Coordinazione di veicoli multipli per problemi di Rendez-Vous, Flocking e Deployment

Massimo Bortolato, Matteo Bustreo, Nicola Natali

Abstract— **In questo articolo si presentano alcuni algoritmi di controllo e coordinazione per gruppi di robot mobili autonomi con visibiltá limitata.**

Si affronta aprofonditamente il problema del rendez-vous, per il quale vengono proposti alcuni algoritmi distribuiti e senza memoria. Una dimostrazione formale della loro correttezza viene derivata utilizzando un modello astratto dei veicoli, in cui ciascun agente é un punto materiale mobile con dinamica descrivibile per mezzo di un integratore lineare a tempo continuo. Successivamente si utilizza un modello piú realistico per descrivere i veicoli, tenendo conto delle asincronicitá, degli errori di misura dei sensori, e dei vincoli anolonomi. I risultati ottenuti inducono a ritenere che l'applicazione di questi algoritmi su veicoli reali dia risultati analoghi a quelli ottenuti in simulazione.

Vengono in seguito descritti i problemi del deployment e del flocking, analizzandone le problematiche principali e illustrando alcuni possibili approcci risolutivi.

I. Introduzione

Le recenti evoluzioni tecnologiche nel campo dei sistemi di controllo per veicoli, le enormi capacità di calcolo e di comunicazione dell'elettronica attuale, nonché l'avvento della tecnologia della miniaturizzazione dei sistemi elettromeccanici hanno elevato l'interesse per veicoli che possono interagire in maniera autonoma con altri veicoli e con l'ambiente che li circonda per svolgere, anche in presenza di incertezza ed avversità, compiti oltre la capacità dei singoli veicoli.

Le aree di applicazione includono la coordinazione di veicoli aerei e sottomarini autonomi, sistemi automatici per la guida in autostrada, carrelli elevatori autonomi e robot mobili, operazioni di ricerca e soccorso, operazioni in ambienti pericolosi, esplorazione, ricognizione, sorveglianza, monitoraggio ambientale per la stima e la prevenzione di inquinamento e incendi, applicazioni militari.

Nonostante ciascuna di queste aree ponga le proprie specifiche difficoltà, si possono tuttavia ritrovare parecchi aspetti in comune. Nella maggior parte dei casi, i veicoli interagiscono tra loro per svolgere il compito loro affidato, ma sono altresì disaccoppiati l'un l'altro, nel senso che il comportamento di un veicolo non influisce direttamente sul comportamento degli altri. Ogni veicolo utilizza lo stesso algoritmo condiviso da tutti, che permette di prendere una decisione sulla base della sola informazione disponibile localmente, informazione che può essere soggetta ad incertezze e ritardi di trasmissione. Sistemi che operano sotto queste condizioni sono riferiti come *Sistemi Distribuiti*.

In questo articolo si introducono e si studiano tre dei principali problemi che interessano la coordinazione di veicoli autonomi: il *rendez-vous*, il *flocking* e il *deployment*. Il problema del rendez-vous consiste nel ricercare degli algoritmi per far convergere un certo numero di robot, inizialmente sparsi su di un piano, all'interno di una piccola area. Questo problema può essere visto come un caso particolare del problema più generale del riposizionamento dei veicoli in predeterminate formazioni geometriche o distribuzioni. I problemi di formazione nascono spontaneamente nella cooperazione multi-robot (per esempio, riposizionarsi prima di iniziare la cooperazione, oppure riposizionarsi per caricare le batterie), ma sono anche importanti per la loro stretta relazione con il problema del consenso, nel quale i robot devono raggiungere un accordo su questioni che li riguardano. Per esempio, se i robot possono muoversi mantenendo una disposizione in linea retta, significa che possono risolvere un certo numero di problemi di consenso, come eleggere un leader (cioé, il primo robot delle retta), ordinarsi (cioé, scegliere l'ordine in cui comparire nella retta), ed assegnarsi differenti mansioni.

Il problema del flocking è anch'esso un problema di coordinazione del movimento di un grande numero di robot interagenti con un comune obiettivo di gruppo. In natura è frequente vedere gruppi di animali che si muovono in modo coordinato formando "sciami ordinati". Questo capita ad esempio per uccelli, pesci, api, greggi. Ogni animale decide autonomamente il proprio comportamento in base alla percezione locale della dinamica del gruppo. Il movimento d'insieme che si percepisce è quindi il risultato di una densa interazione tra i comportamenti relativamente semplici di ogni individuo. Nel flocking si cerca di individuare le regole fondamentali che ogni agente deve rispettare in modo da ottenere un comportamento simile per tutti i veicoli di un dato

insieme.

Infine, il problema del deployment, rappresenta una delle più interessanti applicazioni della coordinazione di più veicoli: l'obiettivo è quello di ottimizzare la comunicazione ed il movimento dei robot in modo da esplorare un dato spazio. Sotto certi punti di vista questo problema rappresenta l'inverso del problema del rendezvous, infatti è evidente come in questo caso la prima cosa da fare sia cercare di massimizzare la distanza tra i veicoli, garantendo la conservazione della connessione tra gli stessi.

Questo modo di utilizzare un insieme di veicoli ha assunto particolare importanza nella ricerca militare perché offre la possibilità di bonificare territori ostili o generare mappe di zone sconosciute (specialmente territri urbani, sotterranei o particolarmente complessi dal punto di vista topologico), e nell'esplorazione spaziale (ad esempio per la ricerca di forme di vita su Marte), esempio estremo di ambiente ostile.

