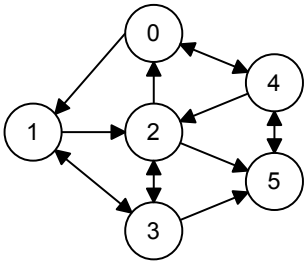
Figura 1.2: Grafi completi indiretti con numero di vertici da 3 a 7.

Nei grafi connessi, una coppia di nodi che non presenta archi di collegamento diretto, per poter comunicare, deve necessariamente sfruttare uno o più nodi intermedi della rete. Dato che l'instradamento dei pacchetti avviene per mezzo di quest'ultimi, risulta evidente come, con l'aumentare dei nodi, aumenti anche il numero di messaggi necessari per una singola comunicazione.

Un altro problema relativo alle MANET, e più in generale relativo a tutti i sistemi mobili autonomi, è causato dalla memorizzazione e dall'aggiornamento della struttura di rete da parte dei singoli agenti. Tale aspetto è necessario per il corretto instradamento dei pacchetti, poiché ogni nodo deve sempre sapere dove inoltrare i pacchetti ricevuti o generati affinché giungano al destinatario corretto. Per far ciò, generalmente si usano delle *tabelle di routing* o altri meccanismi simili, che fungono da matrici di adiacenza per il nodo e sono rappresentati da basi di dati in cui vengono elencate le regole per il corretto instradamento. Un semplice esempio di una tabella di routing è illustrato in tabella 1.1, dove con *costo* si intende il numero di nodi intermedi necessari per raggiungere la destinazione[6].

Dal momento in cui la configurazione della rete può modificarsi ad ogni step dell'algoritmo di coordinamento, occorre far attenzione che ciascun nodo

[6]In generale non è detto che il costo rappresenti il numero di *hop* intermedi, in quanto si può tenere conto ad esempio anche del peso del collegamento nel caso di archi pesati. In tal caso, il costo è dato dalla somma totale dei pesi sui singoli archi per raggiungere il nodo finale.



Destination	Cost	Next hop
Node-0	1	Node-0
Node-1	1	Node-1
Node-3	1	Node-3
Node-4	2	Node-0
Node-4	2	Node-5
Node-5	1	Node-5

Tabella 1.1: Esempio di una tabella di routing (nodo 2).

mantenga sempre aggiornate le informazioni relative alla topologia assunta. Tra i fattori che influenzano particolarmente questo aspetto, si identificano facilmente: la possibilità di riconfigurazione dei collegamenti tra i nodi, la possibilità di inserimento o di eliminazione di un nodo dalla rete e l'aumento della distanza fra gli stessi (che è indispensabile non solo per stimare il *costo* del collegamento, ma anche per controllare che il collegamento non si interrompa e non si superi la distanza massima di copertura del segnale).

In conclusione, nella progettazione e nello sviluppo di un algoritmo di coordinamento distribuito per sistemi multi-robot mobili, è necessario far attenzione ai seguenti aspetti fondamentali:

Topologia di rete

È forse l'aspetto principale dal quale derivano tutte le scelte implementative e progettuali successive. Una rete con un gran numero di collegamenti è sicuramente più robusta e tollerante ai guasti, ma richiede al contempo un supporto alla comunicazione molto più complesso, nonché un numero di nodi cospicuo.

Aggiornamento della rete

Qualora si preveda una struttura dinamica della rete, la particolare configurazione assunta dai nodi deve essere periodicamente aggiornata e disponibile quanto prima a tutti i nodi. Occorre tener presente che tale aspetto non interessa solamente i casi in cui è possibile aggiungere o rimuovere agenti dal team, bensì molte volte è necessario che la distanza fra i singoli robot venga costantemente monitorata per non perdere la connettività, qualora questa rappresenti una forte specifica di progetto.

Tipologia degli agenti

Non è detto che tutte le unità mobili autonome debbano svolgere lo stesso lavoro. Al contrario, per diversi motivi molto spesso si preferisce offrire maggior *intelligenza* solo ad alcuni nodi e non a tutti, ad esempio perché la sensoristica è troppo costosa ed è disponibile in numero ristretto, oppure perché alcuni agenti hanno minor potenza computazionale e non sono in grado di gestire ulteriori apparati esterni.

Altri aspetti altresì importanti non vengono affrontati in questa trattazione, poiché riguardano un stadio progettuale ad un livello più basso, tra cui la scelta del *sistema operativo* (preferibilmente *real-time* e capace di gestire anche i task di controllo per il movimento, oltre a quelli necessari per il coordinamento) e del particolare *protocollo di networking* a livello applicativo.

1.3 Stato dell'arte

Facchinetti, Franchino e Buttazzo [**11**] hanno proposto una strategia di coordinamento completamente distribuita con l'obiettivo di mantenere la connettività fra i robot attraverso lo scambio periodico delle informazioni sullo stato.

Lo schema suggerito è basato su una struttura organizzata secondo il paradigma *Leader/Follower*, in cui i *Leader* si muovono verso destinazioni predefinite per raggiungere l'obiettivo globale, mentre i *Follower* hanno lo scopo di coordinarsi per mantenere la connettività tra i *Leader*.

Il risultato descritto nella ricerca consiste nel caso di studio composto da due soli leader, ciascuno dei quali deve raggiungere una destinazione prestabilita. Un follower è un membro del team dedicato ad aiutare i due leader nell'obiettivo, facendo sì che la comunicazione fra di essi non venga persa. I nodi comunicano fra loro attraverso il cammino $\mathcal{P}$, definito come il percorso tra i leader r_{l_1} e r_{l_2} in cui ogni vertice del grafo viene visitato una sola volta. Non è detto però, che il numero di nodi m di $\mathcal{P}$ sia pari al numero di nodi n del grafo ($m \leqslant n$), poiché il percorso $\mathcal{P}$ evolve dinamicamente nel tempo (sarebbe più corretto parlare di $\mathcal{P}(t)$). Dal punto di vista della connettività, $\mathcal{P}$ rappresenta il cammino "a più alta

priorità" tra i due leader, ovvero il percorso la cui connettività deve essere sempre garantita per ottenere la piena operatività del team.

Dalla Teoria dei Grafi si nota che il concetto più vicino a $\mathcal{P}$ è il *cammino hamiltoniano* [**12**], un percorso all'interno di un grafo che tocca tutti i vertici visitandoli esattamente una volta sola. Purtroppo, un grafo può non avere un cammino hamiltoniano ed oltretutto l'algoritmo di ricerca per costruzione costituisce un problema *NP-completo*. Il caso migliore si ha quando $m = n$, per cui il percorso $\mathcal{P}$ rappresenta esattamente un cammino hamiltoniano.

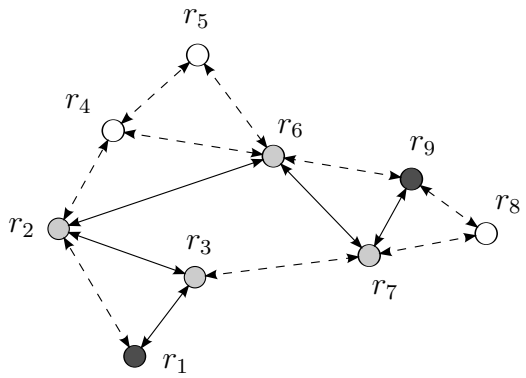


Figura 1.3: Topologia di rete proposta in [**11**].

In riferimento all'esempio di grafo illustrato in figura 1.3 viene descritta la seguente classificazione dei nodi:

Leader: *grigio scuro*, hanno il compito di eseguire dei task specifici e vengono scelti in base alle loro capacità in termini di sensori e attuatori;

Inner Follower (IF): *grigio chiaro*, mantengono la connettività diretta fra i leader attraverso il percorso $\mathcal{P}$, rappresentato da linee bidirezionali piene;

Outer Follower (OF): *bianco*, hanno l'obiettivo di rimanere il più possibile vicino agli IF per poter essere inclusi dinamicamente al percorso $\mathcal{P}$ e diventare IF a loro volta.

Le linee tratteggiate rappresentano i collegamenti wireless tra i nodi che hanno la possibilità di ascoltarsi gli uni con gli altri. È opportuno notare che $\mathcal{P}$ è utilizzato per mantenere la connettività fra i nodi che devono scambiarsi informazioni; ciononostante, non è detto che quello sia il percorso più breve. Qualora esistesse

un percorso alternativo a distanza minore, questo viene comunque sfruttato per il forwarding multi-hop (in figura 1.3 il percorso più breve è dato da $r_1 \leftrightarrow r_3 \leftrightarrow r_7 \leftrightarrow r_9$).

All'istante iniziale in cui viene definita la rete, non vi è alcuna differenza fra i nodi. La rete è completamente connessa e ancora non sono stati scelti i compiti da assegnare ai diversi agenti.

1.3.1 Inizializzazione

La fase di inizializzazione della rete attua le seguenti operazioni:

1. Si ***scelgono i due leader*** r_{l_1} e r_{l_2} in base al compito specifico da portare a termine;
2. Una volta selezionati i leader, i nodi interagiscono fra loro per ***definire il percorso*** $\mathcal{P}$, definendo anche la partizione fra i due insiemi di follower, IF e OF. In questa fase non è necessario trovare il cammino più lungo possibile affinché tutti i nodi vengano inclusi (il cammino hamiltoniano), poiché $\mathcal{P}$ si estende dinamicamente col tempo per inserire gli OF nel path;
3. Ogni nodo r_i ***invia periodicamente*** in broadcast le proprie informazioni di stato e ***usa quelle ricevute*** per pianificare i propri movimenti.

Al termine di questa fase, la rete è in grado di muoversi in modo che ogni nodo porti a termine il suo obiettivo sulla base del compito assegnato.

1.3.2 Strategia di movimento

L'obiettivo dei **leader** è di raggiungere la destinazione assegnata per portare a termine la missione. Essi però non possono muoversi il più veloce possibile così da aumentare le performance globali del sistema, ma devono far attenzione ad adattare la loro velocità in modo da non perdere il collegamento con i due agenti vicini all'interno del percorso $\mathcal{P}$.

Al contrario, i follower devono restare uniti e rimanere a supporto dei due leader. In particolare, gli **inner follower** $r_{p_i} = r_j \in \mathcal{P}$ che formano il cammino primario

ricevono informazioni provenienti dai nodi predecessori e da quelli successori, detti *nodi critici*. I leader, in questo caso, sono considerati come semplici IF, per cui $r_{p_1} = r_{l_1}$ e $r_{p_m} = r_{l_2}$. Per mezzo delle informazioni ricevute, ciascuno inner follower si muove calcolando il punto medio x_{med} tra i suoi due nodi critici. Dato il nodo r_{p_i} con predecessori e successori rispettivamente $r_{p_{i-1}}$ e $r_{p_{i+1}}$, indicando con x_i la posizione del nodo i-esimo, il set point ad ogni passo dell'algoritmo viene calcolato secondo l'equazione 1.1.

$$x_{med} = \frac{x_{p_{i-1}} + x_{p_{i+1}}}{2} \tag{1.1}$$

Il nodo deve adattare la sua direzione e la sua velocità in modo da raggiungere ad ogni step il punto medio x_{med} calcolato, ogni volta che riceve le informazioni dai sui nodi critici.

Il movimento degli **outer follower** segue invece un'altra direzione. Il loro scopo si incentra sul rimanere il più possibile vicino agli IF, in modo da diventare parte integrante del percorso principale a loro volta. Ogni OF, sia esso r_i, si calcola e memorizza la distanza g_i in termini di nodi intermedi dal primo IF disponibile, secondo la seguente regola:

- Un nodo appartenente a $\mathcal{P}$ ha sempre $g_i = 0$;
- Un nodo con un collegamento diretto con un inner follower ha $g_i = 1$;
- Un nodo con collegamenti solo con altri outer follower ha $g_i = \min_{r_j \leftrightarrow r_i} \{g_j\} + 1$.

Grazie a tale informazione che viene periodicamente distribuita dai nodi insieme alla loro posizione, ciascun OF ha il compito di seguire il nodo che mantiene un g_i minore del proprio. Il nodo che ha invece $g_i = 1$ viene eletto a IF solo quando la distanza euclidea fra lui e due nodi consecutivi di $\mathcal{P}$ è minore del raggio r di trasmissione[7].

$$\left\|x_i - x_{p_j}\right\| \leq r \quad \wedge \quad \left\|x_i - x_{p_{j+1}}\right\| \leq r \tag{1.2}$$

A questo punto, il nuovo follower diventa parte integrante del percorso principale, permettendo così un aumento dell'estensione di rete massima pari a $2r$.

[7]Il raggio di propagazione delle onde elettromagnetiche si suppone sempre costante e uguale per tutti i nodi del grafo.

1.3.3 Vantaggi e limitazioni

Il vantaggio più grande nell'uso della strategia di rete appena descritta risiede nel numero di informazioni scambiate fra i nodi all'interno del percorso $\mathcal{P}$. Agli inner follower, infatti, non serve sapere nient'altro se non la posizione dei due nodi che stanno prima e dopo di lui nel percorso, così da permettere di calcolarsi il punto medio come da equazione 1.1.

Un altro vantaggio consiste nella possibilità di estendere la rete con un numero arbitrario di nodi supplementari. La presenza degli outer follower in prossimità del percorso principale fornisce un notevole supporto alla connettività globale e permette una maggior robustezza del protocollo, grazie alla presenza di link talvolta ridondanti.

In compenso, la strategia proposta lascia una serie di problematiche e di questioni non affrontate. Innanzitutto, il movimento dei nodi di $\mathcal{P}$ non garantisce che la connettività sia sempre mantenuta fra i diversi inner follower. Questo perché il solo calcolo del punto medio non porta ad un controllo sulla rottura del link, che di per sé è generalmente causata da un'estensione dei leader.

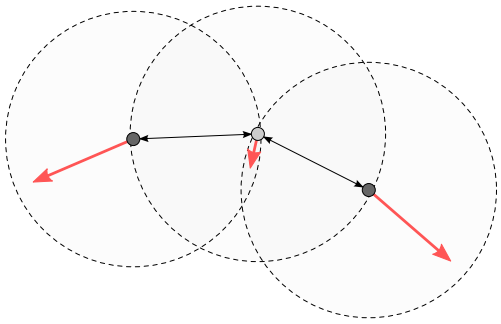


Figura 1.4: Problema del movimento dei leader.

Come si vede in figura 1.4 (le frecce in rosso indicano la distanza e il verso che deve percorre ciascun nodo per raggiungere l'obiettivo), anche una velocità ridotta del leader non assicura che esso rimanga entro il raggio dell'IF vicino. In questo caso, il collegamento tra il leader e il follower è destinato inevitabilmente a rompersi, facendo sì che la rete perda la connettività fra i due leader e non sia più

in grado di portare a termine il compito (salvo un riavvicinamento non previsto dei nodi).

Inoltre, non viene completamente affrontato il caso di tre o più leader, che inesorabilmente aumenta la possibilità di rottura dei collegamenti nel cammino principale, così come non viene in alcun modo analizzata la possibilità di evitare le collisioni sia fra i nodi che con eventuali ostacoli durante il movimento.

1.4 Approccio proposto

Partendo dalla strategia di coordinamento presentata nella sezione 1.3, si è cercato di ampliare la soluzione cercando quanto più possibile di risolvere le problematiche affrontate durante l'analisi.

L'idea iniziale prende spunto dalle reti neurali atte all'organizzazione autonoma dei neuroni sullo spazio, meglio note come *Self-Organizing Map* (**SOM** o **Reti di Kohonen**[8]), che verranno illustrate nella sezione 2.1. Tali reti fanno uso dell'apprendimento non supervisionato per produrre una rappresentazione discreta dei campioni di training nello spazio di ingresso. Grazie alla loro capacità di mantenere delle proprietà topologiche, rispetto alle normali reti neurali, le SOM sono particolarmente utili per la visualizzazione di un numero di dati molto elevato. Tuttavia, come la grande maggioranza delle reti neurali artificiali, anche le mappe auto-organizzanti non rappresentano un esempio di sistema autonomo distribuito. Lo scopo primario della ricerca si incentra, infatti, sull'adattamento delle SOM al problema proposto, cercando di estendere l'algoritmo di apprendimento in modo che i singoli passi vengano eseguiti autonomamente sui nodi, rendendo di fatto il sistema completamente distribuito.

Con questo presupposto è stato sviluppato un simulatore di reti di Kohonen, che permette l'addestramento e la modifica dei parametri della rete in base al numero di neuroni dello strato d'uscita e al numero di campioni di ingresso nel *training-set*.

[8]Dal nome del suo ideatore Teuvo Kohonen.

Idealizzando i neuroni della SOM come gli agenti del team, si è deciso di organizzare la rete secondo una struttura topologica a forma di "*linea spezzata aperta*" come in figura 1.5. Così facendo, ciascun agente ha la possibilità di comunicare solamente con il vicino precedente e il vicino successivo[9], mantenendo ad un livello basso il numero di messaggi necessari per la coordinazione.

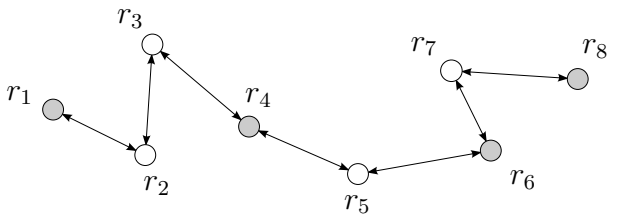


Figura 1.5: Esempio di rete organizzata a *linea spezzata aperta*.

La soluzione proposta, inoltre, non si limita al caso di due soli unità leader atte a raggiungere l'obiettivo globale della rete, bensì si apre ad un numero ipoteticamente infinito di nodi "intelligenti". Ad ogni leader verrà assegnata una destinazione predefinita, da raggiungere nel più breve tempo possibile cercando di non perdere mai la connettività globale della rete.

In ultima analisi (vedi capitolo 4), si suppone di dotare tutti i nodi di una sensoristica di base aggiuntiva oltre a quella necessaria per il movimento e per la connettività, composta di un certo numero di sensori di prossimità. Tali sensori possono essere montati in numero arbitrario su ogni agente e con caratteristiche diverse per ogni singolo robot. Lo scopo di tale operazione è il rilevamento di oggetti che ostacolano il passaggio delle unità o che ne deviano la traiettoria, siano essi rappresentati da muri, pali o cartelli, piuttosto che altri agenti della rete che si muovono incrociando le traiettorie.

[9]Il concetto di *vicinato* viene preso dalla terminologia delle mappe auto-organizzanti.

Capitolo 2

Architettura del sistema

Nel seguente capitolo verrà illustrato in dettaglio la struttura del sistema multi-robot proposto. La strategia adottata si pone nell'ambito dei *modelli cinematici a tempo discreto* in cui la posizione di ogni agente viene aggiornata in funzione della variazione tra uno scatto temporale e l'altro della variabile discreta. Da esso deriva anche la dimensione del vettore velocità in termini di modulo, direzione e verso ad ogni passo di controllo.