Per la natura pratica del problema del deployment, molto legata allo scopo specifico che di volta in volta si cerca di raggiungere, risulta difficilmente trattabile in modo formale e unificato, quindi si é scelto di presentare le problematiche in modo descrittivo, cercando di porre l'accento sulle molte sfaccettature e complicanze che insorgono quando si cerca di applicare in un ambiente reale gli algoritmi proposti.

I sistemi che verranno studiati coinvolgono spesso, oltre ad un numero enorme di interazioni, anche numerose non linearità, come quantizzazioni e modellizzazioni di disturbo, quindi una trattazione rigorosamente formale e matematica del problema risulta molto complessa e non sempre possibile. Per questo motivo si farà massiccio uso della simulazione MATLAB dei processi in gioco, mentre rigorose formalizzazioni matematiche saranno esposte solamente nei casi di estrema semplificazione e idealizzazione.

II. Preliminari

In questa sezione vengono riportate alcune definizioni preliminari che saranno utilizzate nel resto del documento.

A. Rete di veicoli e Grafi

Un grafo orientato è una coppia $G = (V, A)$ in cui V è un insieme finito, ed A è una famiglia di coppie ordinate di elementi di V. Gli elementi di V sono chiamati vertici di G, mentre gli elementi di A sono chiamati archi. Gli archi sono quindi coppie del tipo (i, j) con $i, j \in V$. L'ordine in cui appaiono i vertici è rilevante, ed in genere $(i, j) \neq (j, i)$.

La rete di comunicazione formata dai veicoli può venire descritta per mezzo di grafi in cui ogni vertice v_i rappresenta l'i-esimo robot e in cui ogni arco (i, j) rappresenta la possibilità, per il veicolo i-esimo, di comunicare con il veicolo j-esimo. In seguito verrà indicato con $N = |V|$ il numero di vertici di G.

Due vertici si dicono adiacenti se esiste un arco che li collega, ed il numero di vertici adiacenti ad un vertice v_i si dice grado di v_i e si indica con $d_G(v_i)$.

Un cammino (orientato) di G é una sequenza di archi successivi $a_1, a_2, ...a_k \in A$ del tipo $a_1 = (v_1, v_2), a_2 = (v_2, v_3), ..., a_k = (v_k, v_{k+1})$. La presenza di un tale cammino fa sì che v_{k+1} possa dirsi raggiungibile a partire da v_1. Un grafo si dice *fortemente connesso* se esiste un cammino da w a v per ogni coppia $w, v \in V$ (e quindi anche da v ad w). In generale non si ha a che fare con grafi simmetrici, ovvero se $(i, j) \in A$ non é necessariamente vero che $(j, i) \in A$, quindi il concetto di raggiungibilità non è simmetrico. Nel caso in cui ciò avvenga il grafo viene anche detto non orientato.

Un grafo si dice bilanciato se per ogni nodo la somma degli archi entranti è pari alla somma degli archi uscenti. Ne segue che un grafo non orientato é un grafo bilanciato.

I grafi possono essere descritti per mezzo di opportune matrici in modo da formalizzare matematicamente i problemi su di essi definiti.

La *matrice di adiacenza* A di un grafo orientato G è la matrice $N \times N$ con elementi

$$a_{ij} = \begin{cases} 1 & \text{se } (i, j) \in A \\ 0 & \text{se } (i, j) \notin A \end{cases}$$

La *matrice laplaciana* L di un grafo orientato G è la matrice $N \times N$ definita da $L = D - A$, dove D è la matrice diagonale di ordine N che ha sulla diagonale principale il grado di ciascun vertice $D = \text{diag}(d_G(v_1), d_G(v_2), ..., d_G(v_N))$. Nel caso di grafi non orientati le matrici di adiacenza A e laplaciana L risultano essere simmeriche.

Per costruzione la somma degli elementi di una riga della matrice laplaciana L di un grafo è pari a zero. Indicando con $\mathbf{1}$ il vettore $\mathbf{1} = (1, \ldots, 1)^T$ si può dunque scrivere $L\mathbf{1} = 0$, vale a dire $\mathbf{1}$ è autovettore destro di L relativo all'autovalore zero. Per il teorema di Gershgorin tutti gli autovalori di L sono interni al disco chiuso nel piano complesso centrato in $\Delta + 0j$ e con raggio pari a Δ, dove $\Delta = \max_i(d_G(v_i))$, cioè pari al massimo grado del grafo. Di conseguenza tutti gli autovalori della matrice laplaciana hanno parte reale non negativa. Per grafi non orientati, la laplaciana, in quanto simmetrica,

ha autovalori reali che possono essere così ordinati

$$0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N \leq 2\Delta \tag{1}$$

Se il grafo G è fortemente connesso l'autovalore zero è isolato, $\lambda_2 > 0$. Il secondo autovalore λ_2 di L, chiamato *connettività algebrica* gioca un ruolo importante nella misura delle prestazioni degli algoritmi di coordinazione e consenso. Il suo significato, così come l'importanza dell'analisi spetrale della matrice laplaciana di un grafo, sarà chiarito in sezione III.

Un'altra matrice che ci sarà utile in seguito è la matrice di *Perron* di parametro ϵ, $P(\epsilon)$, associata al grafo G, definita come

$$P = I - \epsilon L \tag{2}$$

dove L è la laplaciana e ϵ è uno scalare $\epsilon \in (0, 1/\Delta]$. La matrice di Perron ha quindi tutti gli elementi non negativi.

Una matrice non negativa viene chiamata *stocastica* se la somma degli elementi delle sue righe vale uno, *doppiamente stocastica* se è stocastica e la somma degli elementi delle sue colonne è pari a uno. Una matrice stocastica è *primitiva* (o *ergodica*) se ha un solo autovalore di modulo massimo.

La matrice di Perron P di parametro $\epsilon \in (0, 1/\Delta]$ di un grafo orientato G con massimo grado Δ gode delle seguenti proprietà:

1) P è stocastica con autovalore uno associato all'autovettore $\mathbf{1}$;
2) tutti gli autovalori di P sono interni al cerchio unitario;
3) se G è bilanciato, allora P è doppiamente stocastica;
4) se G è fortemente connesso e $0 < \epsilon < 1/\Delta$, allora P è primitiva.

Un'importante proprietà delle matrici primitive ci è data dal seguente teorema:

Teorema 1 (Perron-Frobenius): Sia P una matrice primitiva con autovettori sinistro e destro w e v, rispettivamente, tali che

$$Pv = v, \quad w^T P = w^T, \quad v^T w = 1$$

Allora $\lim_{k\to\infty} P^k = vw^T$
Questo risultato ci sarà utile in sezione III-A per dimostrare la convergenza di un algoritmo per il rendez-vous.

III. Algoritmi per il Rendez-Vous

Il problema del rendez-vous consiste nel raggruppare all'interno di una piccola area un certo numero di veicoli inizialmente sparsi su un piano bidimensionale.

Intuitivamente il problema potrebbe essere risolto in maniera banale chiedendo a tutti gli agenti di dirigersi verso l'origine del piano, o verso un altro punto prefissato. Questo approccio ha però due grandi limitazioni. Innanzitutto risulta essere poco economico, in quanto se inizialmente i robot sono tutti concentrati in un'unica area distante dal punto di ritrovo, verrebbe loro richiesto di percorrere inutilmente molta strada per giungerci. In secondo luogo presuppone che ciascun agente sia in grado di conoscere la propria posizione assoluta (x, y), cosa che in molti casi reali non è vera dal momento che spesso i veicoli non sono equipaggiati di strumenti GPS, e possono solo calcolare la posizione relativa degli oggetti che li circondano.

Verranno ora illustrati alcuni algoritmi che risolvono il problema del rendez-vous per una rete di robot mobili. Il problema verrà affrontato partendo da condizioni fortemente semplificate e generalizzate: in sezione III-A i veicoli sono modellati come punti materiali inseriti in una rete tempo-invariante e in grado di muoversi istantameamente lungo le direzioni x e y. In sezione III-B si suppone che il raggio di comunicazione di ciascun robot sia limitato, e che quindi possa rilevare e comunicare solo con quei veicoli che si trovano entro una distanza $D > 0$. La rete di sensori diviene in questo caso tempo-variante. In sezione III-D verrano infine introdotti vincoli anolonomi nella descrizione del moto dei robot.

A. Punti materiali in una rete di sensori tempo invariante

Si considerano i veicoli come punti materiali che si muovono nel piano (x, y) secondo relazioni lineari e tempo invarianti. Si ipotizza che possano ricevere ed elaborare le informazioni di cui necessitano istantaneamente, quindi l'evoluzione del sistema può venire descritta per mezzo di un modello a tempo continuo. Si ipotizza, inoltre, che la rete di comunicazione tra i veicoli sia simmetrica e tempo-invariante, vale a dire se esiste l'arco (i, j), allora esiste anche l'arco (j, i), e se l'arco (i, j) esiste al tempo t_0, allora esso esiste anche $\forall t > t_0$.

La posizione q_i di un generico veicolo v_i sarà descritta per mezzo di variabili complesse:

$$q_i(t) \quad = \quad x_i(t) + jy_i(t) \tag{3}$$

Sotto queste condizioni è possibile utilizzare una semplice strategia di controllo distribuito:

$$\dot{q}_i(t) = \sum_{j\in N_i} a_{ij}(q_j(t) - q_i(t)) \tag{4}$$

In appendice I si dimostra che il protocollo (4) garantisce la convergenza asintotica nel baricentro a partire da qualsiasi condizione iniziale $q(0)$.

Si nota che la (4) può essere riscritta come

$$\dot{q}(t) = -Lq(t) \tag{5}$$

dove L è la matrice laplaciana del grafo che rappresenta la rete costituita dall'insieme dei veicoli. Come osservato in sezione II il vettore costante $\mathbf{1}$ è autovettore relativo all'autovalore nullo, perciò il protocollo (4) si rivela essere un algoritmo economico, nel senso che se i veicoli sono tutti nello stesso punto, allora essi rimangono fermi. La (5) ci aiuta a comprendere inoltre il significato del secondo autovalore λ_2 di L, chiamato "connettività algebrica". Esso è infatti un indice della velocità di convergenza dell'algoritmo: maggiore è, più velocemente si estingue il transitorio del sistema (5) e quindi più velocemente il protocollo (4) conduce alla convergenza. Per questo motivo l'analisi spettrale della laplaciana L è molto studiata da chi è interessato allo studio dei comportamenti emergenti in una rete di sistemi distribuiti e da chi si propone di progettare la disposizione ottima di una rete di sensori.