La ricerca si incentra per lo più nel definire una strategia di coordinamento adattabile ad una squadra di velivoli autonomi, in particolar modo ai *quadricotteri* come noto dalla sezione 1.1. Nella trattazione non si affrontano difatti i problemi relativi a *vincoli olonomi* presenti sulla maggior parte degli agenti autonomi, il che pone il lavoro perfettamente in linea con gli aeromobili di questo tipo. Questo perché i quadricotteri sono in grado di modificare i tre angoli di rotazione *beccheggio*, *rollio* e *imbardata* (in riferimento al centro di massa) in maniera pressoché indipendente dalla posizione e dalla velocità del velivolo stesso. Inoltre hanno la possibilità di muoversi (o di rimanere in posizione stabile) svincolandosi dal riferimento anteriore, sfruttando solo beccheggio e rollio, che è invece fondamentale per aeroplani o la maggior parte dei rover[1].

[1]Robot di terra dotati di ruote.

2.1 Self-Organizing Map

Nel 1982, il professore finlandese Teuvo Kohonen [**13**] ha descritto il modello di una rete neurale artificiale che sfrutta l'apprendimento non supervisionato per generare mappe sensoriali simili a quelle osservate nella corteccia somatosensoriale del cervello umano[2]. Le ha chiamate *mappe auto-organizzanti* (molto spesso si trovano anche sotto il nome di *reti di Kohonen*) e forniscono un modo per rappresentare dati multi-dimensionali in spazi di dimensione molto più bassa (generalmente una o due dimensioni), attraverso una tecnica nota come *quantizzazione vettoriale*[3].

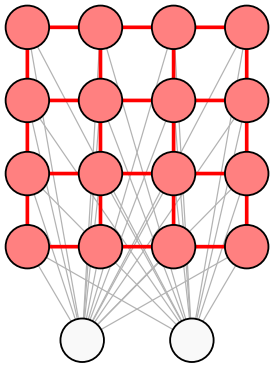


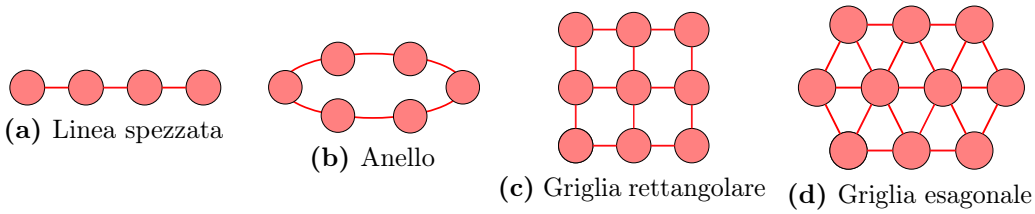
Figura 2.1: SOM con neuroni d'uscita organizzati a griglia rettangolare.

La struttura della rete neurale è composta da due *layer* e da collegamenti che partono da ciascun neurone nello strato di ingresso e arrivano a ciascun neurone nello strato d'uscita, detti *pesi*. Ogni neurone di uscita ha associato un vettore dei pesi della stessa dimensione del vettore di ingresso. Rispetto alle comuni reti neurali, però, le SOM permettono di mantenere una struttura topologica tra i neuroni del secondo strato, inserendo il concetto di *vicinato*. In particolare è possibile definire una qualunque struttura permanente fra i neuroni d'uscita (vedi figura 2.1, in cui le linee rosse non rappresentano dei veri e propri collegamenti, ma mostrano solo le adiacenze tra i nodi, poiché non esistono interconnessioni

[2]La corteccia somatosensoriale è una regione del cervello nel lobo parietale che si occupa di raccogliere le afferenze prevalentemente di tipo tattile e le informazioni propriocettive dai recettori muscolari e articolari.

[3]Il processo di riduzione delle dimensioni dei vettori rappresenta essenzialmente una forma di compressione dei dati.

laterali all'interno del reticolo) grazie alla quale è possibile esprimere il concetto di vicinato e di *distanza* fra i neuroni all'interno della griglia. In figura 2.2 sono invece rappresentati alcuni esempi di organizzazione topologica dei neuroni nel secondo strato.



(a) Linea spezzata **(b)** Anello **(c)** Griglia rettangolare **(d)** Griglia esagonale

Figura 2.2: Esempi di organizzazione del vicinato nelle *SOM*.

Durante l'algoritmo di apprendimento, i nodi partecipano – in maniera competitiva – alla specializzazione di un parte del reticolo SOM in modo da rispondere a particolari pattern d'ingresso (così come avviene nel cervello umano). La capacità di generalizzare della rete consiste nella possibilità di riconoscere particolari campioni d'ingresso che non ha mai incontrato durante l'addestramento. Il nuovo campione viene "assimilato" dalla rete che allo stesso tempo viene aggiornata sotto il punto di vista dei pesi dei neuroni d'uscita.

In breve, l'apprendimento delle SOM si compone dei seguenti step:

1. Si inizializza la rete con valori dei pesi scelti in maniera del tutto casuale per ogni neurone sullo strato d'uscita;
2. Si sceglie arbitrariamente un vettore di ingresso (*pattern*) dal *training-set* e lo si fornisce in ingresso alla rete neurale;
3. A differenza delle reti convenzionali[4], si esamina ogni nodo per determinare quale fra tutti ha il vettore dei pesi più "simile" all'input fornito. Il *vincitore* viene generalmente chiamato **BMU** (*Best Matching Unit*);
4. Determinato il BMU vengono attivati tutti i nodi a lui "*vicini*" in base alla topologia scelta per il secondo strato e alla *funzione di vicinato*. Que-

[4]In cui l'output di ogni neurone j viene calcolato applicando una particolare *funzione di attivazione* al prodotto vettoriale tra il vettore d'ingresso X e quello dei pesi W_j: $y_j = f_a\left(X \cdot W_j\right) = f_a\left(\|X\| \cdot \|W_j\| \cdot \cos\theta\right) = f_a\left(\sum_{i=1}^{n} x_i \cdot w_{ji}\right)$.

st'ultima sfrutta il *raggio di vicinato* che viene inizializzato ad un valore alto (tipicamente al dimensione del reticolo) per poi diminuire ad ogni step, e viene utilizzata per definire quanti e quali neuroni intorno al vincitore verranno attivati in questa fase. Ogni nodo trovato in questo intorno fa parte del vicinato del BMU[5];

5. Ad ogni neurone selezionato vengono modificati i pesi in modo proporzionale alla loro distanza dal vincitore, per renderli più "simili" all'ingresso, di fatto attraendoli verso il campione selezionato. È anche possibile che, al contrario, la funzione causi l'allontanamento – dei pesi – di alcuni neuroni eccitati dal vettore di input[6];
6. Si riparte dallo step 2 continuando per un totale di N iterazioni.

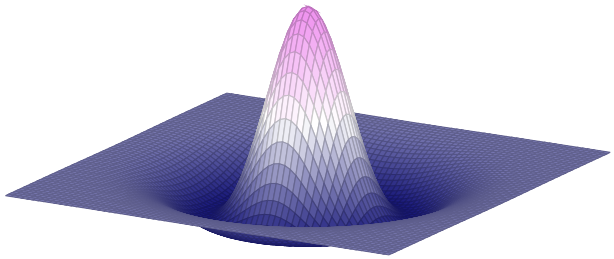


Figura 2.3: Laplaciano di una Gaussiana a dominio bidimensionale.

2.1.1 Apprendimento competitivo

La specializzazione dei neuroni nel riconoscere ingressi simili permette, alla fine del processo di apprendimento, di definire il mapping tra lo spazio degli ingressi e lo spazio discretizzato della mappa. Affinché tale mapping sia possibile

[5]L'estremizzazione di questa tecnica porta all'aggiornamento del solo vincitore. Tale approccio è noto in letteratura col nome di *Winner-Take-All* (WTA) e rappresenta un caso particolare delle reti di Kohonen in cui la funzione di vicinato è non lineare e del tipo $\Phi(d) = \begin{cases} 1 & d = 0 \\ 0 & \text{altrimenti} \end{cases}$, dove d rappresenta la distanza di un dato nodo dal vincitore.

[6]Un esempio di funzione di questo tipo è il noto *Mexican-Hat* rappresentato generalmente con un laplaciano di una Gaussiana (LoG) $\nabla^2 G = \frac{4}{\sigma^2}\left(1 - \frac{x^2+y^2}{\sigma^2}\right) e^{-\frac{x^2+y^2}{\sigma^2}}$ o come differenza di gaussiane (vedi figura 2.3).

è necessario definire una funzione che quantifichi la somiglianza fra il vettore di ingresso e il vettore dei pesi di ciascun neurone.

Nello spazio vettoriale n-dimensionale, con n dimensione del vettore di ingresso, la soluzione che viene comunemente adoperata fa uso della *distanza euclidea* fra i due vettori[7]:

$$d_j = \|X - W_j\| = \sqrt{\sum_{i=1}^{n} (x_i - w_{ji})^2} \tag{2.1}$$

Una volta determinato il rapporto di similitudine con il vettore di ingresso diventa immediata la ricerca del BMU. In riferimento all'equazione 2.1, la scelta più opportuna si ottiene selezionando il nodo che ha il minor d_j rispetto a tutti gli altri, ovvero quello più vicino al pattern fornito. In questo senso, i neuroni competono globalmente ad ottenere un migliore risposta della rete rispetto agli ingressi forniti.

L'elezione del vincitore causa un aggiornamento del suo vettore dei pesi e di quello dei vicini. La funzione di vicinato Φ (*neighborhood function*) viene applicata in base alla struttura topologica del reticolo, come mostrato in figura 2.4. Il BMU eccita un insieme di neuroni che fanno parte di un suo intorno di raggio σ, il quale viene decrementato ad ogni iterazione. In pratica Φ è una funzione dipendente da σ e dalla posizione del neurone j considerato rispetto al vincitore u, quindi è più corretto parlare di $\Phi(u, j, \sigma)$.

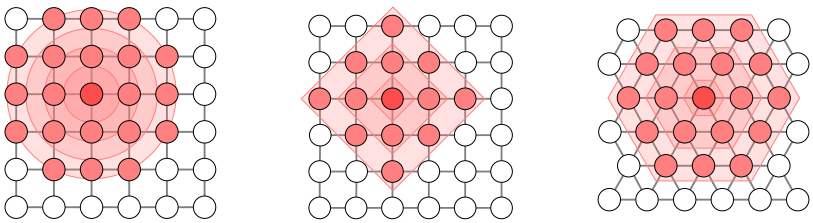


Figura 2.4: Esempi di *bolla di attivazione* del vincitore.

[7]Un altro modo semplice di stabilire la somiglianza è utilizzare la sola somma pesata degli ingressi, ovvero l'uscita prodotta del neurone nel caso di funzione di una attivazione lineare: $y_j = f_a(X \cdot W_j) = X \cdot W_j = \sum_{i=1}^{n} x_i \cdot w_{ji}$.

L'aggiornamento di tutti i pesi associati ai neuroni nello strato d'uscita avviene secondo l'equazione 2.2.

$$W_j(t+1) = W_j(t) + \alpha \cdot [X - W_j(t)] \cdot \Phi(u, j, \sigma) \tag{2.2}$$

Questo è il fattore che distingue le mappe auto-organizzanti dalle reti competitive, in quanto non si limitano ad imparare solo la distribuzione dei vettori con cui sono addestrate, ma tengono conto della loro ripartizione topologia.

Nell'equazione 2.2, il parametro α è detto *coefficiente di apprendimento* (learning rate) e, come σ, è una variabile che evolve nel tempo in maniera decrescente, in particolare α varia nell'intervallo $[0, 1]$. Il decadimento di tali variabili viene definito ad ogni passo dell'algoritmo tipicamente in maniera esponenziale (ad argomento negativo); tuttavia è possibile esplicitare arbitrariamente l'andamento di questi parametri in base alla tipologia di applicazione.

2.2 Modello del sistema

Le reti SOM sono comunemente utilizzate come coadiuvanti per la visualizzazione di grandi quantità di dati. Esempi applicativi sono il *clustering*[8], la *compressione dei dati* o la risoluzione approssimativa di problemi NP-completi come il *problema del commesso viaggiatore*[9] (*Travelling Salesman Problem*, TSP).

Prendendo spunto proprio dal TSP, si è deciso di implementare una mappa le cui adiacenze fra i nodi dello strato d'uscita formano una linea spezzata aperta così come mostrato in figura 2.2a[10]. A tal scopo, si è modellata la rete SOM per

[8] Il *centroide* di ogni cluster è rappresentato da un nodo della rete.

[9] L'equivalente della ricerca di un ciclo hamiltoniano all'interno di un grafo pesato.

[10] In TSP invece la configurazione topologica della rete è ad anello o a linea spezzata chiusa (vedi figura 2.2b). Si faccia attenzione che nel problema del commesso viaggiatore il numero neuroni scelti per ottenere una possibile soluzione al problema è generalmente maggiore rispetto al numero di destinazioni che il commesso viaggiatore deve visitare, tipicamente il doppio. Uno dei primi problemi affrontati durante lo sviluppo del tester di reti di Kohonen è stato quello di far sì che la rete convergesse con un numero di pattern d'ingresso pari al numero di nodi, cercando di evitare, come molto spesso accade, che i neuroni si interponessero fra due diversi input (vedi figura 2.5).

poterla arrangiare al problema di coordinamento di sistemi multi-robot.

In particolare, si definiscono le associazioni logiche che seguono:

- Ogni neurone modella un agente mobile con coordinate nello spazio pari ai pesi associati;
- Il numero di neuroni nello strato d'ingresso, e quindi dei pattern nel training-set, definisce la dimensione spaziale in cui si affronta il problema (2D o 3D);
- I campioni forniti in ingresso rappresentano le destinazioni da raggiungere da parte dei leader della rete. Come previsto dall'algoritmo di apprendimento, tali punti vengono presentati alla rete più volte in modo che ogni neurone si avvicini sempre più ad una destinazione (ovvero che ogni robot raggiunga l'obiettivo).

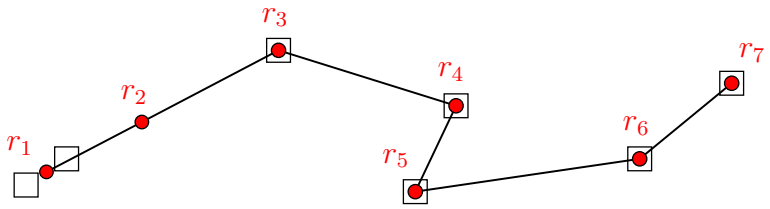


Figura 2.5: Interposizione dei neuroni r_1 e r_2 della SOM fra due coppie diverse di campioni del training-set.

Purtroppo è evidente dalla sezione 2.1.1 che il processo di apprendimento delle reti di Kohonen non è distribuito; poiché la scelta del vincitore ad ogni passo iterativo viene effettuata a livello globale, ad esempio scegliendo il nodo la cui distanza rispetto alla destinazione fornita è la più breve, risulta obbligatoria una riconsiderazione sul metodo di ricerca del BMU. In altri termini, i singoli robot devono potersi muovere in maniera autonoma, senza che nessuna entità globale li avvisi su chi fra loro è più vicino a ciascun punto d'arrivo.

La strategia adottata non fa uso di metodi di consenso distribuito per scegliere l'agente che verrà eletto vincitore ad ogni iterazione (il che genererebbe un aumento notevole del numero di messaggi scambiati nella rete) bensì elimina totalmente il problema fissando un certo numero di BMU in fase di inizializzazione, pari al numero di destinazioni. È necessario inoltre che sia stabilito un ordine fra i target,

tale da rendere possibile il calcolo della distanza massima che può essere coperta dal dato numero di agenti.

Ogni BMU eletto viene detto **Master** e non è altro che un nodo leader (del modello Leader/Follower) a cui viene assegnata una specifica destinazione. La scelta dei nodi da far diventare master che è stata adoperata anche durante la fase testing verrà meglio illustrata nella sezione 3.2 e si basa sulla distanza fra le coordinate di una target e l'altro. Le altre scelte implementative riguardano la funzione di vicinato Φ e l'aggiornamento del raggio di vicinato σ e del coefficiente di apprendimento α:

$$\begin{aligned} \Phi(u, j, \sigma_t) &= \begin{cases} e^{-\frac{(u-j)^2}{2 \cdot {\sigma_t}^2}} & |u - j| \leq \sigma_t \\ 0 & \text{altrimenti} \end{cases} \\ \alpha_{t+1} &= \alpha_t \cdot \alpha_{decay} \\ \sigma_{t+1} &= \sigma_t \cdot \sigma_{decay} \end{aligned} \tag{2.3}$$

L'andamento della funzione lineare da cui è ricavata la funzione di vicinato non lineare utilizzata nell'equazione 2.3 è illustrato in figura 2.6.

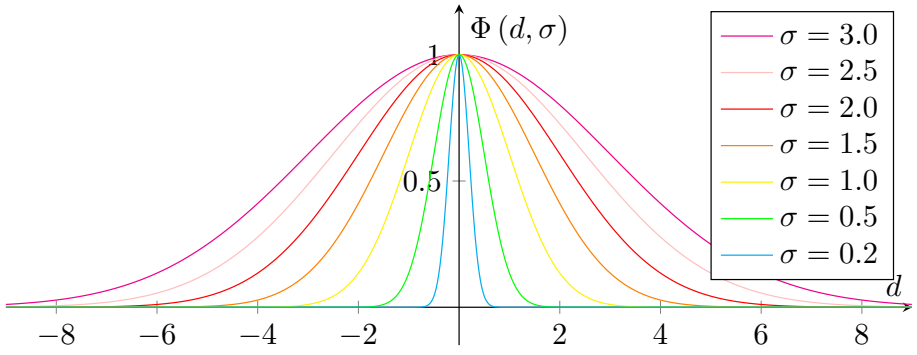


Figura 2.6: Funzione di vicinato $\Phi(d, \sigma) = e^{-\frac{d^2}{2 \cdot \sigma^2}}$ al variare di σ.

2.3 Classificazione dei nodi

Il modello di rete proposto nella presente tesi prende spunto sia dalle mappe auto-organizzanti di Kohonen descritte nella sezione 2.1 che dalla struttura proposta da Facchinetti, Franchino e Buttazzo in [**11**]. Nella sua forma definitiva i nodi sono suddivisi in due gruppi principali:

- **Master:** sono nodi speciali della rete, atti a portare a termine il compito per cui la stessa rete è stata formata. La scelta di tali agenti può essere fatta in base alla dotazione che hanno a bordo (in termini di sensori e di attuatori), in modo del tutto casuale o in qualunque altra maniera l'utente ritenga più doverosa. È sicuramente opportuno porre l'attenzione sulla distanza intermedia lungo la linea spezzata fra un master e l'altro affinché sia possibile la copertura di tutti i punti d'arrivo[11].

- **Follower:** sono tutti gli altri nodi della rete. Il loro compito è di rimanere quanto più vicino possibile ai master per supportare la comunicazione durante il movimento. Come sarà più chiaro nella sezione 3.5, la quasi totalità degli accorgimenti adottati per i follower per evitare che i collegamenti fra i nodi vengano interrotti sono validi anche per i master.