La convergenza dell'algoritmo appena esposto non è limitata ai soli grafi non orientati. In [1] si dimostra che anche per un generico grafo orientato purché bilanciato (i grafi non orientati sono un caso particolare di questa classe), il protocollo (4) garantisce la convergenza asintotica per ogni stato iniziale nel baricentro del sistema.

Per l'implementazione MATLAB e la simulazione del protocollo (4) è necessario procedere alla discretizzazione (del primo ordine) dell'algoritmo. Si ottiene:

$$q_i(t+1) = q_i(t) + \mathrm{dt} \sum_{j \in N_i} a_{ij}(q_j(t) - q_i(t)) \tag{6}$$

dove dt è l'intervallo di campionamento che si suppone sufficientemente piccolo. La (6) si può esprimere in forma matriciale:

$$q(t+1) = Pq(t) \tag{7}$$

dove la matrice $P = I - \mathrm{dt}L$ è la matrice di Perron di parametro dt del grafo. Dall'analisi condotta in sezione II possiamo dimostrare che la mappa (7) garantisce la convergenza dei robot nel baricentro della loro posizione iniziale $q(0)$, per ogni $q(0)$. Infatti per ipotesi la rete costituita dai veicoli è simmetrica, e quindi il grafo che disegna è non orientato e perciò bilanciato. Inoltre è anche fortemente connesso. Ne segue che la matrice P, per valori sufficientemente piccoli dell'intervallo di campionamento dt, è primitiva e doppiamente stocastica. Dalla teoria dei sistemi sappiamo che la soluzione di (7) è data da

$$q(t) = P^t q(0) \tag{8}$$

e si ha convergenza se $\lim_{t\to\infty} P^t < \infty$. Il teorema di Perron-Frobenius ci dice che il limite esiste per le matrici primitive. Ne segue che si ha

$$\lim_{t\to\infty} q(t) = v(w^T q(0)) \tag{9}$$

dove v e w sono gli autovettori destro e sinistro di P che soddisfano le ipotesi del teorema di Perron-Frobenius. Dato che P è doppiamente stocastica, ha autovettore destro pari a $v = \mathbf{1}$ e sinistro $w = 1/N\mathbf{1}$. Di conseguenza si ha la convergenza in

$$q_i(t) \longrightarrow \frac{1}{N}\mathbf{1}^T q(0) \tag{10}$$

vale a dire ogni veicolo converge nel baricentro.

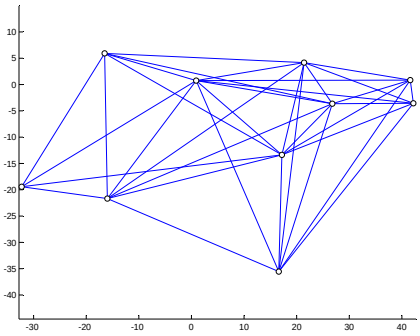
In figura 1 sono riportati i risultati della simulazione condotta al calcolatore. Il codice MATLAB è invece riportato in appendice III-B.

B. Veicoli con visibilità limitata: rete di sensori tempo-variante

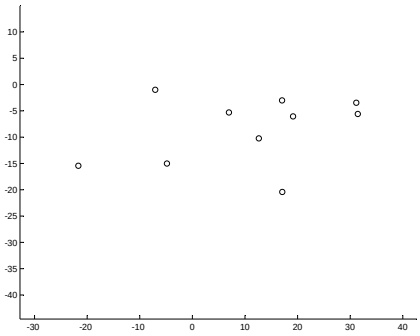
In molti casi pratici i veicoli montano on-board dei sensori che immagazzinano, elaborano e trasmettono informazioni. É ragionevole supporre che il raggio percettivo dei sensori sia finito, inoltre, con i veicoli, anche i sensori sono in continuo movimento. Avvicinandosi ed allontanandosi uno dall'altro, i veicoli possono perdere alcune connessioni e formarne di nuove, quindi si ha a che fare con una rete di sensori tempo-variante.

Sotto queste nuove condizioni il protocollo (4) sopra esposto non riesce a garantire la convergenza dei robot in un'unico punto. É evidente come sia probabile la perdita di connessione nel grafo di comunicazione, evento che provoca la separazione dei robot in gruppi e quindi il ricongiungimento dei veicoli in punti separati. Ogni veicolo, infatti, avvicinandosi al baricentro dei robot nella sua visuale, non garantisce il mantenimento della distanza di visibilità massima con tutti i veicoli in comunicazione all'istante iniziale. In figura 2 viene riportato un semplice esempio di come ciò sia possibile, mentre in figura 3 vengono riportati gli esiti della simulazione al computer che effettivamente provano come questo algoritmo non sia sodisfacente.

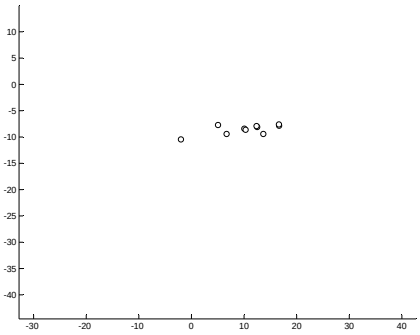
Introdurremo ora un algoritmo in grado di garantire la connessione del grafo. Nel seguito si suppone che ciascun robot abbia un raggio di visibiltà limitato $D > 0$ e che sia in grado di calcolare (o conoscere) la posizione relativa dei robot che sono entro la sua sfera di visibilità.



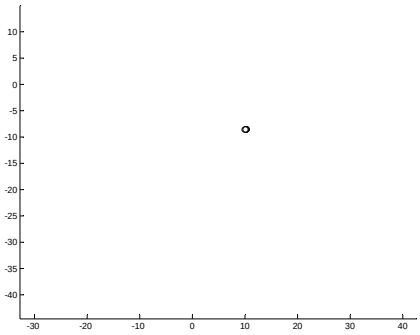
(a) t = 0



(b) t = 10

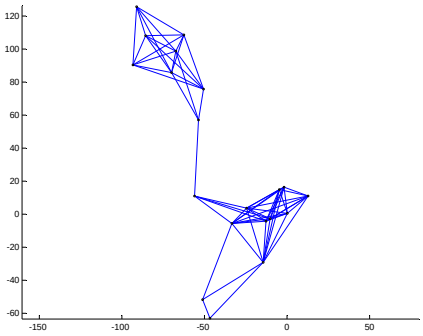


(c) t = 40

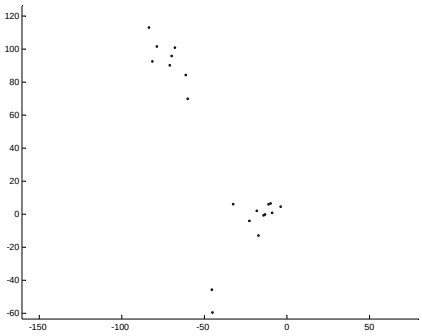


(d) t = 150

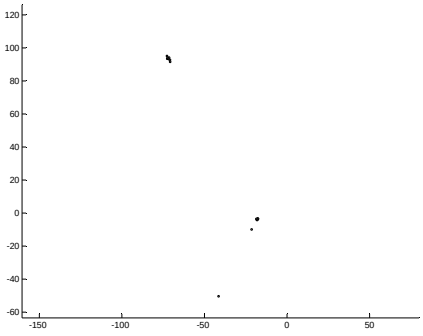
Fig. 1. Risultato della simulazione al calcolatore per $N = 10$ robot del protocollo (4). In sottofigura (a) è riportata la situazione iniziale con evienzialto il grafo di connessione dei veicoli; nelle sottofigure (b), (c), (d) è riportata la situazione per t=10, t=40, t=150.



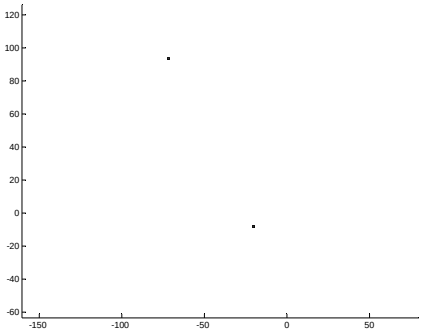
(a) t = 0



(b) t = 10

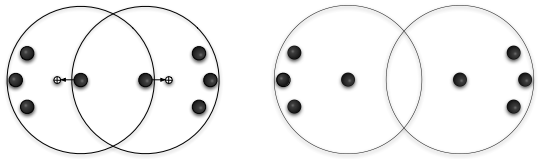


(c) t = 40



(d) t = 150

Fig. 3. Risultato della simulazione al calcolatore per $N = 20$ robot con visibilità limitata che operano secondo il protocollo (4) riportato in sezione III-A. Come si può osservare il grafo, inizialmente connesso, si suddivide in due sottografi isolati.



(a) grafo connesso (b) grafo sconnesso

Fig. 2. Esemplificazione schematica del caso in cui due veicoli, inizialmente in comunicazione, si sconnettono avvicinandosi al baricentro dei robot compresi nel raggio visivo

Si impone inoltre che la sua velocità massima sia limitata, cosi chè la distanza che percorre dalla sua posizione $q_i(t)$ all'istante t, alla successiva, $q_i(t+1)$, non possa eccedere una costante positiva $\delta > 0$.

L'algoritmo prevede che ripetutamente ogni robot:

1) calcoli la posizione relativa di quei robot che gli sono visibili;
2) individui i robot a massima distanza entro il suo raggio di visibilità: se sono più di uno, allora si pone
$$u_i = 0, \tag{11}$$
mentre se esiste un solo robot, v_i^M, a massima distanza si pone:
$$u_i = \sigma_i \frac{q_i^M(t) - q_i(t)}{\|q_i^M(t) - q_i(t)\|} \tag{12}$$
dove q_i^M è la posizione del veicolo v_i^M e σ_i è così definita:
$$\sigma_i = \min(\delta, \|q_i^M(t) - q_i(t)\|). \tag{13}$$
3) si muova verso la sua nuova posizione:
$$q_i(t+1) = q_i(t) + u_i \tag{14}$$

L'algoritmo che proponiamo prevede dunque che ad ogni intervallo temporale i robot si dirigano verso quel veicolo a maggior distanza entro la loro sfera di visibilità; se invece esiste più di un veicolo a maggior distanza entro la sfera di visibiltà di un robot, allora quel robot rimane fermo.