Come in tutti i problemi di coordinamento autonomo distribuito, la particolare configurazione del sistema dipende fortemente anche dall'insieme d'ingresso. In tal senso, occorre tener presente che nonostante non esista un vero e proprio vincolo generale rispetto al numero di follower, è invece necessario che il numero di master sia esattamente pari al numero di destinazioni.

In realtà, è possibile stabilire in maniera esplicita anche i vincoli sul numero totale di nodi nella rete, da cui è facile derivare le restrizioni sul numero di follower e sulla loro posizione all'interno del reticolo.

[11] Qualora l'insieme di ingresso ammetta almeno una copertura possibile.

2.4 Cinematica discreta

Poiché il contributo del presente lavoro di tesi è finalizzato ad offrire anche una piattaforma di simulazione (nata dall'evoluzione del tester di reti SOM) che verrà utilizzata anche in fase di valutazione delle prestazioni, è necessario che il controllo del movimento nella rete sia opportunamente analizzato anche in funzione delle variabili di controllo[12].

L'evoluzione dello stato del sistema è governato prima di tutto dal parametro temporale discreto $t = \{0, 1, 2, 3, \dots\}$ che rappresenta l'indice di una sequenza o di una successione di un insieme (ipoteticamente infinito) di istanti di tempo separati da intervalli di dimensione regolare. Lo stato delle singole unità dev'essere valutato al termine di ogni *finestra temporale* T allo scopo di informare i nodi vicini dell'evoluzione globale della rete.

Tralasciando per un secondo l'algoritmo di coordinamento, ad ogni *passo di controllo* ciascun agente conosce e memorizza le informazioni sul proprio stato in termini di posizione e orientamento, rispettivamente p_i (vettore nello spazio euclideo in relazione ad un riferimento arbitrario O che denota l'*origine*) e θ_i. Ai fini dello studio del comportamento cinematico è sufficiente idealizzare p_i come un semplice vettore a due componenti $[x_i, y_i]$, dove $x_i, y_i \in \mathbb{R}$, mentre $\theta_i \in [0, 2\pi]$ quantifica l'angolo ottenuto considerando la "*testa*" del robot in relazione all'asse delle ascisse[13].

Il vettore velocità è comunemente definito come la derivata del vettore posizione rispetto al tempo; tuttavia in un contesto quantizzato in cui si ha a che fare con domini di riferimento non lineari, si considera il semplice rapporto incrementale

[12] Lo studio delle cause e delle circostanze che determinano il moto dei corpi non sarà trattato, in quanto fortemente dipendete dalle caratteristiche fisiche e meccaniche dei robot impiegati. Ci si limita ad esaminare brevemente solo la cinematica dei punti, trascurando alcuni aspetti relativi alla loro dinamica.

[13] In cui si è preso in considerazione il caso di un sistema di coordinate ortogonale e bidimensionale. Qualora si voglia aggiungere una terza dimensione spaziale al problema, è necessario che l'orientamento non sia più considerato come un angolo, ma come una coppia di angoli oppure attraverso un vettore unitario la cui coda è posizionata in $p_i = [x_i, y_i, z_i]$.

dato dall'equazione 2.4.

$$v_i\left(t+1\right) = \frac{p_i\left(t+1\right) - p_i\left(t\right)}{T} \tag{2.4}$$

Come spesso accade nel campo dei controlli automatici, si prevede una soglia di saturazione della velocità v_{max} in quanto si ha il problema inverso di stabilire il punto di arrivo al passo di controllo $t+1$. Prendendo in esame un sistema in cui vengono trascurati i contributi istantanei relativi all'accelerazione (come in questo caso), si calcola la nuova destinazione in base alla velocità di soglia come in equazione 2.5.

$$\begin{aligned} x_i\left(t+1\right) &= x_i\left(t\right) + \Big[v_{max} \cdot \cos\varphi_i\left(t\right)\Big] \cdot T \\ y_i\left(t+1\right) &= y_i\left(t\right) + \Big[v_{max} \cdot \sin\varphi_i\left(t\right)\Big] \cdot T \end{aligned} \tag{2.5}$$

Dove si è considerato il vettore velocità $v_i\left(t\right) = v_{max} \cdot \hat{v_i}\left(t\right)$[14] con angolo rispetto alle ascisse pari a $\varphi_i\left(t\right)$.

Le stesse considerazioni valgono per l'angolo di orientamento del velivolo. Come già specificato all'inizio del capito 2, non si esaminano gli effetti dovuti ad eventuali vincoli olonomi, perciò si mantiene l'orientamento del robot completamente indipendente dalla direzione del movimento e dalla loro posizione. Occorre però fare attenzione che così facendo la velocità di rotazione risulta anch'essa indipendente dalla velocità di movimento. Si definisce perciò una velocità angolare limite ω_{max} espressa in radianti al secondo grazie alla quale è possibile stabilire l'orientamento ad ogni passo di controllo:

$$\theta\left(t+1\right) = \theta\left(t\right) + \omega_{max} \cdot T \tag{2.6}$$

[14] $\hat{v_i}\left(t\right)$ è il versore della velocità nella direzione del moto.

Capitolo 3

Algoritmo di coordinamento

Lo schema suggerito per la MANET è quello di un digrafo connesso privo di cicli e di pesi negli archi in cui tutti i nodi hanno al massimo due collegamenti, in altri termini di un grafo lineare (*graph path*) con vertici di grado 2 e 1 e con due soli nodi terminali [**14**].

Ciascun terminale è dunque a conoscenza del nodo che lo precede e di quello che lo segue nella linea del grafo. In questo modo, non vi è la necessità di comunicare in broadcast le informazioni sullo stato di ciascun agente ai fini del mantenimento della connettività, poiché per ognuno di essi è necessaria la sola informazione sull'*indirizzo a livello fisico* a cui inoltrare direttamente i messaggi.

In questo capitolo verranno esposte le strategie di movimento associate a ciascun robot del team, in relazione alle due classificazioni presentate nella sezione 2.3. Saranno illustrate anche le tecniche per evitare che la connettività venga persa fra due nodi consecutivi del reticolo, cosicché sia sempre possibile muovere la formazione per raggiungere i target assegnati, qualora l'insieme di destinazioni fornito ammetta almeno una soluzione possibile.

Ai fini della trattazione, si suppone che tutti gli agenti comunichino attraverso dei link punto-punto dipendenti dal raggio di ricetrasmissione r. Tale parametro rappresenta una costante ben nota al team. Tutte le unità hanno la capacità di inviare e ricevere messaggi entro una distanza massima definita da r. I fenomeni ondulatori delle onde elettromagnetiche, dovuti ad *attenuazione*, *riflessione*, *rifra-*

zione, *dispersione*, *diffrazione* e *scattering*, non vengono trattati nel seguito. La propagazione dei segnali si suppone perfettamente costante all'interno del cerchio (o della sfera) di trasmissione centrata nel baricentro di ciascun nodo.

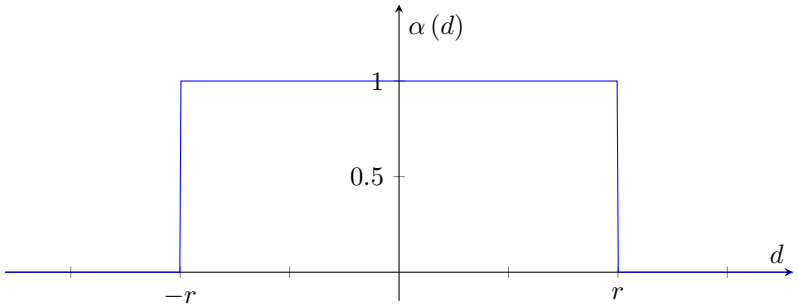


Figura 3.1: Coefficiente di attenuazione α delle onde elettromagnetiche, supposto dipendente dalla sola distanza d e costante nell'intervallo $[-r, r]$.

3.1 Inizializzazione

All'istante iniziale i robot sono idealmente identici e si trovano ad una distanza tale da permettere loro di definire la conformazione di rete. Qualora anche l'equipaggiamento che li compone sia il medesimo, non vi è alcun limite nella scelta dei master nell'insieme dato, eccezion fatta per particolari esigenze dell'utente, il quale può essere obbligato ad optare per una distinzione specifica fra le due classi dei nodi.

Sotto tale aspetto, considerando le unità mobili del tutto equivalenti, si procede alla fase di inizializzazione come segue:

1. Si identificano univocamente gli agenti della rete ***numerandoli*** in maniera crescente da 1 a n, siano $r_1, \ldots, r_n$;
2. Si ***costruisce la MANET*** costruendo una *polilinea aperta* in cui i nodi si susseguono in maniera ordinata rispetto al loro indice identificativo. In questo modo, dato un agente r_i si stabilisce il suo successore e il suo predecessore rispettivamente come r_{i-1} e r_{i+1}. Tali nodi sono anche detti *vicini diretti* del nodo r_i;

3. Allo stesso modo, si ***numerano le destinazioni*** in ordine crescente da 1 a m, siano $d_1, \ldots, d_m$.

Definita la conformazione topologica della rete, si può passare ad eleggere i master scegliendoli fra i nodi. Occorre precisare però che nel caso in cui la scelta dei master sia già stata dettata da fattori esterni, allora l'inizializzazione della rete deve essere aggiornata di conseguenza, come specificato al termine della sezione 3.2.

3.2 Scelta dei master

A questo punto, si suddividono i nodi nelle due classi di appartenenza: *master* e *follower*. Tale processo prende in esame le coordinate dei punti d'arrivo $d_1, \ldots, d_n$ forniti in ingresso e si compone delle operazioni che seguono:

1. Il primo robot del team r_1 viene sempre eletto a master e gli viene assegnata la destinazione d_1;
2. Sia $i = 2$ l'indice della nuova destinazione da considerare e $j = 1$ l'indice dell'ultimo master prescelto;
3. Si calcola la distanza δ fra il punto d_i e il precedente d_{i-1}:

$$\delta = \|d_i - d_{i-1}\|$$

4. In base al raggio di ricetrasmissione r, si determina il numero minimo di nodi intermedi c necessari a raggiungere d_i come:

$$c = \left\lceil \frac{\delta}{r} \right\rceil$$

5. Il nodo con indice $j = j + c$ diviene l'i-esimo master della rete a cui viene associata l'i-esima destinazione d_i;
6. Si incrementa i e si riparte dallo step 3 per un totale di $m - 2$ iterazioni;
7. L'ultimo agente r_n viene anch'esso sempre scelto come master.

Poiché le categorie dei vari robot sono complementari e in numero ridotto, è evidente che tutti gli altri agenti che non sono stati promossi allo stadio di master diventano follower. Tale distinzione è immutabile durante tutto il processo di coordinamento e non è stato analizzato alcun modo per far sì che sia possibile un rimescolamento fra i due gruppi.

Come più volte specificato, l'insieme dei punti di arrivo $D = \{d_1, \ldots, d_m\}$ dev'essere tale da permetterne la copertura con un numero prestabilito n di nodi. A tal scopo, andrebbe prima verificata la *condizione di fattibilità* che può essere brevemente descritta nella definizione che segue.

Controllo di fattibilità preliminare

Dato un insieme ordinato di punti $D = \{d_1, \ldots, d_m\}$, esiste sempre almeno una copertura possibile dei nodi del grafo $G = \{r_1, \ldots, r_n\}$ se viene verificata la condizione 3.1.

$$\sum_{i=2}^{m} \left\lceil \frac{\|d_i - d_{i-1}\|}{r} \right\rceil \leq n \tag{3.1}$$

Nella disequazione 3.1 il membro a sinistra della disuguaglianza rappresenta la distanza di ciascun intervallo tra un target d_i e il successivo, normalizzato al raggio di copertura. Tale valutazione identifica il numero di agenti minimo necessario ad assicurare a tutti gli m master il raggiungimento del proprio punto obiettivo.

Ovviamente i master possono essere scelti a priori in base alle particolari esigenze dell'utente. In tali circostanze non c'è bisogno di definire un'inizializzazione della rete ad-hoc per questo caso d'uso, bensì è sufficiente che la rete venga adeguata in modo che gli agenti su cui si fa affidamento per eseguire il task globale si trovino nella posizione calcolata per i master all'interno del grafo. Il gestore della MANET non dovrà far altro che occuparsi di dislocare gli effettivi attuatori dell'obiettivo globale nel giusto ordine in mezzo ai follower in base alle regole appena descritte. L'algoritmo di aggiornamento viene lasciato a discrezione del progettista, poiché esula dall'argomento della trattazione.

3.3 Variabili di stato

Per garantire un servizio di comunicazione continuativo fra i diversi master della rete per mezzo di ponti intermedi costruiti sui follower, è altresì importante capire quali sono le variabili di interesse necessarie durante il movimento.

Nella sezione 2.4 è stato analizzato un primo set di variabili che identificano lo stato interno degli agenti. Per mezzo delle considerazioni maturate in questo capitolo, è possibile tracciare lo *stato interno* dei nodi della rete attraverso la descrizione schematizzata in tabella 3.1.

Descrizione	**Simbolo**	**Dominio**	**Note**
Identificativo	i	$[0, n]$	
Posizione	p_i	$\mathbb{R}^2/\mathbb{R}^3$	
Orientamento	θ_i	$[0, 2\pi]$	
Destinazione	d_j	$\mathbb{R}^2/\mathbb{R}^3$	Solo per i master
Learning rate	α_i	$[0, 1]$	Solo per i master
Neighbour radius	σ_i	$[0, n]$	Solo per i master

Tabella 3.1: Stato interno di ciascuna unità mobile autonoma.

Come si nota, non tutti i nodi usano i parametri che definiscono il suo stato, al contrario alcuni di essi sono disponibili solo per i master. I follower, infatti, possono evitare di istanziare i parametri di cui non hanno bisogno, evitando in questo modo di allocare blocchi di memoria inutilizzati.

Al tempo stesso, bisogna far attenzione al fatto che *tutti* gli agenti della MANET devono memorizzare anche lo stato dei loro vicini. Questo vuol dire che occorre prevedere una struttura logica che descrive il modello concettuale dei nodi adiacenti a quello considerato in termini delle sue variabili di stato (ad esclusione delle informazioni relative ai soli master, ovvero *destinazione*, *learning rate* e *neighbour radius*).

Come sarà più comprensibile nella sezione 3.4, i parametri α_i e σ_i non sono globali e non hanno visibilità a livello rete, bensì sono fortemente legati al master che li detiene. Dalla teoria sulle mappe auto-organizzanti e in merito a quanto già descritto nei paragrafi 2.1 e 2.2, non è possibile che i due coefficienti della SOM

siano noti a tutti i nodi in ogni istante, poiché si sta analizzando il problema in un ambito distribuito.

Nella soluzione proposta non vengono previsti metodi di sincronizzazione dei parametri neurali all'interno del sistema del multi-robot, piuttosto si preferisce che i follower vengano del tutto esclusi dal calcolo del *coefficiente di apprendimento* e del *raggio di vicinato*, demandando tali operazioni ai soli master. Ciascun master calcola i suoi parametri ad ogni passo dell'algoritmo e provvede ad inoltrarli agli altri robot per *propagazione* nel percorso. È chiaro che per portare avanti una strategia di questo tipo, i coefficienti inviati da un qualsiasi master A non devono alterare quelli di un altro ipotetico master B. Quest'ultimo dovrà dunque provvedere ad inoltrare il messaggio agli agenti che lo susseguono ignorandone il contenuto, oppure scartare lo stesso terminando così la propagazione.

3.4 Avvio dei master

Le considerazioni che seguono non tengono conto della conformazione dell'ambiente fisico in cui operano i velivoli autonomi, tantomeno fanno riferimento ad una *regione di lavoro ammissibile* descritta da un perimetro entro il quale limitare il movimento dei droni. Anche la presenza di eventuali ostacoli o la possibilità che i vari agenti collidano fra loro viene attualmente accantonata e sarà meglio analizzata nel capitolo 4.

Al termine dell'inizializzazione della rete, ogni robot conosce esattamente quale compito dovrà portare a termine in merito al suo specifico gruppo di appartenenza; sa anche quali sono i nodi con cui deve periodicamente comunicare per garantire un servizio continuativo di comunicazione. Questa particolarità rappresenta, infatti, un requisito indispensabile per il funzionamento del sistema, senza la quale sarebbe praticamente impossibile portare a termine la missione assegnata al team.

Al tempo $t = 0$ tutti i nodi sono fermi nella loro posizione p_i. Il movimento parte in un primo luogo dai master, i quali, conoscendo le coordinate esatte della destinazione finale d_j da raggiungere, calcolano il set-point intermedio in cui si

porteranno all'istante $t+1$. In seguito aggiornano i *parametri neurali* ad essi associati e comunicano il nuovo stato ai vicini.

Nello specifico ciascun **master** esegue nell'ordine le seguenti azioni:

1. Determina il vettore distanza $\mathbf{d}$ tra la posizione attuale p_i e la destinazione finale d_j, in termini di modulo, direzione e verso. Analizzando il problema nello spazio vettoriale euclideo a due dimensioni si ottiene:

$$\mathbf{d} = d_j - p_i = \begin{cases} \|\mathbf{d}\| = \sqrt{\left(x_{d_j} - x_i\right)^2 + \left(y_{d_j} - y_i\right)^2} \\ \angle\mathbf{d} = \arctan\left(\frac{y_{d_j} - y_i}{x_{d_j} - x_i}\right) \end{cases}$$

2. Lo spazio reale che è in grado di percorrere all'interno della finestra temporale T è funzione della velocità di soglia v_{max} e del proprio coefficiente di apprendimento α_i:

$$\Delta\delta = \alpha_i \cdot v_{max} \cdot T \,^1$$

3. All'istante successivo il master si trova in un punto intermedio definito da:

$$p_i\left(t+1\right) = p_i\left(t\right) + \Delta\delta \cdot \frac{\mathbf{d}}{\|\mathbf{d}\|}$$

4. Vengono dunque aggiornati i parametri SOM α_i e σ_i come descritto in equazione 2.3;

5. Infine il master invia un messaggio ai nodi adiacenti del grafo da propagare a cascata fino ai master successivi come mostrato in figura 3.2. Il contenuto del messaggio è invece riassunto in tabella 3.2.

Le operazioni appena descritte non avvengono ad ogni passo di controllo, ma sono state definite per essere diffuse in maniera periodica ad ogni scatto temporale $T_a = k_a T$, dove k_a rappresenta una *costante di controllo moltiplicativa* stabilita

[1] Più precisamente, $\Delta\delta$ andrebbe moltiplicato anche per il valore assunto dalla funzione di vicinato per il particolare valore di σ_i. Tuttavia, come specificato nella sezione 2.2 e illustrato in figura 2.6, il master della rete è considerato l'equivalente del BMU nelle reti SOM e la funzione di vicinato calcolata sul *vincitore* assume sempre valore unitario: $\Phi\left(u, u, \sigma_u\right) = 1$.