In appendice II si dimostra che questo algoritmo garantisce la connessione della rete di veicoli, e che sotto alcune ipotesi non restrittive garantisce la convergenza dei robot in un unico punto.

Il codice MATLAB che implementa questo protocollo è riportato in appendice III-C, mentre in figura 4 e 5 sono riportati i risultati della simulazione condotta al calcolatore, i quali evidenziano come ci sia effettivamente convergenza in un unico punto.

Teniamo a sottolineare che il protocollo (14) è distribuito e privo di memoria. Distribuito perchè ciascun agente stabilisce la propria traiettoria sulla base della sola informazione che riesce a ricavare dai propri sensori; senza memoria perchè la posizione $q_i(t+1)$ al tempo $t+1$ dell'i-esimo robot è calcolata a partire dalla posizione al tempo t dei robot che riesce a vedere, indipendentemente dalla storia precedente del sistema. Ne segue che l'algoritmo è facilmente implementabile. Inoltre un algoritmo senza memoria gode dell'importante proprietà di essere *auto-stabilizzante*, poichè lavora correttamente anche in presenza di un numero finito di errori transitori, dal momento che il verificarsi di un errore non si propaga nel tempo.

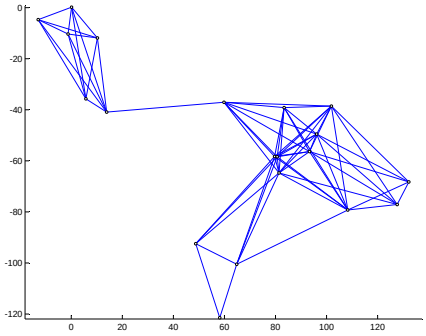
C. Analisi delle prestazioni

Analizzeremo ora le prestazioni del protocollo (14) illustrato in sezione III-B sotto condizioni più realistiche di quelle fin qui considerate. In particolare ci interessa studiarne il comportamento al variare del grado di sincronismo dei veicoli e in presenza di rumore di misura.

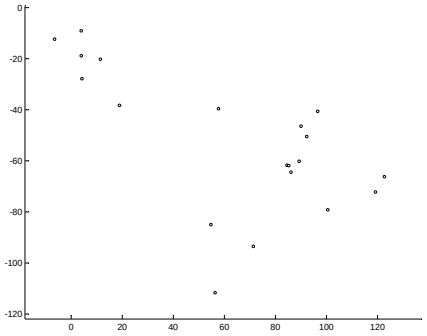
a) Sincronismo: In una rete reale di veicoli mobili è sensato supporre che il lavoro svolto dai robot non sia sincrono, ma che ciascun agente calcoli la propria posizione e si muova in maniera asincrona rispetto agli altri agenti. È importante dunque che un algoritmo di rendezvous sia robusto e non degradi le proprie prestazioni al variare del grado di sincronismo della rete.

Considereremo come indice di prestazione il numero medio di fasi compiute dai veicoli per entrare entro un'area di raggio prefissato e pari a tre. Il raggio di visibiltà dei veicoli sarà impostato a $D = 50$, mentre la massima distanza che un veicolo può compiere in una fase sarà $\delta = 1$. Si svolgeranno diversi esperimenti per $N = 10$, $N = 50$, $N = 100$ robot, utilizzando sempre la stessa configurazione iniziale di veicoli e facendo contemporaneamente variare la frazione λ_s di veicoli attivi in ciascun istante. Per ogni tipo di esperimento si faranno 10 prove diverse. I risultati che si sono ottenuti sono riportati in tabella I, mentre il listato completo dei risultati di simulazione è riportato in appendice IV-A. Come si può notare dalla tabella I, il numero medio di fasi di ciascun veicolo non subisce variazioni statisticamente significative al variare del grado di sincronismo λ_s.

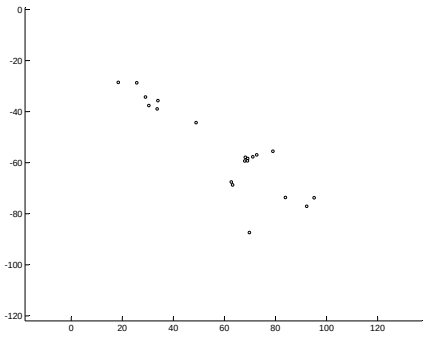
b) Rumore di misura: Abbiamo ipotizzato che i veicoli siano in grado di calcolare la posizione relativa degli altri agenti che li circondano utilizzando i sensori di cui sono dotati. Sappiamo che nella realtà i sensori forniscono misure corrotte da rumore. É dunque importante che un algoritmo di rendez-vous sia rubusto contro gli errori di misura.