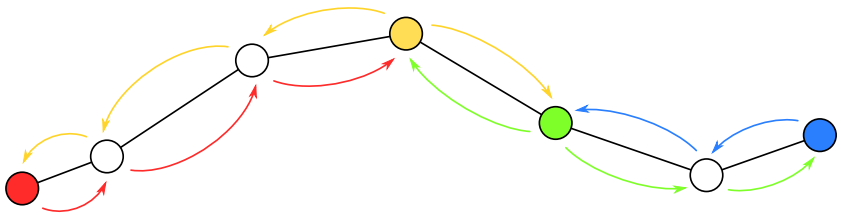


Figura 3.2: Propagazione dei messaggi di *comando* da ciascun nodo master verso gli altri nodi del grafo.

Descrizione	Simbolo
Master	i
Destinazione	d_j
Learning rate	α_i
Neighbour radius	σ_i

Tabella 3.2: Messaggio di *comando* inviato da ciascun master al nodo precedente e a quello successivo nella linea del grafo (o ad uno solo dei due se è il master considerato è $r_i \in \{r_1, r_n\}$).

dall'utente fino a qualche decina di unità[2]. A prescindere da tale valutazione, viene sempre verificata la condizione per cui $T \leq T_a$.

La diffusione del messaggio di *comando* ai follower di cui al punto 5 può essere interrotta per uno dei tre motivi seguenti:

- L'agente che ha ricevuto il messaggio è un master;
- Il nodo i che riceve il messaggio non fa parte del vicinato diretto del master u che lo ha inviato, ovvero $|u - i| \leq \sigma_u$ [3];
- Al follower che riceve l'informazione è stata già assegnata una destinazione

[2]In fase di testing si è fatto variare il parametro k_a in un intervallo compreso da un minimo di 1 fino a un massimo di 20 circa.

[3]Se il nodo i non fa parte del *vicinato* di u vuol dire che la funzione $\Phi(u, i, \sigma_u) = 0$ e che non è stato scelto per avvicinarsi alla destinazione d_j (con la terminologia delle SOM si dice che il nodo non si *specializza* nel riconoscere quel particolare pattern d'ingresso). In tal caso si evince che se il nodo non verrà *attratto* da quel set-point prestabilito, allo stesso modo si comporteranno i nodi che lo seguono nella linea di rete, pertanto è inutile divulgare ulteriormente un messaggio che sarà ignorato.

da seguire da parte di un altro master[4].

Effettivamente, le azioni elencate nei punti precedenti non bastano a mantenere la connettività, poiché i master si muovono indipendentemente dalla posizione dei follower, incorrendo così nello stesso problema riscontrato in [**11**] e descritto in sezione 1.3.3 (vedi figura 1.4).

L'algoritmo di coordinamento presentato, infatti, deve essere opportunamente modificato al punto 3 secondo le regole che verranno illustrate anche per i follower. Si tenga presente inoltre che, al di là delle attività di aggiornamento dei master ad ogni intervallo T_a, tutti gli agenti della MANET si mettono in moto con cadenza regolare T, secondo le regole esposte in sezione 3.5.

3.5 Strategia di movimento

Durante lo studio condotto per la concretizzazione di un piano di coordinamento fra velivoli mobili autonomi, si è notato che la ricerca di una meta ideale da assegnare dinamicamente a ciascun robot affinché non venga interrotto l'insieme dei collegamenti punto-punto non è affatto semplice. Tale complicazione è dovuta alla quantità di casi d'uso crescente che si ottengono adoperando una strategia piuttosto di un'altra.

La soluzione proposta in questa tesi, difatti, fa uso di un insieme di possibili *target* da attribuire a ciascun nodo in funzione sia del suo stato interno, che di quello dei nodi vicini. Tale approccio si è rivelato oltremodo vantaggioso per evitare situazioni di *stallo*, in cui due agenti adiacenti risultano interdetti dal movimento poiché giunti al limite dello spazio di copertura, e per trovare un metodo che permettesse la *prevenzione delle collisioni*.

Partendo da uno stato in cui i master hanno inviato il messaggio di *comando* (*command*) in cui specificano ai follower la destinazione da raggiungere, ogni unità indipendente realizza parallelamente le seguenti azioni:

[4]Come sarà più chiaro nella sezione 3.6, le destinazioni che vengono conferite ai follower non rimangono immutabili durante il processo di coordinamento, ma al contrario sono riviste dinamicamente durante l'evoluzione della rete.

1. Si costruisce un insieme ordinato di *target* in base alla priorità assegnata ad ogni punto. Con *priorità* si intende un criterio di organizzazione dei punti per cui, qualora non sia possibile arrivare al primo target (tralasciandone al momento le cause), si tenta il movimento verso il target successivo, e così via fino all'ultimo della lista. Sia $\Gamma_i = \left\{\gamma_i^{(1)}, \ldots, \gamma_i^{(g)}\right\}$ l'insieme ordinato dei target[5] costruito col metodo seguente:

$$\Gamma_i = \begin{cases} \{d_j\} & \text{se } r_i \text{ è un master} \\ \{d_j,\, p_i^m\} & \text{se } r_i \text{ è un follower} \end{cases} \tag{3.2}$$

   Si sottolinea come nell'uguaglianza 3.2 il termine d_j non sia impostato a priori per i follower, bensì si fa riferimento a quello ricevuto da qualche master nel messaggio *command*; mentre il termine p_i^m indica il punto medio fra il nodo precedente e quello successivo, secondo la relazione:

$$p_i^m = \frac{p_{i-1} + p_{i+1}}{2}$$

2. Si calcola il coefficiente SOM φ_i ad indicare l'influenza della distanza dal master u sul reticolo:

$$\varphi_i = \Phi\left(u, i, \sigma_i\right)\ ^{6}$$

3. Per ogni target $\gamma_i^{(h)}$ nell'insieme Γ_i prelevato in ordine crescente, si mette in atto la procedura che segue:

   (a) Si determinano le coordinate del punto più lontano che può essere raggiunto dal nodo in funzione della posizione dei vicini diretti (eventualmente solo uno dei due se i è un vertice terminale del grafo). Tali coordinate sono calcolate limitando il vettore distanza che unisce p_i e con $\gamma_i^{(h)}$ nella zona di intersezione fra i due cerchi (o sfere) di raggio r con centro nella posizione dei nodi adiacenti. Prendendo in esame ancora una volta il caso bidimensionale è possibile formalizzare il problema valutando la retta passante per i due punti p_i e $\gamma_i^{(h)}$ con equazione

[5] La dicitura $\gamma_i^{(h)}$ sta ad indicare il target del nodo i-esimo con priorità $h = \{1, \ldots, g\}$.
[6] Anche in questo caso σ_i è quello ricevuto da uno dei master più vicini, difatti vale $\sigma_i = \sigma_u$.

parametrica $y = m \cdot x + q$ e i due cerchi centrati in p_{i-1} e p_{i+1}:

$$\begin{cases} y = m \cdot x + q \\ (x - x_{i-1})^2 + (y - y_{i-1})^2 \leq r^2 \\ (x - x_{i+1})^2 + (y - y_{i+1})^2 \leq r^2 \end{cases} \tag{3.3}$$

Dal sistema 3.3 si ottiene un segmento contenuto nella regione di piano descritta dall'intersezione dei due cerchi e delineato dalla posizione della retta. Dei due estremi del segmento così ottenuto, si sceglie l'unico che si trova in direzione del target $\gamma_i^{(h)}$, come illustrato in figura 3.3, sia $\gamma_i^{(h)\prime}$;

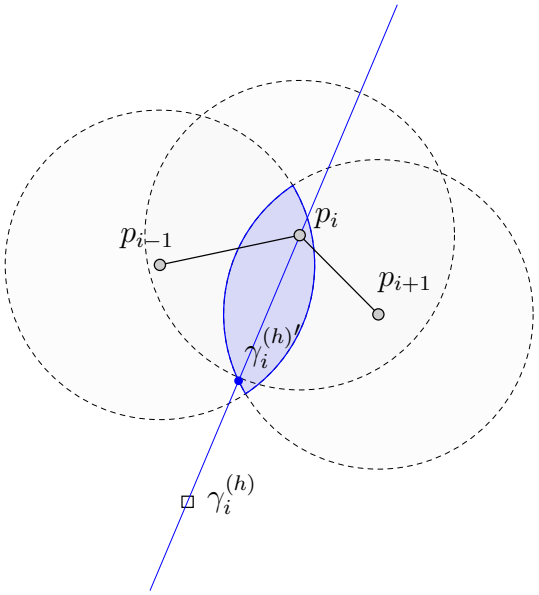


Figura 3.3: Punto limite in cui si può spingere ciascun nodo in funzione della posizione del nodo precedente e di quello successivo nel grafo.

(b) A questo punto si calcola la distanza percorribile all'interno della finestra di controllo $\Delta\delta$ e la variazione angolare $\Delta\psi$ nello stesso periodo, entrambi limitati dai parametri neurali della SOM:

$$\begin{cases} \Delta\delta = \alpha_i \cdot \varphi_i \cdot v_{max} \cdot T \\ \Delta\psi = \alpha_i \cdot \varphi_i \cdot \omega_{max} \cdot T \end{cases}$$

(c) La destinazione e l'orientamento che si ottengono in riferimento al target $\gamma_i^{(h)'}$ considerato sono:

$$\begin{cases} p_i' = p_i + \Delta\delta \cdot \frac{\mathbf{d}}{\|\mathbf{d}\|} \\ \theta_i' = \theta_i + \Delta\psi \cdot \beta \end{cases} \tag{3.4}$$

Nel sistema 3.4 si è indicato con $\mathbf{d}$ il vettore differenza $\gamma_i^{(h)'} - p_i$, mentre con β si intende la variazione fra l'orientamento attuale e l'angolo del vettore $\mathbf{d}$ (direzione di movimento), ovvero $\beta = \theta_i - \angle\mathbf{d}$ [7];

(d) Se il punto p_i' è diverso da p_i allora si esce dal ciclo scegliendo la nuova destinazione e il nuovo orientamento ricavati per il passo di controllo corrente. Viceversa, si incrementa h e si effettuano nuovamente i conti dal punto 3a;

4. Si aggiornano le coordinate e l'orientamento del robot con i nuovi valori ricavati all'iterazione precedente:

$$\begin{cases} p_i\,(t+1) = p_i'\,(t) \\ \theta_i\,(t+1) = \theta_i'\,(t) \end{cases} \tag{3.5}$$

5. Al termine di ogni movimento, vengono informati gli agenti adiacenti della MANET trasmettendo loro lo stato interno come descritto in sezione 3.3;

6. Se l'agente in questione appartiene al sottoinsieme dei master ed è trascorso un periodo di tempo maggiore o uguale a T_a, allora vengono ricalcolati anche i parametri α_i e σ_i ed inviati al resto della rete per propagazione (vedi sezione 3.2).

Le operazioni sopra descritte rappresentano il completamento del caso generale affrontato per il movimento iniziale dei master, il quale può essere completamente sostituito dalle suddette.

[7] Si faccia attenzione che β non è espresso in valore assoluto, bensì il suo segno indica il verso della rotazione. In particolare $\beta > 0$ esprime un verso di rotazione *antiorario* nel piano euclideo, mentre $\beta < 0$ il verso *antiorario*.

Nel sistema 3.3 è stato invece illustrato il nodo fondamentale dell'algoritmo. Indipendentemente dalla destinazione o dal *target intermedio* che ciascun velivolo si prefigge di raggiungere, esso permette di far sì che ad ogni passo di controllo il movimento si concentri sempre all'interno dell'intersezione dei due cerchi (o sfere) costruiti sui nodi vicini. In questo modo, poiché $\|p_i - p_{i-1}\| \leq r$ e $\|p_i - p_{i+1}\| \leq r$, si è sempre certi che la connettività non venga mai interrotta lungo il percorso del grafo.

Il motivo per cui si è scelto di avere un insieme di target Γ_i rispetto ad un unico punto da inseguire è presto detto. Durante le fasi di testing, infatti, si sono verificate situazioni di stallo in cui ogni robot si occupava di raggiungere correttamente la destinazione assegnata ma la rete nel complesso risultava *bloccata* ed impossibilitata al movimento.

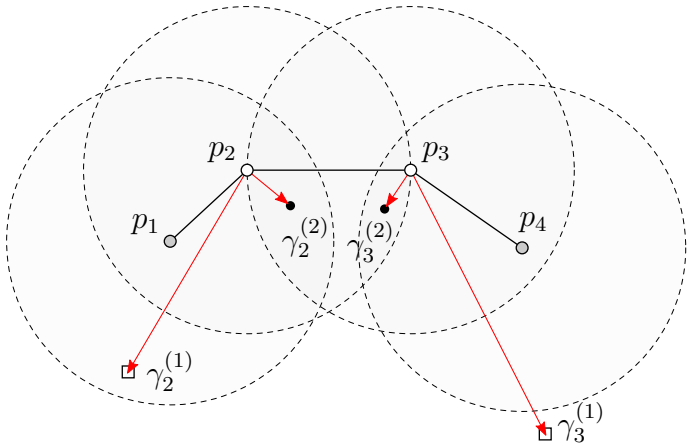


Figura 3.4: *Stallo* del follower causato da una distanza limite fra loro i due al raggio di copertura r.

Un esempio chiarificante è illustrato in figura 3.4. I due follower (in bianco) hanno il compito di raggiungere ognuno la destinazione $\gamma_i^{(1)}$ che hanno ricevuto dal master più vicino (in grigio scuro), però si trovano ad una distanza tale fra loro per cui il moto è fortemente compromesso: per l'appunto non hanno la possibilità di spostarsi ulteriormente poiché si trovano sulla linea di confine dopo la quale uscirebbero dal *cerchio di copertura* dell'altro. Per evitare una simile situazione di *deadlock di movimento*, giacché il target $\gamma_i^{(1)'} = p_i$, si propone di spostare l'attenzione sul punto medio $\gamma_i^{(2)} = p_m^i$ fra i nodi vicini nel grafo,

nel ruolo di target "*temporaneo*". Così facendo, ciascun follower è in grado di sbloccarsi dalla condizione di stallo per almeno un passo di controllo T, cosicché può tentare nuovamente di raggiungere l'obiettivo precedentemente stabilito al passo successivo.

3.6 Definizione dei messaggi

È doveroso soffermarsi brevemente sulla descrizione dei messaggi generati dai diversi agenti e sulle operazioni intraprese una volta ricevuti. Le tipologie di messaggio che possono giungere a ciascun robot sono tre:

- **New position:** si sta informando il nodo che il velivolo autonomo adiacente nel grafo con identificatore univoco i ha cambiato il suo *stato interno* spostandosi nel punto di coordinate p_i e con orientamento θ_i;
- **Command:** è il messaggio inviato da un master e propagato lungo la rete. Un master che riceve questo messaggio lo ignora completamente; i follower invece hanno due alternative:
  - Se è stata già assegnata loro una destinazione il messaggio viene rifiutato per un totale di `RefuseLimit` volte;
  - Quando `RefuseCounter > RefuseLimit` si modificano i parametri SOM del follower in questione[8].

  Indipendentemente dall'aggiornamento di tali parametri, il follower i che ha ricevuto il messaggio dal nodo j inoltra il messaggio al nodo successivo con indice $2i - j$.
- **Arrived:** è anch'esso un messaggio inviato da un master della rete per informare i follower che è giunto a destinazione. I nodi che avevano ricevuto da tale master lo stesso punto obiettivo terminano anch'essi il movimento fintantoché un altro non li "*risveglia*" assegnandogliene uno nuovo. Come il messaggio *command*, anche questo viene ignorato dai master e propagato tra i follower nel percorso.

[8]Nella fase di testing è stato impostato `RefuseLimit = 3`.

Il contenuto di tali messaggi viene brevemente schematizzato in tabella 3.3. Si osservi che l'orientamento fornito dall'agente i-esimo con il messaggio *new position* non viene utilizzato dal nodo ricevente per coordinare il movimento, tuttavia viene lo stesso mantenuto per sviluppi futuri (ad esempio per supportare eventuali vincoli olonomi).

New position	
Mittente	i
Posizione	p_i
Orientamento	θ_i

Command	
Mittente	i
Master	u
Destinazione	d_j
Learning rate	α_u
Neighb. radius	σ_u

Arrived	
Mittente	i
Master	u
Destinazione	d_j

Tabella 3.3: Contenuto dei messaggi scambiati fra i vari nodi della rete.

Capitolo 4

Collision avoidance

Per quanto elaborata e funzionale, la strategia descritta nei capitoli precedenti non tiene conto di eventuali sovrapposizioni fra le coordinate dei vari UAV.

Allo scopo di evitare le collisioni tra quadricotteri, un possibile *workaround* potrebbe essere quello di limitare la *zona di lavoro* di ciascun velivolo ad una diversa altitudine, cosicché i droni abbiano la capacità di muoversi solo all'interno di iperpiani paralleli di altezza variabile. Tuttavia, si tratta certamente di una soluzione assai inefficiente.

In questo capitolo viene descritto un metodo per consentire ai robot di rilevare la presenza degli altri senza conoscerne la posizione esatta, semplicemente dotandoli di sensori aggiuntivi. Utilizzando un equipaggiamento di questo tipo, ognuno di loro deve essere in grado non solo di schivare gli altri quando le traiettorie si intrecciano, ma anche di evitare eventuali ostacoli ambientali che si presentano nel tragitto.

Uno dei vantaggi dell'algoritmo di coordinamento distribuito proposto è la flessibilità. Questo perché è stato ideato in maniera modulare ed è indipendente rispetto al problema del *collision avoidance*; difatti la tecnica proposta in questo capitolo può essere perfettamente sostituita con altri metodi altrettanto validi scelti dall'utente.

4.1 Sensori di prossimità

I sensori di prossimità sono, come suggerisce il nome, dei sensori in grado di percepire ed indicare la presenza di oggetti all'interno di un *campo sensibile* in prossimità del sensore stesso. Possono essere utilizzanti anche per determinare la distanza rispetto ad un oggetto, producendo un segnale in uscita che non è una semplice funzione binaria del tipo *on-off*, ma direttamente proporzionale alla distanza rilevata.

I sensori a cui si farà riferimento nel seguito sono i **sensori ad ultrasuoni** (o *sensori ultrasonici*). Essi fanno uso di un trasduttore elettroacustico di tipo piezoceramico per emettere onde sonore a frequenza elevata, nel range compreso tra 40 e 200 kHz. Il tempo necessario all'onda per incontrare un eventuale oggetto presente all'interno della *portata nominale* e ritornare all'emettitore sotto forma di *eco* fornisce la distanza dell'obiettivo con estrema precisione.

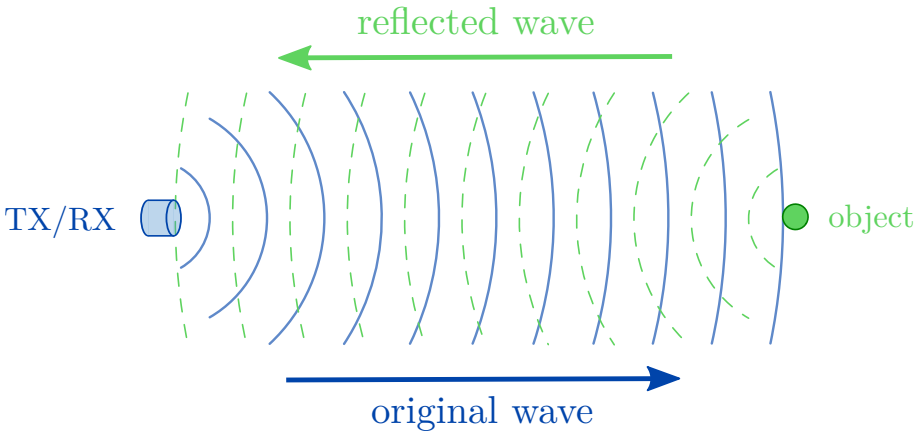


Figura 4.1: Principio di funzionamento dei sensori ad ultrasuoni.

Il vantaggio nell'uso della tecnologia ad ultrasuoni per applicazioni con sensori di prossimità sta nell'accuratezza nettamente superiore rispetto a quella ottenibile con tecnologia induttiva o capacitiva, a parità di dimensione geometrica. Al contempo sono anche immuni ai disturbi elettromagnetici. Nonostante si faccia uso in genere di sensori ottici per la rilevazione di oggetti distanti, i sensori ultrasonici non hanno il problema fondamentale di dipendere dalle caratteristiche

cromatiche superficiali dell'oggetto rilevato e sono in grado di accorgersi anche di superfici trasparenti[1].

Ad ogni modo, il tipo di tecnologia scelta non influisce sul metodo con cui vengono evitati gli ostacoli incontrati, al contrario si fa riferimento solo ad una forma d'onda (*beam*) di tipo conico caratterizzata da una **distanza massima** e da un **angolo di apertura** prefissato che definiscono la zona di accettazione della portata nominale. Qualunque sensore di prossimità che può essere ricondotto ad una funzione caratteristica di questo tipo (vedi figura 4.2) con errore trascurabile, può essere utilizzato secondo le tecniche definite in seguito[2].

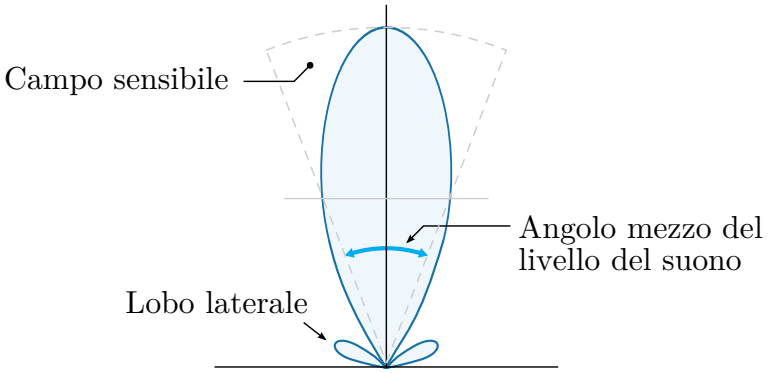


Figura 4.2: Fascio conico di ultrasuoni *reale* (blu) prodotto da un sensore ultrasonico e composto dal lobo principale e da quelli laterali confrontato con il fascio utilizzato nell'algoritmo (grigio tratteggiato).

I parametri caratteristici di ciascun sensore sono schematizzati in tabella 4.1.

Descrizione	**Simbolo**	**Dominio**
Indice del sensore	i	$\mathbb{Z}^+$
Angolo di apertura	ϕ_i	$[0, 2\pi]$
Dimensione del fascio	d_i	$\mathbb{R}^+$
Posizione del sensore	s_i	$[0, 2\pi]$

Tabella 4.1: Parametri distintivi di ciascun sensore in ogni velivolo.

[1] Gli unici oggetti che non possono essere rilevati con sensori ad ultrasuoni sono quelli composti da materiali *fonoassorbenti*.

[2] Si considera una zona rilevabile definita solo dal *main lobe* e approssima ad un cono di raggio ϕ e di altezza d, centrato nel baricentro del quadricottero.

4.2 Equipaggiamento

Come si nota dalla tabella 4.1, ogni drone nella squadra può possedere una sensoristica a bordo diversa da quella degli altri. Lo schema con cui vengono infatti montati i sensori su ciascuna unità autonoma indipendente viene lasciato a discrezione dell'utente, con la sola limitazione che il punto di partenza del fascio corrisponda con il centro di massa del quadricottero stesso[3].

In fase di inizializzazione, ogni nodo è a conoscenza dell'insieme dei sensori S_i che sono stati montati sullo stesso. Con $s_i^{(j)}$ si indica la posizione angolare del sensore j-esimo sul quadricottero di indice i. Nell'esempio in figura 4.3 è stato rappresentato un insieme S_i composto da 8 sensori con stessa dimensione del campo sensibile ($d_i^{(1)} = \cdots = d_i^{(8)}$) e medesimo angolo di apertura pari a $\phi_i^{(j)} = \frac{2}{9}\pi\,\text{rad} = 40°$, distanziati fra loro di 45°, ovvero:

$$S_i = \left\{ s_i^{(1)}, \ldots, s_i^{(8)} \right\} = \left\{ 0, \frac{\pi}{4}, \frac{\pi}{2}, \ldots, \frac{7\pi}{4} \right\} \text{ rad}$$

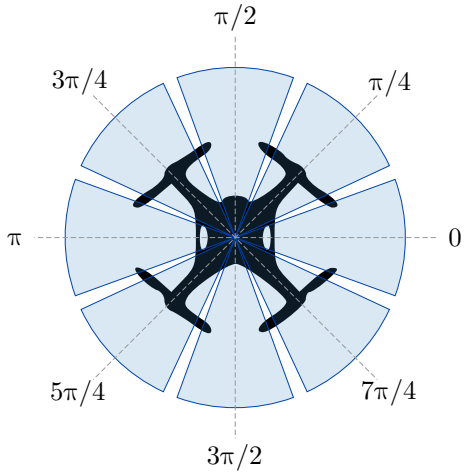


Figura 4.3: Esempio di dotazione sensoriale su un quadricottero.

[3]Si tenga presente che le caratteristiche geometriche del robot non rappresentano un ostacolo alla diffusione dei raggi, in quanto la maggior parte dei sensori ultrasonici hanno la possibilità di settare la distanza minima rilevabile entro la quale scartare gli effetti dovuti ad oggetti troppo vicini.

4.3 Obstacle detection

Purtroppo i sensori di prossimità più comuni permettono di determinare la posizione di un oggetto all'interno della portata nominale esclusivamente in termini di distanza, ma non di angolazione. Questo vuol dire che un ostacolo può trovarsi sia sul limite del campo sensibile che al centro, senza che il quadricottero abbia la possibilità di distinguere fra questi due casi.

La soluzione che viene proposta nel seguito della trattazione si basa sul fatto che ogni sensore "*vede*" gli oggetti solo all'interno di una determinata zona. Partendo da questo presupposto, ogni qualvolta un sensore rileva un oggetto si cerca di attuare una forma di *repulsione* in verso opposto alla posizione $s_i^{(j)}$ del sensore stesso e direttamente proporzionale alla distanza rilevata.

Per chiarire meglio sia il problema che la soluzione presentata si fa l'esempio di un robot con 8 sensori di prossimità adiacenti con medesima apertura di 45° che permettono di coprire tutto l'angolo giro sul piano bidimensionale. Si suppone che siano presenti anche due ostacoli come illustrato in figura 4.4.

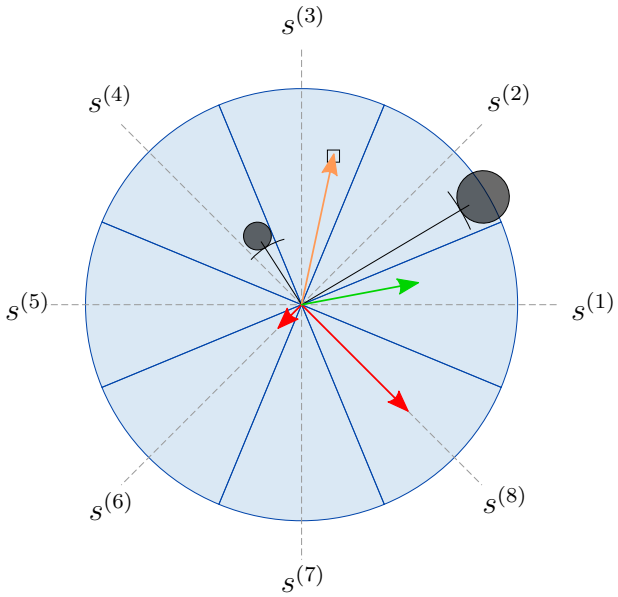


Figura 4.4: Esempio di *obstacle detection* e *collision avoidance* in un quadricottero con 8 sensori identici.

Il drone ha l'obiettivo raggiungere il punto rappresentato dal quadrilatero equilatero, il cui vettore distanza è descritto dal vettore in arancione. Una volta che i sensori 2 e 4, rispettivamente in posizione $s^{(2)}$ e $s^{(4)}$, rilevano la presenza di ostacoli, vengono calcolati i *vettori di repulsione* (indicati in rosso) corrispettivi in funzione della distanza dal centro di massa, siano $\mathbf{r}_2$ ed $\mathbf{r}_4$.

In generale, se la distanza rilevata dal sensore j-esimo è pari a δ_j, allora la forma di tali vettori con partenza nel baricentro velivolo viene regolata dalla formula:

$$\mathbf{r}_j = \begin{cases} \|\mathbf{r}_j\| = d^{(j)} - \delta_j \\ \angle\mathbf{r}_j = s^{(j)} + \pi + \left(\theta - \frac{\pi}{2}\right) \end{cases} \tag{4.1}$$

Nel sistema 4.1 si è indicato con $d^{(j)}$ la distanza del campo di attivazione del sensore j-esimo (supposta uguale per tutti). Questo vuol dire che il modulo del vettore di repulsione viene disciplinato dalla distanza letta dal sensore in maniera proporzionale, in particolare più è lontano l'oggetto, minore sarà la sua norma. Il termine $\left(\theta - \frac{\pi}{2}\right)$ sta ad indicare che la posizione di ogni sensore è definita rispetto all'orientamento del drone con uno sfasamento di 90°.

Determinati tutti gli $\mathbf{r}_j$ (tenendo presente che quando non viene individuato alcun ostacolo il modulo di $\mathbf{r}$ è nullo) si calcola la *somma vettoriale* tra i diversi vettori di repulsione ed il vettore che parte dalla posizione dell'agente e arriva ad una delle destinazioni $\gamma^{(h)\prime}$, come definito in sezione 3.5. Il risultato finale è mostrato in figura 4.1 attraverso il vettore in verde.

Per essere precisi, occorre modificare l'algoritmo di coordinamento affinché supporti un allontanamento graduale dagli ostacoli rilevati ad ogni passo di controllo T. In particolare, si aggiorna la prima equazione del sistema 3.4 come segue:

$$p_i{}' = p_i + \Delta\delta \cdot \frac{\mathbf{d}}{\|\mathbf{d}\|} + \sum_k \mathbf{r}_k \quad , \quad \text{con } \mathbf{d} = \gamma_i^{(h)\prime} - p_i \tag{4.2}$$

4.4 Limitazioni al movimento

La strategia appena descritta permette di aggirare gli ostacoli continuando ad inseguire la destinazione preposta. Si tenga presente che la direzione dei vettori di repulsione è fortemente legata all'orientamento dell'UAV, poiché definita con riferimento statico rispetto al versante anteriore.

Questa situazione porta però ad uno spiacevole inconveniente quando si ha un sensore posizionato nello stesso verso dell'orientamento θ_i coincidente con la direzione del moto. In questo caso d'uso particolare, un eventuale ostacolo rilevato rallenterebbe il movimento fino ad un punto in cui il robot si ferma perché $\mathbf{r}_j$ lo limita nel proseguire.

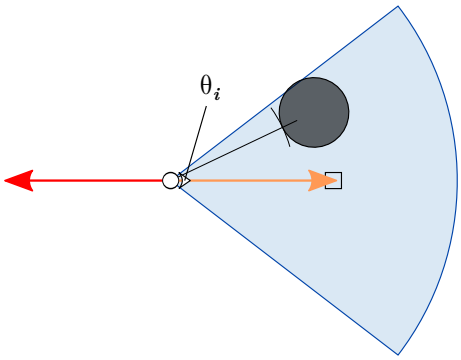


Figura 4.5: Problema dello stallo nel movimento nella strategia di *collision avoidance* proposta.

In figura 4.5 si può notare come la somma fra il vettore di movimento al passo di controllo e quello di repulsione assuma valore nullo. In pratica i due hanno stessa direzione e stesso modulo ma verso opposto[4], nonostante l'ostacolo non sia perfettamente allineato alla traiettoria. Tale intoppo è dovuto al fatto che i sensori considerati non forniscono un metodo per rilevare l'angolo di incidenza con cui incontrano gli oggetti.

[4]Si faccia attenzione che nel raggiungere il *target temporaneo* ogni nodo si può muovere al massimo da una velocità prefissata v_{max}. Difatti, le destinazioni considerate in questo capitolo e schematizzate con dei quadrati sono da intendersi "per questo passo di controllo". Per l'appunto, i vettori di colore arancione in figura 4.4 e 4.5 hanno modulo massimo pari a $\Delta\delta = \alpha_i \cdot \varphi_i \cdot v_{max} \cdot T$ come si evince anche dall'equazione 4.2.

Si osservi che la situazione di stallo così ottenuta si verifica solo se la posizione del sensore, l'orientamento del velivolo e il vettore di movimento sono allineati entro un certo margine. Se così non fosse, infatti, il drone sarebbe in grado di accorgersi dell'ostacolo sfruttando un altro sensore con cui è equipaggiato, ottenendo in questo modo un vettore di repulsione con angolo differente.

Proprio grazie a questa considerazione si definisce una soluzione pratica e poco invasiva: poiché l'obiettivo è far ruotare il quadricottero, si inserisce nell'insieme dei target Γ_i una nuova destinazione descritta dall'equazione 4.3.

$$\gamma_i^{(h+1)} = p_i + \mathbf{v} \qquad \text{con} \begin{cases} \|\mathbf{v}\| = d_i^{(j)} \\ \angle\mathbf{v} = \theta_i + \frac{\pi}{2} \cdot \mathcal{B}'\left(\frac{1}{2}\right) \end{cases} \tag{4.3}$$

Dove $\mathcal{B}'\left(\frac{1}{2}\right)$ è una distribuzione uniforme discreta di Bernoulli con supporto $\{-1, 1\}$, ad indicare che si ha il 50% di possibilità di scegliere un punto a destra o a sinistra di p_i rispetto al suo orientamento θ_i. In questo modo l'uguaglianza 3.2 che descrive l'insieme Γ_i per ogni nodo può essere modificata come segue:

$$\Gamma_i = \begin{cases} \{d_j,\, p_i + \mathbf{v}\} & i = master \\ \{d_j,\, p_i^m,\, p_i + \mathbf{v}\} & i = follower \end{cases} \tag{4.4}$$

In ultima analisi, è doveroso precisare che non sempre la condizione di fattibilità descritta nella sezione 3.2 è valida. Essa è sicuramente applicabile in un contesto privo di ostacoli ambientali. Tuttavia, in presenza di un ambiente costellato di impedimenti fisici, si potrebbero presentare particolari configurazioni topologiche delle destinazioni (e degli ostacoli), tali da determinare una distorsione naturale del grafo. In questi casi non può essere garantita una copertura dell'insieme dei set-point con il numero predefinito di nodi disponibili.

In figura 4.6 viene rappresentato in maniera grafica un esempio della questione appena citata: il numero di nodi assegnato avrebbe la capacità di raggiungere tutti i punti obiettivo per mezzo dei master (in grigio) necessari a portare a termine la missione, ma poiché sono presenti degli scogli paesaggistici ineliminabili, l'ultimo agente non è in grado di estendere la rete fino alla destinazione assegnatagli.

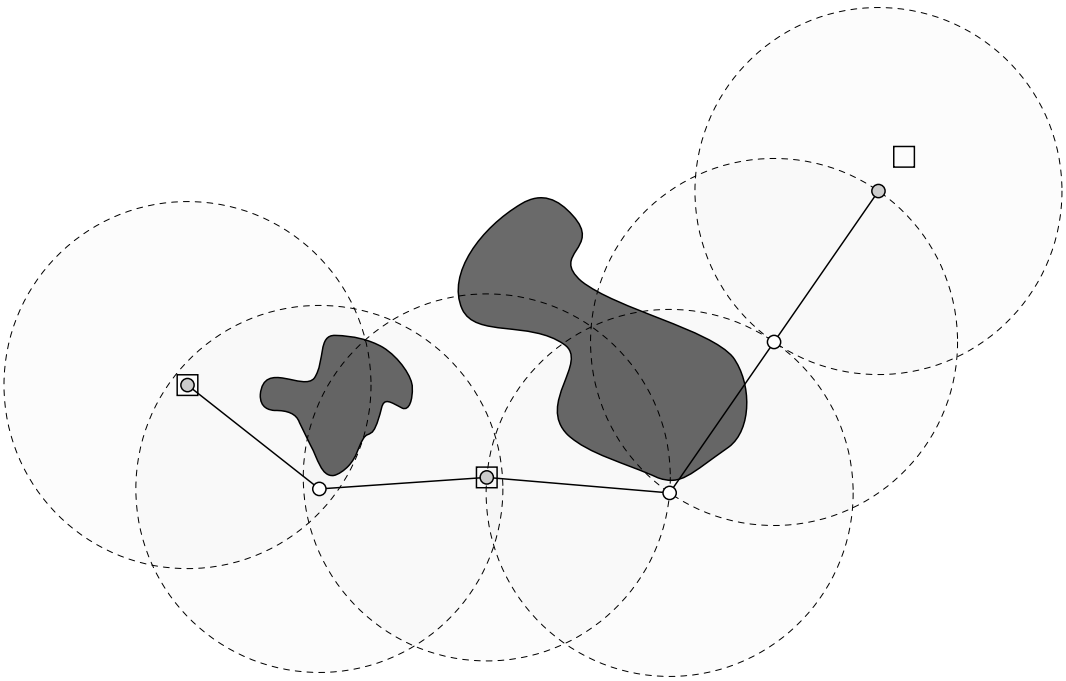


Figura 4.6: Esempio di conformazione dell'ambiente che non ammette soluzione di copertura, nonostante sia verificata la *condizione di fattibilità preliminare*.

Capitolo 5

Ambiente di simulazione

In questo capitolo verrà illustrato l'ambiente di testing che è stato sviluppato per permettere di osservare l'evoluzione del sistema multi-robot durante il perseguimento in simulazione del compito di rete preposto.