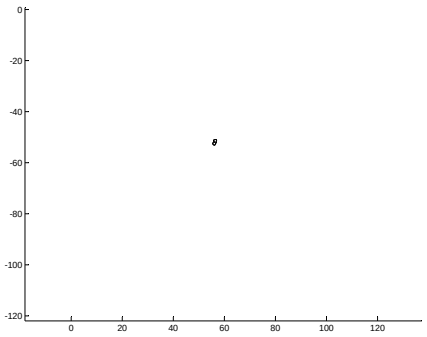
(a) t = 0



(b) t = 10

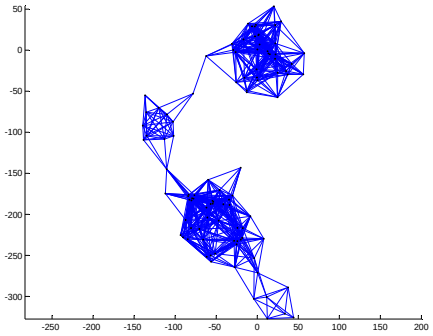


(c) t = 40

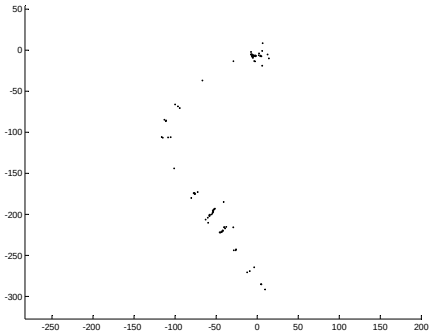


(d) t = 150

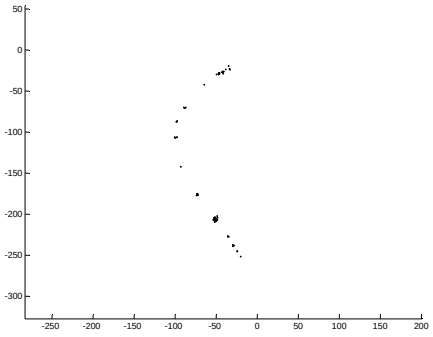
Fig. 4. Risultato della simulazione al calcolatore per $N = 20$ robot con visibilità limitata che operano secondo il protocollo (14). In sottofigura (a) è riportata la situazione di partenza ed il grafo di visibilità. Come si può osservare la connessione del grafo è mantenuta.



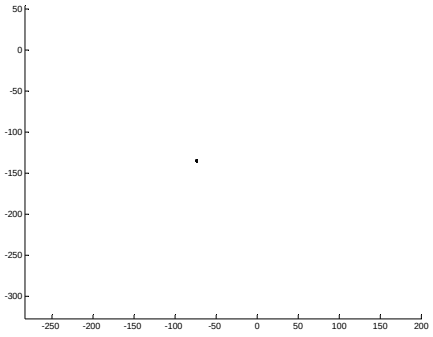
(a) t = 0



(b) t = 50



(c) t = 100



(d) t = 400

Fig. 5. Risultato della simulazione al calcolatore per $N = 100$ robot con visibilità limitata che operano secondo il protocollo (14). Come si può osservare, i veicoli isolati rimangono fermi per evitare la disconnessione del grafo.

TABLE I

RISULTATI DELLE SIMULAZIONI CON ROBOT ASINCRONI

	$N = 10$	$N = 50$	$N = 100$
$\lambda_s = 100\%$	63.00	166.00	133.00
$\lambda_s = 80\%$	64.11	169.50	135.57
$\lambda_s = 60\%$	65.72	167.92	134.86
$\lambda_s = 40\%$	65.13	172.95	135.06
$\lambda_s = 20\%$	67.01	172.03	136.97

Si suppone che questi errori siano la somma di due componenti distinte: errore di direzione ed errore di distanza. L'errore di direzione sarà specificato come il massimo scostamento assoluto in gradi dalla direzione vera, mentre l'errore di distanza come il massimo errore relativo percentuale. In particolare, se indichiamo con θ e d la direzione e la distanza vera di un robot rispetto ad un altro, la misura corrotta da rumore fornita dai sensori sarà pari a $\theta + \mu$ per quanto riguarda la direzione e $d(1 + \nu/100)$ per quanto riguarda la distanza, dove μ e ν sono variabili aleatorie uniformi $-E \leq \mu, \nu \leq E$, e E è un parametro.

Verranno condotte simulazioni per $N = 10, 25, 50$ veicoli, e per $E = 5, 10, 30, 50$. Si condurrano tre tipi di prove: rumore solo sulla direzione, rumore solo sulla distanza e rumore su direzione e distanza. Come prima, verrà considerato come indice di prestazione il numero medio di fasi eseguite da ciascun veicolo per giungere entro un disco di raggio tre e contenente tutti i robot. Le tabelle II, III e IV riportano i risultati delle simulazioni per i tre tipi di esperimenti. In appendice IV-B è riportato il listato completo dei dati di simulazione.

TABLE II

RISULTATI DELLE SIMULAZIONI CON ERRORE DI MISURA SULLA SOLA DIREZIONE

	$N = 10$	$N = 25$	$N = 50$
no errore	40.00	113.00	89.00
$E = 5$	40.00	113.50	88.80
$E = 10$	40.00	114.00	88.60
$E = 30$	41.70	119.50	92.50
$E = 50$	45.90	129.50	102.40

TABLE III

RISULTATI DELLE SIMULAZIONI CON ERRORE DI MISURA SULLA SOLA DISTANZA

	$N = 10$	$N = 25$	$N = 50$
no errore	37.00	72.00	116.00
$E = 5$	37.00	72.00	116.00
$E = 10$	37.00	72.00	116.00
$E = 30$	37.00	72.00	116.00
$E = 50$	37.00	72.00	116.00