Per la realizzazione dell'interfaccia si è scelto di utilizzare una tecnologia facilmente rintracciabile sia nei moderni *personal computer* che in altri dispositivi meno elaborati (quali smartphone o tablet). In particolare, si è deciso di programmare l'ambiente di simulazione in linguaggio Javascript e HTML5/CSS3. In questo modo, la definizione dei pulsanti e delle routine collegate ad essi è stata resa particolarmente semplice, grazie soprattutto al supporto offerto dal browser. L'intero processo di debugging è stato messo in pratica attraverso il *Chrome Developer Tools*.

Per quanto riguarda invece la rappresentazione dei nodi, sono state seguite le linee guida del *World Wide Web Consortium*, che definiscono la tecnologia *Scalable Vector Graphics*, meglio nota con l'abbreviazione *SVG* **[15]**. La semplicità di questo metalinguaggio sta nella sua derivazione dall'XML e quindi nella possibilità di aggiornare dinamicamente gli elementi inclusi nel DOM (*Document Object Model*). Così facendo, si ha il vantaggio di non dover rinnovare drasticamente l'interfaccia ad ogni cambiamento, come accade con i più comuni strumenti di grafica computerizzata, bensì è sufficiente modificare gli attributi specifici dei singoli tag per ottenere il risultato desiderato.

In seguito, si è deciso di abbandonare il suddetto approccio per optando per un'alternativa che facesse uso di una rappresentazione spaziale più simile a quella terrestre. Pertanto, si è realizzato un nuovo ambiente di testing che fa uso delle *Google Maps API* **[16]** e che permette di definire il posizionamento dei droni attraverso le specifiche di longitudine e latitudine. Nonostante la rappresentazione delle immagini sia celata dietro elementi di tipo *canvas*, vi è comunque la possibilità di disegnare oggetti sulla mappa sfruttando il metodo vettoriale.

5.1 Trasformazione di coordinate

Rispettando l'approccio modulare già utilizzato per definire la strategia di coordinamento, si definisce una struttura dati intermedia che modella la posizione delle singole unità. Più che una classe, si definisce un'interfaccia di nome `Coordinates` con i seguenti metodi:

- `Distance(a, b):` determina la distanza fra i punti `a` e `b`;
- `MiddlePoint(a, b):` restituisce il punto intermedio fra `a` e `b`;
- `Bering(a, b):` calcola l'angolo (*bearing*[1]) della retta passante per le coordinate dei punti `a` e `b`;
- `Destination(a,` θ`, d):` determina la destinazione calcolata partendo dalle coordinate di `a` con un angolo di partenza θ percorrendo una distanza pari a `d` in linea retta;
- `GoogleMaps():` trasforma le coordinate correnti in un formato utilizzabile dalle API di supporto.

Quanto descritto nel seguito fa riferimento alla specifica implementazione dell'interfaccia `Coordinates` utilizzata per il simulatore. Ciascuna coordinata è delineata dall'insieme di variabili schematizzate in tabella 5.1.

[1]Termine utilizzato in navigazione per indicare l'angolo in direzione del moto.

Descrizione	Simbolo	Dominio	Un. misura
Latitudine	ϕ	$[0, 2\pi]$	radianti [rad]
Longitudine	λ	$[0, 2\pi]$	radianti [rad]
Altitudine	h	$\mathbb{R}$	metri [m]

Tabella 5.1: Attributi identificativi della classe `Coordinates`.

5.1.1 Sistema considerato

Il motivo principale per cui è necessario definire i metodi precedentemente esposti è legato al fatto che il sistema preso in considerazione è quanto più simile a quello terrestre. Lo standard *World Geodetic System 1984* (WGS84) è un sistema di riferimento geocentrico non inerziale basato su un insieme coerente di costanti e parametri che descrivono la dimensione della Terra, la forma, la gravità e i campi geomagnetici. È il modello a cui fa riferimento anche il sistema GPS (*Global Positioning System*) gestito dal governo degli Stati Uniti[2], ed è basato su una forma planetaria terrestre di tipo *ellissoidale*, le cui caratteristiche sono mostrate in tabella 5.2.

Descrizione	Simbolo	Valore
Semiasse maggiore	a	$6\,378\,137.0$ m
Ellitticità terrestre	$1/f$	$298.257\,223\,563$
Velocità angolare nominale media	ω	$7\,292\,115 \times 10^{-11}$ rad/s
Costante gravitazionale geocentrica	GM	$3\,986\,004.418 \times 10^{8}$ m^3/s^2

Tabella 5.2: Coefficienti del sistema *WGS84*.

L'adozione di questo particolare sistema di riferimento, unitamente ai metodi che saranno esplicitati nelle sezioni seguenti, rende possibile l'adattamento in tempi brevi dell'algoritmo di coordinamento proposto e dell'intero lavoro svolto per l'ambiente di simulazione, così da facilitare il passaggio alla fase di testing direttamente sui velivoli autonomi.

Nel seguito si indicherà con $R = a$ il raggio della Terra a livello dell'equatore e con $\varepsilon = \sqrt{f\,(2 - f)} = 298.257\,223\,563$ la sua eccentricità.

[2]Il governo americano è anche responsabile della definizione del WGS84 per mezzo del *National Geospatial-Intelligence Agency* (NGA) del dipartimento della difesa.

5.1.2 Ortodromia

In geometria sferica, l'*ortodromia* rappresenta l'arco di circonferenza massima (*great-circle path*) che unisce due punti nella circonferenza costruita intersecando la superficie della sfera con un piano passante per il suo centro.

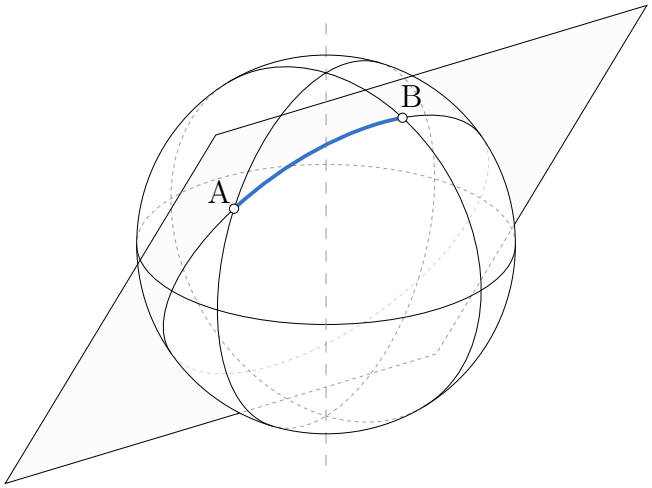


Figura 5.1: Ortodromia tra i punti A e B di una sfera.

Distanza Per determinare la distanza fra due punti $P_1 = (\phi_1, \lambda_1)$ e $P_2 = (\phi_2, \lambda_2)$ sulla *circonferenza massima* di raggio R non si fa uso della *formula dell'emisenoverso*:

$$\text{haversin}(\theta) = \frac{\text{versin}(\theta)}{2} = \frac{1 - \cos(\theta)}{2} = \sin^2\left(\frac{\theta}{2}\right)$$

Essa è definita come la metà del senoverso e rappresenta un esempio di funzione mal-condizionata quando i due punti sono fortemente antipodali. Per questo motivo si ricorre alla formula di Vincenty **[17]**, largamente usata in geodesia per via dell'accuratezza di 0.5 mm (0.020") che si ottiene anche quando viene applicata all'ellissoide terrestre. Siano $\Delta\phi$ e $\Delta\lambda$ rispettivamente la differenza di latitudine e di longitudine fra i due punti P_1 e P_2, allora la distanza d fra essi è data da:

$$d = R \cdot \arctan\left(\frac{\sqrt{(\cos\phi_2 \cdot \sin\Delta\lambda)^2 + (\cos\phi_1 \cdot \sin\phi_2 - \sin\phi_1 \cdot \cos\phi_2 \cdot \cos\Delta\lambda)^2}}{\sin\phi_1 \cdot \sin\phi_2 + \cos\phi_1 \cdot \cos\phi_2 \cdot \cos\Delta\lambda}\right)$$

Rotta iniziale e finale Poiché sulla circonferenza massima l'angolo di incidenza con i meridiani varia in funzione della posizione, si forniscono due metodi per calcolare il *bearing* iniziale e quello finale su una linea ortodromica.

$$\theta_{initial}(P_1, P_2) = \arctan\left(\frac{\cos\phi_2 \cdot \sin\Delta\lambda}{\cos\phi_1 \cdot \sin\phi_2 - \sin\phi_1 \cdot \cos\phi_2 \cdot \cos\Delta\lambda}\right)$$
$$\theta_{final}(P_1, P_2) = \theta_{initial}(P_2, P_1) + \pi$$

Destinazione Dato un punto di iniziale $P = (\phi, \lambda)$ e l'angolo di partenza θ espresso in radianti, il punto di arrivo P_d dopo aver percorso una distanza pari a d è dato da:

$$P_d = \begin{cases} \phi_d = \arcsin\left(\sin\phi \cdot \cos\frac{d}{R} + \cos\phi \cdot \sin\frac{d}{R} \cdot \cos\theta\right) \\ \lambda_d = \lambda + \text{atan2}\left(\sin\theta \cdot \sin\frac{d}{R} \cdot \cos\phi,\ \cos\frac{d}{R} - \sin\phi \cdot \sin\phi_d\right) \end{cases}$$

Con $\text{atan2}(y, x)$ si intende una funzione offerta dalla maggior parte dei linguaggi di programmazione che, oltre a restituire l'angolo di un vettore $(x, y) \neq (0, 0)$, permette di determinare anche in quale dei quattro quadranti è posizionato. In pratica atan2 riesce a calcolare il risultato di $\arctan\left(\frac{y}{x}\right)$ e ad esplicitarlo esattamente nell'intervallo $(-\pi, \pi]$.

Qualora il linguaggio di programmazione utilizzato non fornisca quest'ultimo metodo, si può definire la seguente realizzazione pratica:

$$\text{atan2}(y, x) = \begin{cases} 0 & y = 0 \\ \arccos\left(\frac{x}{\sqrt{x^2 + y^2}}\right) & y > 0 \\ -\arccos\left(\frac{x}{\sqrt{x^2 + y^2}}\right) & y < 0 \end{cases}$$

Punto medio Il punto medio $P_m = (\phi_m, \lambda_m)$ su un'ortodromia delimitata dai punti P_1 e P_2 viene calcolato come descritto in equazione 5.1.

$$
\begin{aligned}
B_x &= \cos\phi_2 \cdot \cos\Delta\lambda \\
B_y &= \cos\phi_2 \cdot \sin\Delta\lambda \\
P_m &= \begin{cases} \phi_m = \operatorname{atan2}\left(\sin\phi_1 + \sin\phi_2,\ \sqrt{(\cos\phi_1 + B_x)^2 + {B_y}^2}\right) \\ \lambda_m = \lambda_1 + \operatorname{atan2}\left(B_y,\ \cos\phi_1 + B_x\right) \end{cases}
\end{aligned} \tag{5.1}
$$

5.1.3 Lossodromia

Le *linee lossodromiche* sono segmenti sulla superficie terrestre che tagliano tutti i meridiani con lo stesso angolo. Si tratta di un concetto di importanza fondamentale in navigazione. Infatti, sebbene una lossodromia non rappresenti il percorso più breve tra due punti (come si nota dalla figura 5.2), essa risulta molto utile nelle carte nautiche in cui i meridiani sono raffigurati con rette parallele.

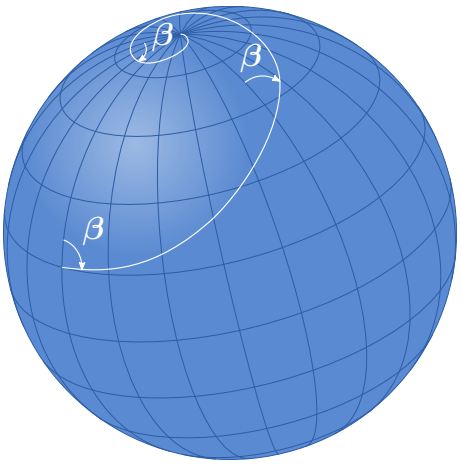


Figura 5.2: Lossodromia tra due punti sulla superficie terrestre.

Nonostante le formule studiate per determinare la distanza, l'angolo di partenza e il punto medio su un percorso lossodromico non vengano utilizzate nell'ambiente di testing, si è ritenuto opportuno renderle disponibili all'utente per eventuali usi futuri.

Distanza Dati due punti $P_1 = (\phi_1, \lambda_1)$ e $P_2 = (\phi_2, \lambda_2)$, la distanza d che li separa seguendo un percorso lossodromico è data da:

$$\begin{aligned}\Delta\psi &= \ln\left(\frac{\tan\left(\frac{\pi}{4}+\frac{\phi_2}{2}\right)}{\tan\left(\frac{\pi}{4}+\frac{\phi_1}{2}\right)}\right)\\ q &= \frac{\Delta\phi}{\Delta\psi}\ ^{3}\\ d &= R\cdot\sqrt{\Delta\phi^2+q^2\cdot\Delta\lambda^2}\end{aligned}$$

Con $\Delta\phi$ e $\Delta\lambda$ si è indicato rispettivamente la differenza tre le due latitudini e le due longitudini. Affinché la lossodromia considerata sia quella più breve, occorre modificare preventivamente il valore di $\Delta\lambda$ se il suo valore in modulo è maggiore di π (si calcola sull'antimeridiano):

$$\Delta\lambda = \begin{cases}\Delta\lambda - 2\pi & \Delta\lambda > 0\\ \Delta\lambda + 2\pi & \Delta\lambda < 0\end{cases}$$

Rotta iniziale e finale Per definizione, il *bearing* lungo una linea lossodromica è costante durante tutto il percorso:

$$\begin{aligned}\Delta\psi &= \ln\left(\frac{\tan\left(\frac{\pi}{4}+\frac{\phi_2}{2}\right)}{\tan\left(\frac{\pi}{4}+\frac{\phi_1}{2}\right)}\right)\\ \theta &= \operatorname{atan2}\left(\Delta\lambda, \Delta\psi\right)\end{aligned} \tag{5.2}$$

Punto medio Le coordinate del punto medio $P_m = (\phi_m, \lambda_m)$ si determinano mediante le formule in equazione 5.3:

[3] Sulle rette di intersezione tra il piano dell'orizzonte astronomico e il piano dell'equatore celeste, dette *linee Est-Ovest*, il valore di $\Delta\psi$ è nullo, perciò il valore di q deve essere calcolato attraverso $q = \cos\phi_1$.

$$\begin{aligned}
\phi_m &= \frac{\phi_1 - \phi_2}{2} \\
f_1 &= \tan\left({}^{\pi}/_{4} + {}^{\phi_1}/_{2}\right) \\
f_2 &= \tan\left({}^{\pi}/_{4} + {}^{\phi_2}/_{2}\right) \\
f_m &= \tan\left({}^{\pi}/_{4} + {}^{\phi_m}/_{2}\right) \\
\lambda_m &= \frac{(\lambda_2 - \lambda_1)\ln(f_m) + \lambda_1 \ln(f_2) - \lambda_2 \ln(f_1)}{{}^{f_2}/_{f_1}}
\end{aligned} \tag{5.3}$$

5.1.4 Proiezioni

Attraverso un'attenta analisi degli strumenti forniti nelle sezioni precedenti e in riferimento al sistema di coordinamento distribuito presentato in questa tesi, si può notare che non sono stati presentati tutti i mezzi per poter calcolare i target intermedi definiti dall'algoritmo su una superficie sferica.

Data la complessità delle espressioni della geometria sferica, si è deciso invece di fornire l'utente di tre diversi metodi di *proiezione cartografica cilindrica*. Le proiezioni sono il risultato di trasformazioni matematiche di coordinate geografiche in punti espressi nel piano cartesiano. Sono largamente utilizzate in cartografia per rappresentare su un piano la superficie terrestre, nonostante abbiano un intrinseco ed ineliminabile fattore di distorsione.

Proiezione equirettangolare Rappresenta la più semplice ed immediata forma di proiezione cilindrica di una mappa su un piano. I meridiani sono raffigurati da linee verticali equispaziate pari alla distanza fra i paralleli, il che causa una forte distorsione orizzontale direttamente proporzionale alla distanza con l'equatore.

$$\begin{cases} x = R \cdot \cos\phi \cdot \lambda \\ y = R \cdot \phi \end{cases} \quad \Leftrightarrow \quad \begin{cases} \phi = \frac{y}{R} \\ \lambda = \frac{x}{R \cdot \cos\phi} \end{cases}$$

Proprio a causa di queste deformazioni spaziali (vedi figura 5.3) ha poco uso nelle mappe catastali e nella navigazione. Tuttavia, se le coordinate di due punti considerati non sono molto lontane dall'equatore e la distanza fra loro è breve, il vantaggio nell'utilizzo di questa proiezione risiede nell'immediatezza del calcolo.

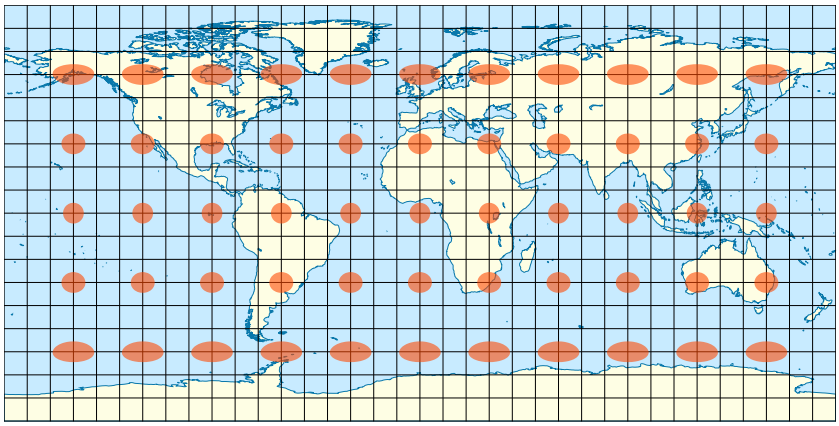


Figura 5.3: Indicatore di *Tissot* della proiezione equirettangolare.

Proiezione di Mercatore La proiezione cartografica presentata nel 1569 dall'astronomo e matematico fiammingo Gerhard Kremer (noto in latino come *Gerardus Mercator* e poi italianizzato come *Gerardo Mercatore*) è diventata lo standard d'utilizzo in ambiente nautico grazie alla capacità di rappresentare linee lossodromiche come semplici segmenti che intersecano i meridiani con lo stesso angolo. Una variante di questa proiezione viene utilizzata dal servizio di visualizzazione online di Google Maps.

Nonostante raffiguri un tipo di proiezione *conforme*, la scala di rappresentazione dei paralleli nella proiezione di Mercatore presenta un aumento esponenziale ed infinito dall'equatore ai poli. Proprio per questo motivo, si assume che tutte le zone con una latitudine pari o superiore a 85° abbiano una deformazione tale da renderne infattibile la rappresentazione sulla mappa, perciò vengono di fatto troncate. Per le zone polari si può utilizzare la proiezione azimutale UPS (*Universal Polar Stereographic*), che non verrà trattata nel seguito.

Nella formula per il cambio di coordinate in equazione 5.4 si fa riferimento alla generalizzazione per l'ellissoide WGS84 come spiegato in sezione 5.2.

$$\begin{cases} x = R \cdot \lambda \\ y = R \cdot \ln\left[\tan\left(\frac{\pi}{4} + \frac{\phi}{2}\right) \cdot \left(\frac{1 - \varepsilon \sin\phi}{1 + \varepsilon \sin\phi}\right)^{\frac{\varepsilon}{2}}\right] \end{cases} \tag{5.4}$$

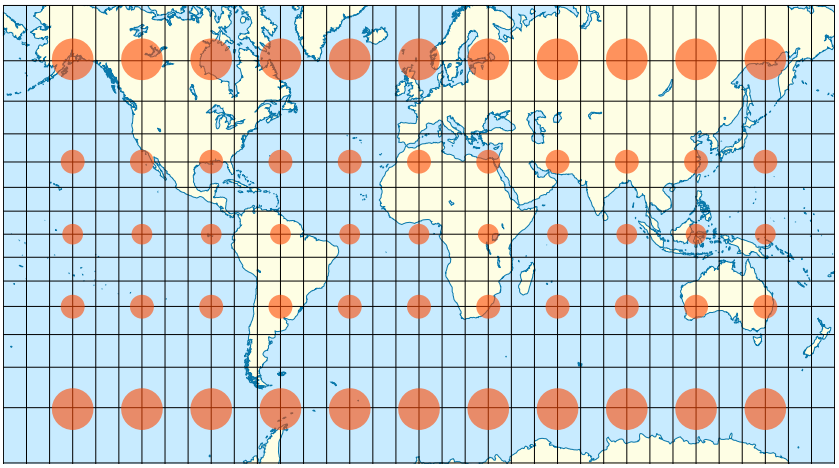


Figura 5.4: Indicatore di *Tissot* della proiezione di Mercatore.

Variante della proiezione di Mercatore La proiezione che viene utilizzata dal simulatore è una versione personalmente modificata della *proiezione secante di Mercatore*. A differenza delle comuni proiezioni cilindriche, in cui le proporzioni all'equatore rispecchiano fedelmente le distanze geografiche, con una distorsione crescente man mano che ci si allontana da esso, è stato definito un metodo che permettesse di ottenere una maggior precisione per le piccole e medie distanze indipendentemente dalla latitudine considerata.

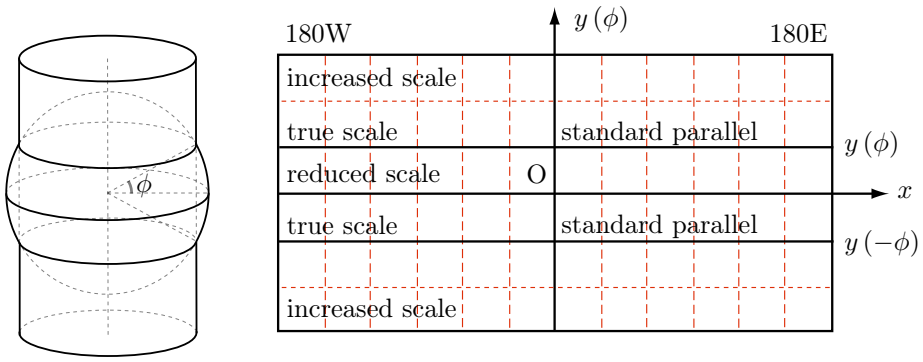


Figura 5.5: Proiezione cilindrica di Mercatore, versione con secante a ϕ.

Il concetto cardine su cui si basa questa proiezione è regolato dall'uso che se ne fa nell'algoritmo. Allo stato attuale, si attua un cambio di sistema di riferimento, da sferico a planare, ogni qualvolta si debbano operare calcoli troppo complessi (o semplicemente non noti) in geometria sferica. Per piccole distanze, dunque, la

proiezione è sicuramente la soluzione ottimale, soprattutto se nella zona geografica considerata si ha una deformazione spaziale praticamente nulla.

Nella variante di Mercatore a secante si "riduce" il raggio del cilindro di proiezione in funzione di un valore di latitudine noto: $r = R \cdot \cos\phi$. In questo modo, è possibile perpetuare la conversione da (ϕ, λ) a (x, y) con tale variante (opportunamente aggiornata per l'ellissoide terrestre WGS84), in riferimento al *parallelo standard* definito da ϕ invece che dall'equatore.

$$\begin{cases} x = R \cdot \cos\phi \cdot \lambda \\ y = R \cdot \cos\phi \cdot \ln\left[\tan\left(\frac{\pi}{4} + \frac{\phi}{2}\right) \cdot \left(\frac{1 - \varepsilon\sin\phi}{1 + \varepsilon\sin\phi}\right)^{\frac{\varepsilon}{2}}\right] \end{cases} \tag{5.5}$$

Purtroppo, invertire l'equazione 5.5 non è assolutamente banale. Perciò, per poter riportare il sistema in coordinate geografiche, è necessaria la conoscenza del parallelo più vicino ϕ' alla zona di lavoro (o di uno dei punti nelle immediate vicinanze). Ciononostante, questo metodo è risultato molto più preciso della semplice proiezione di Mercatore, soprattutto in condizioni di forte distorsione spaziale ($\phi > 50\,°$).

$$\begin{cases} \phi = 2 \cdot \arctan\left[\exp\left(\frac{y}{R \cdot \cos\phi'}\right) \cdot \left(\frac{1 - \varepsilon\sin\phi'}{1 + \varepsilon\sin\phi'}\right)^{-\frac{\varepsilon}{2}}\right] \\ \lambda = \frac{x}{R \cdot \cos\phi'} \end{cases} \tag{5.6}$$

5.2 Simulazione di movimento

Affinché un simulatore sia propriamente detto, ci si aspetta che il movimento del singolo nodo non avvenga istantaneamente nella posizione voluta, ma in maniera crescente e graduale. Per rendere l'applicazione più realistica, si è scelto di implementare un filtro passa-basso per riprodurre il comportamento degli attuatori, sia per il movimento che per la rotazione.

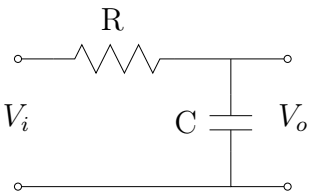


Figura 5.6: Filtro passa-basso.

Partendo dallo schema del filtro passa basso convenzionale, si determina il rapporto tra la tensione in uscita e quella in ingresso (*guadagno*):

$$\begin{cases} V_o = \dfrac{1}{s\mathrm{C}} \cdot I \\ I = \dfrac{V_i - V_o}{\mathrm{R}} \end{cases} \quad \Rightarrow \quad G(s) = \frac{V_o}{V_i} = \frac{1}{s\mathrm{RC}+1} = \frac{\alpha}{s+\alpha}\,,$$

dove si è espresso con $\alpha = \frac{1}{\mathrm{RC}}$ e con $G(s)$ la funzione di trasferimento del sistema nel dominio del tempo continuo. Per riportarsi in un ambito discretizzato, occorre derivare la trasformata Z della funzione ottenuta[4]:

$$G(z) = \frac{Y(z)}{X(z)} = Z\left[\left(\frac{1-e^{-st}}{s}\right)G(s)\right] = \left(1-z^{-1}\right)\cdot Z\left[\frac{G(s)}{s}\right]$$

Ponendo $\rho = e^{\alpha T}$ e utilizzando le tabelle di trasformazione, si ottiene:

$$G(z) = \frac{z-1}{z}\cdot\frac{(1-\rho)\,z}{(z-1)\,(z-\rho)} = \frac{1-p}{z-p} = \frac{(1-\rho)\,z^{-1}}{1-\rho z^{-1}} = \frac{Y(z)}{X(z)}$$

L'ultimo passo consiste nell'esplicitare la funzione d'uscita rispetto all'ingresso, per poi determinare la funzione a tempo discreto desiderata:

$$\begin{gathered} Y(z) = X(z)\cdot\frac{(1-\rho)\,z^{-1}}{1-\rho z^{-1}} = X(z)\cdot(1-\rho)\,z^{-1} + Y(z)\cdot\rho z^{-1} \\ \Downarrow \\ y(k) = (1-\rho)\cdot x\,(k-1) + \rho\cdot y\,(k-1) \end{gathered} \tag{5.7}$$

Il parametro ρ rappresenta il polo del sistema analizzato. Intuitivamente, si

[4]Si ricorda che per definizione vale $z \triangleq e^{sT}$.

può leggere l'equazione 5.7 ottenuta dicendo che l'uscita al passo di controllo k dipende sia dal valore atteso $x(k-1)$, che dal valore assunto al passo precedente $y(k-1)$ in maniera complementare secondo $\rho \in [0,1]$.

Nell'ambiente di simulazione sono presenti due sistemi di controllo, i quali fanno riferimento ciascuno ad un polo diverso:

- **Movimento:** per riprodurre lo spostamento naturale lungo le tre coordinate latitudine, longitudine e altitudine;
- **Rotazione:** permette di ricreare un comportamento in rotazione che non sia istantaneo e pari a ω_{max} ad ogni passo di controllo.

5.3 Simulatore DUC

L'ambiente di simulazione sviluppato prende il nome di **DUC** (*Distributed UAVs Coordinator*). In esso sono presenti metodi per aggiornare tutti parametri a scelta dell'utente, descritti finora in relazione alle quattro classi principali: `Coordinates`, `Sensor`, `Node` e `Map`.

- `Coordinates:` modella la posizione di un punto sulla superficie terrestre con tutte le funzioni di utilità descritte nelle sezioni precedenti;
- `Sensor:` descrive il singolo sensore in termini di *posizione angolare*, *angolo di apertura* e *dimensione del fascio*, così come specificato nel capitolo 4;
- `Node:` definisce il comportamento dei singoli velivoli autonomi durante il movimento, compresi i metodi per lo scambio di messaggi nel grafo e per rispondere agli stimoli dei sensori di prossimità;
- `Map:` è un semplice *wrapper* per le API di supporto, che si interpone tra le classi precedenti e si occupa di aggiornare graficamente la mappa.

Uno *screenshot* panoramico del simulatore DUC è presentato in figura 5.7, da cui è possibile apprezzare il layout completo al momento dell'apertura da browser (nel particolare con *Google Chrome*).

L'intera interfaccia è studiata per lasciare spazio alla visualizzazione del comportamento dei singoli nodi durante il movimento. I tre blocchi principali

corrispondono alle esigenze di regolare i parametri relativi alla mappa (come *zoom*, *inclinazione*, *fullscreen*, ...), al sistema (*numero di nodi*, *velocità massima*, *raggio di copertura*, ...) e ai sensori su ciascun quadricottero (*posizione*, *numero totale*, *distanza massima rilevabile*, ...).

Prima di avviare una simulazione, è possibile scegliere la particolare configurazione da assegnare del sistema multi-robot in termini di:

- **Numero di nodi** totale che compongono il team, considerando la somma di tutti i master e dei follower[5];
- **Numero di destinazioni** che devono essere raggiunte dalla squadra e che determina anche il numero di master e, di conseguenza, la definizione dei follower;
- **Velocità** massima ad ogni passo di controllo, ovvero v_{max}, espressa in *metri al secondo*;
- **Velocità angolare** massima ad ogni passo di controllo, ovvero ω_{max}, espressa in *radianti al secondo*;
- Dimensione della **finestra di controllo** che scandisce gli istanti discreti, ovvero $T = t_{i+1} - t_i$, espressa in *millisecondi*;
- Periodo di **aggiornamento dell'algoritmo**, ovvero $T_a = k_a \cdot T$, che definisce ogni quanti *millisecondi* inviare i parametri SOM da aggiornare, dai master verso i follower;
- Intervallo di **aggiornamento della mappa**, in modo da permettere un'esecuzione più scorrevole anche in presenza di risorse limitate (browser non aggiornato o con supporto ridotto, motore di rendering lento, ...);

Va fatto presente che gli ultimi tre parametri temporali appena descritti sono modificabili solo attraverso il relativo *slider*, in quanto l'apposito campo di testo sulla destra rappresenta solo un'etichetta del valore assunto.

Per quanto concerne invece la scelta dei sensori di prossimità su ciascun UAV, si è ritenuto opportuno mantenere la stessa configurazione per tutti i

[5]Poiché non si hanno particolari esigenze sulla scelta dei master, si considerano i nodi completamente identici, sfruttando le considerazioni fatte nella sezione 3.2.

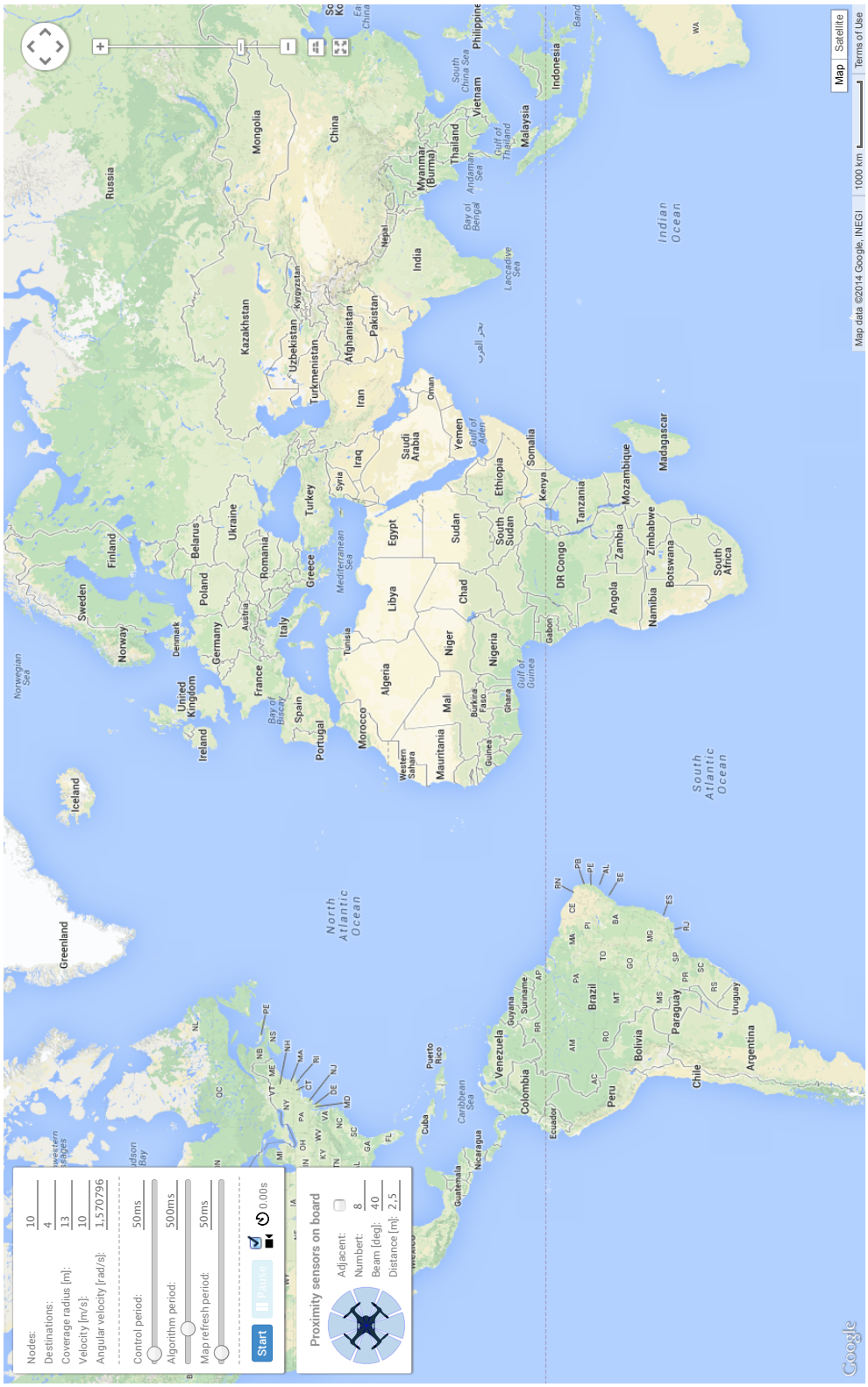


Figura 5.7: Schermata iniziale del simulatore *Distributed UAVs Coordinator*.

quadricotteri. In questo modo, l'utente è in grado di definire la conformazione di tutto l'equipaggiamento aggiuntivo necessario per il rilevamento degli ostacoli e degli altri droni.

Nello specifico, è possibile impostare i seguenti parametri:

- Il **numero di sensori** ad ultrasuoni presenti su ogni unità;
- L'**angolo di apertura** del cono prodotto da ogni fascio ultrasonico;
- La **distanza** massima rilevabile da ogni sensore[6];
- La **posizione** in funzione dell'impostazione di *adiacenza*. Non viene fornita l'opportunità di impostare la posizione angolare in gradi di ogni sensore, ma viene offerta la possibilità di scegliere se disporre tale dotazione in uno dei due modi che seguono:
  - **Non adiacenti**: in base al numero di sensori si calcola una disposizione tale da avere la stessa distanza tra il sensore i-esimo e il successivo, facendo particolarmente attenzione ad evitare le sovrapposizioni tra i fasci;
  - **Adiacenti:** in funzione dell'angolo di apertura si dispongono i sensori in modo che l'estremo laterale sinistro del fascio di uno corrisponda con quello destro del sensore successivo.

  In entrambi i casi, il simulatore suggerisce una collocazione tale da garantire una copertura sensoriale nella direzione dell'angolo di orientamento θ_i del drone i-esimo.

Le costanti che invece non sono disponibili all'utente in questa prima versione di DUC sono schematizzate in tabella 5.3.

Per quanto concerne i comandi per modificare l'area geografica visibile sullo schermo, per *zoommare* sulla mappa, o per le altre opzioni rese disponibili dalle API, si rimanda alla guida ufficiale di Google Maps [**18**]. Uno strumento che è stato invece sviluppato personalmente è l'opzione **follow-me**: selezionando la *checkbox* sopra la piccola camera in nero vicino al pulsante di `Stop` dal menu dei

[6]La distanza minima non è stata parametrizzata; tuttavia si suppongono nulli gli effetti relativi ad eventuali impedimenti dovuti alle caratteristiche fisiche/meccaniche del robot.

Descrizione	Simbolo	Valore
Coeff. apprendimento iniziale	α_0	1
Raggio di vicinato iniziale	σ_0	n [7]
`RefuseLimit`	–	3 [8]
Polo di movimento	ρ_m	0.8
Polo di rotazione	ρ_r	0.8
Zona geografica	–	PSA[9]

Tabella 5.3: Parametri costanti nel simulatore DUC non manipolabili direttamente dall'utente.

comandi, si ottiene una visuale della mappa durante la simulazione in cui sono sempre visibili sia i nodi che le destinazioni, adeguando dinamicamente il livello di zoom e il centro del riquadro di visualizzazione.

5.3.1 Formazione iniziale

Quando viene premuto il pulsante `Start` nel menu dei comandi, si dà inizio alla configurazione iniziale del sistema. DUC procede con la fase di *start-up* secondo le regole definite dall'algoritmo di coordinamento nel capitolo 3.

Ad ogni primo avvio, oppure ad ogni nuova simulazione se il numero dei nodi non è stato modificato rispetto alla prova precedente, si procede al posizionamento degli agenti nelle immediate vicinanze del punto di partenza (aeroporto di Pisa). In seguito, si calcolano le destinazioni da assegnare ai master che verranno eletti, facendo attenzione che l'insieme ottenuto ammetta una soluzione possibile. In particolare, si generano i punti seguendo la condizione di fattibilità preliminare descritta in sezione 3.2.

A questo punto si generano gli ostacoli: si produce un numero random di ostacoli compreso tra 2 e 10 di forma circolare (o cilindrica con altezza infinita) alla stessa latitudine. Ognuno di essi è distante 10 metri dal più vicino e i loro baricentro coincide con il baricentro dei punti obiettivo. Questa semplice

[7] Numero di nodi del grafo.

[8] Vedi sezione 3.6.

[9] Le coordinate del punto iniziale di riferimento sono quelle dell'aeroporto di Pisa-San Giusto, le cui coordinate sono 43.696° N e 10.398 139 5° W.

configurazione ha permesso di verificare la strategia di collision avoidance, poiché fissa una forte probabilità da parte del team di incontrare almeno un ostacolo durante il movimento. Come già specificato, inoltre, la presenza di impedimenti ambientali non assicura la possibilità di copertura di tutti i punti da parte della MANET. Per di più, poiché le destinazioni sono generate in maniera casuale, può capitare che un set-point sia troppo vicino ad un ostacolo oppure che i due si sovrappongano. In tal caso non vi è alcun controllo automatico da parte del sistema, sta all'utente accorgersi dell'impossibilità di portare a termine il task per il particolare caso d'uso.

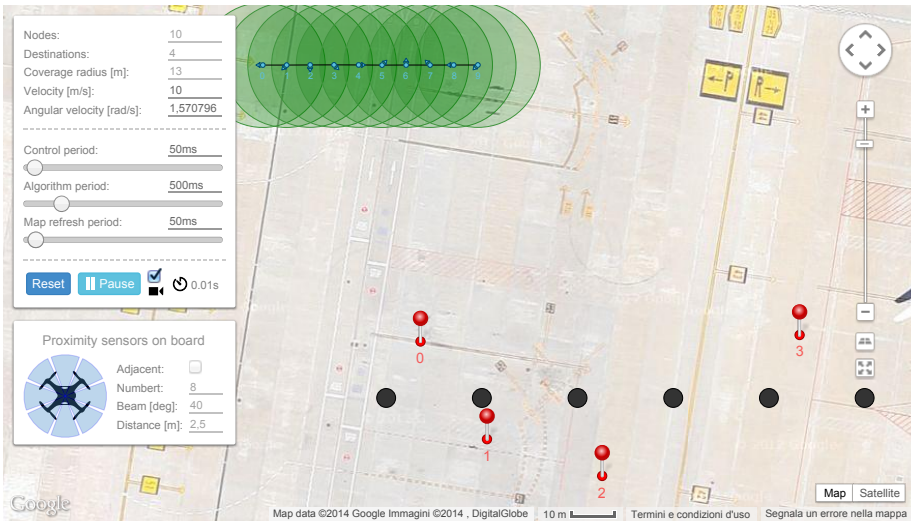


Figura 5.8: Fase di avvio del simulatore DUC in seguito alla generazione del *training-set* casuale in cui sono presenti 6 ostacoli (in nero) alla stessa latitudine.

L'inizializzazione termina con l'elezione dei master secondo il metodo descritto nell'algoritmo di coordinamento e con la disabilitazione di alcuni campi precedente attivi, in particolare:

- Numero di nodi;
- Numero di destinazioni;
- Raggio di comunicazione;
- Tutti i parametri dei sensori (ovvero numero, posizione e dimensioni del fascio).

5.3.2 Coordinamento distribuito

Una volta creata la MANET e definite le coordinate delle destinazioni, si avvia il timer di controllo che funge da semplice *cronometro*. Al tempo stesso, oltre agli input disabilitati, si aggiornano anche i pulsanti per l'avvio/reset/pausa della rete, nello specifico:

- `Start` diventa `Reset`: permette di avviare una nuova simulazione con i parametri in uso e, poiché non è possibile aggiornare il numero dei nodi, riparte dall'ultima posizione in cui si trovavano i quadricotteri stessi;
- `Pause` si attiva: premendo questo pulsante è possibile congelare completamente lo stato attuale della simulazione, mantenendo immutati sia i parametri generali che il timer di di controllo. Ovviamente, l'etichetta del pulsante cambia in `Play` quando DUC viene messo in pausa.

Adesso il controllo passa alla classe `Map`, che dopo aver eliminato eventuali residui delle prove precedenti (in termini di nodi, destinazioni e ostacoli), determina i bordi e le coordinate del punto centrale del frame di visualizzazione che racchiude la rete e i target-point sotto il massimo livello di zoom[10]. Centrata la mappa, si occupa di svegliare tutti gli agenti del team e di chiamare il metodo `Map.draw()` ad ogni intervallo definito dal *periodo di aggiornamento* o ogni volta che viene effettuata una zoommata sullo schermo[11].

Una volta avviati i nodi attraverso il metodo `Node.start()`, ciascuno di essi porta avanti periodicamente ed in maniera indipendente la propria computazione ad intervalli scanditi dal periodo di controllo T.

È a questo punto che ha inizio il coordinamento vero e proprio dei droni: in base alle regole descritte nel capitolo 3, ogni agente ha un ruolo ben definito (master/follower) e procede con gli altri a completare l'obiettivo globale per cui la

[10] In Google Maps la possibilità di ingrandire o di ridurre la mappa non è lineare, ma definita secondo 21 livelli crescenti di zoom.

[11] Quando si ingrandisce o si riduce la dimensione della mappa non vengono ridimensionate allo stesso modo anche le dimensioni degli elementi aggiunti manualmente (cerchi di copertura, identificativi di nodi e destinazioni, ostacoli ...). Per questo motivo la dimensione di tali elementi è definita in maniera dinamica in funzione della scala della mappa secondo la formula $scale = 2^{zoom} \cdot \frac{128}{\pi R \cdot \cos(\phi)}$, dove ϕ è la latitudine nel punto considerato..

stessa rete è stata costituita. Durante gli spostamenti, ogni robot può comunicare solo con il precedente e con il successivo. Per simulare un networking più simile al reale, o quantomeno non ideale, ogni messaggio viene serializzato ed incapsulato in un pacchetto JSON, per poi essere inviato al destinatario dopo un tempo compreso tra 1 e 10ms scelto in maniera randomica.

Ogni volta che un master giunge alla destinazione assegnatagli, invia il messaggio *arrived* (vedi sezione 3.6) agli agenti del grafo secondo le modalità precedentemente descritte in sezione 3.4. Quando tutti i master sono riusciti a spingersi fino al punto in cui dovranno portare a termine il *task* della rete, anche i follower sono a conoscenza del compimento dell'obiettivo, e tutti i robot in generale si fermano nell'ultima posizione in cui si sono portati.

La gestione passa nuovamente alla classe `Map`, che si accorge della situazione al passo di controllo successivo e provvede a richiamare la particolare *callback* definita dall'utente. In questo caso, si tratta solo di effettuare rivedere l'interfaccia dei comandi in termini di:

- Arresto del cronometro e adeguamento del tempo totale trascorso;
- Disattivazione del pulsante di `Pause`;
- Aggiornamento del pulsante di `Reset` in `Start`;
- Riattivazione dei campi di input che erano stati resi indisponibili durante tutto il processo di simulazione.

5.4 Considerazioni

È stato effettuato un numero elevato ed indefinito di test attraverso il simulatore *Distributed UAVs Coordinator* che ha portato ad una serie di valutazioni riassunte in questa sezione.

Innanzitutto, si può notare che il fatto che alcuni campi di controllo non vengano disabilitati durante la prova di simulazione può determinare un aumento o una riduzione del tempo necessario per coprire il set di ingresso dato. Difatti, durante il movimento, è possibile ritoccare dinamicamente alcuni parametri

imprescindibili per le prestazioni del sistema, tra cui il modulo della velocità di spostamento in direzione dei target e la velocità angolare di rotazione. L'intervallo di aggiornamento dell'algoritmo, invece, non rappresenta un fattore fondamentale, poiché descrive solo il periodo entro il quale i master inviano il messaggio *command* ai follower.

Un altro aspetto rilevate è dato dall'uso di variabili di stato espresse nel *sistema MKS* (Metro Kilogrammo Secondo, da cui deriva l'attuale standard del Sistema Internazionale). Il vantaggio rispetto ad altri simulatori che sfruttano una zona di lavoro definita da un riquadro di altezza e larghezza in *pixel* risulta evidente: poiché tutte le dimensioni, comprese le distanze e le estensioni di lunghezza dei nodi e degli ostacoli, sono espresse in metri, si ottiene un impatto visivo immediato della dimensione del problema, anche grazie al confronto con la mappa geografica satellitare presente sullo sfondo. Cambiando difatti parametri fisici della MANET, quali dimensione degli UAV o del raggio di comunicazione, e spostando la zona di lavoro in ambiente abitato, si può portare a termine una simulazione che si avvicina sempre più al comportamento reale, riuscendo anche ad avere un impatto grafico della soluzione da attuare (ad esempio si può capire quale può essere l'estensione superficiale della rete in rapporto all'area urbana considerata).

In ultima analisi, anche se non meno importante, è stata notata la possibilità di stravolgere parte dell'algoritmo di controllo eliminando il messaggio di *command*. Questa osservazione nasce da una serie di prove effettuate in cui è stato modificato l'insieme Γ_i dei target temporanei dei follower ad ogni passo di controllo, invertendo l'ordine proposto in equazione 3.2. Aumentando la priorità del punto medio p_i^m, calcolato sulla base della posizione dei due vicini adiacenti, oltre ad eliminare (anche se non è propriamente vero) la dipendenza dai comandi del master, si ottiene una formazione topologica durante il movimento molto più fluida. Così facendo, i droni si muovono seguendo la posizione "ottimale" nel grafo, in cui si trovano sempre alla stessa distanza sia dal nodo precedente che da quello successivo. Tale aspetto è da considerarsi molto importante, in quanto apre allo studio di una nuova strategia di coordinamento che non fa uso dei parametri neurali derivati dalle SOM, ma sfrutta solo la posizione dei robot e la comunicazione fra vicini nella rete.

Capitolo 6

Conclusioni

Grazie al lavoro di ricerca svolto durante questa tesi, è stato possibile mettere a fuoco le problematiche inerenti al coordinamento di un sistema multi-robot autonomo. È stato inoltre possibile comprendere come questa branca relativamente recente della robotica copra diversi aspetti legati alla progettazione, dallo schema architetturale della rete, allo scambio dei messaggi fra i nodi della MANET, fino al modello cinematico discreto e, ovviamente, le strategie di coordinamento.

Basandosi sull'interazione locale, ovvero in assenza di un centro di calcolo centrale per la *supervisione* degli agenti, è stata studiata e proposta una strategia di organizzazione distribuita concretizzabile grazie allo scambio di messaggi in presenza di collegamenti senza fili. Poiché il networking rappresenta probabilmente l'aspetto più importante nei sistemi distribuiti, l'obiettivo principale è rappresentato dal mantenimento della connettività in ogni istante di movimento, cosicché sia possibile per taluni robot raggiungere dei punti prestabiliti per svolgere i task che dipendono dalla particolare applicazione.

Partendo dall'architettura *Leader/Follower* descritta in [**11**] e dalla teoria sulle mappe auto-organizzanti, è stato formalizzato uno schema di movimento per ciascuna tipologia di robot e una strategia generale per la rilevazione degli ostacoli e il conseguente superamento degli stessi, attraverso una dotazione sensoristica supplementare. Rispetto allo *stato dell'arte* analizzato, si è cercato di ridistribuire le fasi del processo di apprendimento concentrandole sui singoli nodi, abbando-

nando, di fatto, il concetto di competizione tipico delle SOM, ed analizzando il caso generale in cui sono presenti più di due leader.

6.1 Sviluppi futuri

Il lavoro svolto in questa tesi è finalizzato all'implementazione di un sistema distribuito di agenti reali, nel particolar caso quadricotteri. Esistono, in realtà, una serie di accorgimenti e di potenziamenti successivi che vale la pena analizzare.

Innanzitutto, il software di simulazione DUC, nonostante sia perfettamente funzionante, è stato definito in un linguaggio di programmazione non propriamente consono ai sistemi embedded. Difatti, per permettere l'utilizzo delle sue funzionalità su droni, in modo da sfruttare appieno la potenza di calcolo disponibile, è necessaria una conversione preliminare in un linguaggio più vicino a quello con cui è stato programmato dell'elaboratore (generalmente C). Anche la realizzazione della temporizzazione dei vari processi che vengono eseguiti in maniera periodica è fortemente dipendente dall'architettura del sistema autonomo e dal sistema operativo utilizzato.

Rimanendo invece in ambiente simulato, il primo passo è sicuramente quello di modificare la strategia proposta eliminando la componente "*neurale*" introdotta. Come già specificato in sezione 5.4, si può provare ad eliminare completamente la *destinazione indotta* dai master nei follower ad ogni intervallo di aggiornamento T_a dell'algoritmo, puntando unicamente sul calcolo del punto medio tra i nodi vicini (rimane comunque immutato il concetto di insieme dei target Γ_i a fronte di un unico punto di arrivo). Così facendo, però, è necessario rimuovere anche il concetto di *vicinato diretto* definito dal parametro σ_i e dalla funzione $\Phi(d)$.

In merito alle migliorie da apportare al simulatore, oltre all'opzione di scelta della zona di lavoro iniziale in cui svolgere le simulazioni, tutt'ora non presente, vi è la possibilità di sviluppare un applicativo 3D che faccia uso anche dell'altitudine dei velivoli. Difatti, nonostante sia stata prevista una variabile apposita in DUC, essa non viene mai presa in considerazione neanche per il calcolo della distanza fra i nodi della rete. Il passo successivo consiste nel sostituire il tool di

supporto di Google Maps con uno che tenga conto anche degli ostacoli geografici "realmente" presenti nell'ambiente (come *NASA World Wind*, una progetto open-source sviluppato in .NET e JAVA, o *Google Earth*, dal quale è nato recentemente un plugin web per la visualizzazione da browser), e nel modificare le formule per la trasformazione di coordinate in modo da dar peso anche dell'altezza dell'oggetto dal livello del mare.

Per testare, invece, la robustezza dell'algoritmo di coordinamento anche in presenza di disturbi nell'ambiente sarebbe opportuno sfruttare dei simulatore di rete appositamente sviluppati, come *ns* (Network Simulator, attualmente alla versione 3 e difatti noto come ns-3). Allo stesso modo, conviene far uso anche di simulatori di volo esistenti (come *X-Plane*, sotto licenza, o *Flight Gear*, open-source) per ottenere una cinematica del corpo rigido il più possibile fedele alla realtà, invece di analizzare le leggi della dinamica del moto in cui si tengono in considerazione gli eventuali vincoli di velocità e accelerazione.

Un lavoro più impegnativo può essere la rivisitazione della strategia attraverso metodi di *consenso distribuito* che non sono stati analizzati. Sarebbe opportuno, quantomeno, che i nodi comunicassero fra loro durante la fase iniziale in modo che la formazione della rete o la scelta dei master venga definita attraverso un processo di accettazione univoco.

Elenco delle figure

Capitolo 1

Capitolo 2

Capitolo 3

Capitolo 4

Capitolo 5

Elenco delle tabelle

Bibliografia

[1] Wikipedia. *Agente Intelligente*. URL: http://it.wikipedia.org/wiki/Agente_intelligente.

[2] Michael Wooldridge. *An Introduction to Multi-Aagent Systems*. New York: John Wiley & Sons, mag. 2009.

[3] Treccani. *Definizione di Autogiro*. URL: http://www.treccani.it/enciclopedia/autogiro/.

[4] Boeing (ex. McDonnell Douglas). *AV-8B Harrier II V/STOL Aircraft*. URL: http://www.boeing.com/boeing/history/mdc/harrier.page.

[5] Andy Pasztor e John Emshwiller. "Drone Use Takes Off on the Home Front". In: *The Wall Street Journal* (apr. 2012). URL: http://online.wsj.com/article/SB10001424052702304331204577354331959335276.html.

[6] Honeywell. *T-Hawk MAV*. URL: https://commerce.honeywell.com/webapp/wcs/stores/servlet/eSystemDisplay?catalogId=10251&storeId=10651&categoryId=37901.

[7] Northrop Grumman. *RQ-4 Block 10 Global Hawk*. URL: http://www.northropgrumman.com/Capabilities/RQ4Block10GlobalHawk/Pages/default.aspx.

[8] Maurizio Molinari. "Droni Usa per osservare i reattori". In: *La Stampa* (mar. 2011). URL: http://www.lastampa.it/2011/03/17/blogs/finestra-sull-america/droni-usa-per-osservare-i-reattori-bIl3KsGjUsShfHscnJkUlN/pagina.html.

[9] Hangar 18. *O2*. URL: http://hangar18uav.com/osprey.htm.

[10] "Mobile Ad-hoc Networks (MANET)". In: *Proceedings of the Thirty-ninth Internet Engineering Task Force*. The Internet Engineering Task Force.

Munich, Germany, ago. 1997. URL: https://www.ietf.org/proceedings/39/transit97aug-86.htm.

[11] Tullio Facchinetti, Gianluca Franchino e Giorgio Buttazzo. "A Distributed Coordination Protocol for the Connectivity Maintenance in a Network of Mobile Units". In: *The First International Conference on Advances in Mesh Networks*. Cap Esterel, France, ago. 2008, pp. 764–769. URL: http://retis.sssup.it/~giorgio/paps/2008/mesh08.pdf.

[12] Jonathan Gross e Jay Yellen. *Graph Theory and its Applications*. CRC press, 2005.

[13] Teuvo Kohonen. "Self-Organized Formation of Topologically Correct Feature Maps". In: *Biological Cybernetics* 43 (1982), pp. 59–69.

[14] John Adrian Bondy e Uppaluri Siva Ramachandra Murty. *Graph Theory With Applications*. Vol. 6. Macmillan London, 1976, pp. 12–21. ISBN: 0-444-19451-7.

[15] World Wide Web Consortium (W3C). *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*. Ago. 2011. URL: http://www.w3.org/TR/SVG/.

[16] Google Inc. *Google Maps JavaScript API v3*. URL: https://developers.google.com/maps/documentation/javascript/.

[17] Thaddeus Vincenty. "Direct and Inverse Solutions of Geodesics on the Ellipsoid with application of nested equations". In: *Survey review* 23.176 (1975), pp. 88–93.

[18] Google Inc. *Google Maps Help*. URL: https://support.google.com/maps/.