TABLE IV

RISULTATI DELLE SIMULAZIONI CON ERRORE DI MISURA SU DIREZIONE E DISTANZA

	$N = 10$	$N = 25$	$N = 50$
no errore	51.00	143.00	168.00
$E = 5$	51.00	143.00	170.00
$E = 10$	51.00	143.20	169.70
$E = 30$	53.20	150.10	177.30
$E = 50$	58.20	163.30	191.60

Come si può osservare dai valori riportati nelle tabelle II, III, IV le prestazioni dell'algoritmo non sono compromesse dal rumore di misura. In particolare la componente di errore sulla distanza non influisce minimamente sull'algoritmo. All'aumentare dell'errore sulla direzione, invece, le prestazioni calano lievemente: per errori pari a $\pm 50^\circ$ si hanno prestazioni degradate di circa il 15%, che divengono del 5% circa per errori di $\pm 30^\circ$. I dati di tabella IV mostrano che le prestazioni del protocollo di rendez-vous proposto non peggiorano ulteriormente se coesistono errori sulla direzione e sulla distanza.

D. Introduzione dei vincoli anolonomi

Fino a questo punto é stato dimostrato che N veicoli in $\mathbb{R}^2$ con cinematica descritta da un integratore $\dot{q}_i = u_i$ e ingresso di controllo proporzionale alla distanza dal robot piú lontano mantengono la connessione reciproca e convergono in un unico punto. Si cercano di ampliare i risultati ottenuti in questo scenario lineare considerando ogni robot come un uniciclo cinematico con modello di stato non lineare:

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i & 0 \\ \sin\theta_i & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix}$$

dove $(x_i, y_i) \in \mathbb{R}^2$ indica la posizione dell'i-esimo uniciclo, $\theta_i \in [-\pi, \pi)$ é l'orientamento dell'uniciclo e $u_i = (v_i, \omega_i) \in \mathbb{R}^2$ sono gli ingressi di controllo, con v_i velocità lineare e w_i velocità di sterzo (vedi Figura 6) .

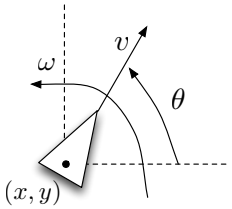


Fig. 6.

Chiameremo r_i la distanza tra gli unicicli i e $i + 1$ e α_i lo scostamento dell'angolazione dell'i-esimo vei-

colo rispetto all'orientamento che dovrebbe avere per avvicinarsi al veicolo i+1-esimo (vedi Figura 7).

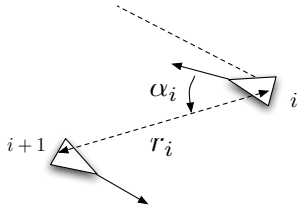


Fig. 7.

In modo analogo a quanto fatto nel modello lineare, si definisce in modo intuitivo la legge di controllo, assegnando al veicolo i-esimo velocitá lineare v_i proporzionale alla distanza r_i, e velocitá angolare ω_i proporzionale all'angolo α_i:

$$v_i = k_r r_i; \quad (15)$$

$$\omega_i = k_\alpha \alpha_i. \quad (16)$$

Al fine di studiare il comportamento qualitativo del sistema, si utilizzano per le simulazioni i valori $k_r = k_\alpha = 1$.

L'applicazione diretta appena esposta dei vincoli anolonomi al codice generato in precedenza, non garantisce la convergenza di tutti i veicoli in un unico punto. Infatti se la distanza tra due veicoli é la massima ma il loro orientamento é sfasato di $180°$, nel momento in cui partono tentando di avvicinarsi, ció che di fatto fanno é allontanarsi rompendo il legame di comunicazione (vedi Figura 8).

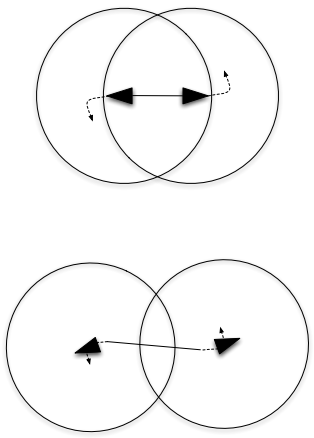


Fig. 8. Esemplificazione schematica del caso in cui due veicoli, inizialmente in comunicazione, si sconnettono a causa della traiettoria che sono costretti a seguire per rispettare i vincoli anolonomi

Per risolvere il problema si sceglie di mantenere nulla la velocitá lineare v_i di ogni veicolo nel caso in cui lo sfasamento α_i con il veicolo da inseguire sia maggiore di $60°$. In questo modo finché un robot non é rivolto verso il veicolo da inseguire rimane fermo, ruotando solamente su se stesso.

Sperimentalmente si osserva che con questo ulteriore vincolo la connessione dei veicoli non viene mai persa, come si può osservare in figura 9.

Il listato MATLAB che implementa questa versione dell'algortimo è riportato in appendice III-D.

E. Inseguimento circolare di unicicli

L'algoritmo elaborato nel paragrafo precedente per risolvere il problema del rendez-vous sotto l'ipotesi di vincoli anolonomi, puó venire adottato al fine di ottenere l'inseguimento circolare di un gruppo di unicicli

Scegliendo opportunamente il valore delle costanti di proporzionalitá velocitá-angolo e velocitá-distanza, ovvero k_α e k_r, é possibile forzare i veicoli a mantenere una disposizione regolare e stazionaria nel tempo.

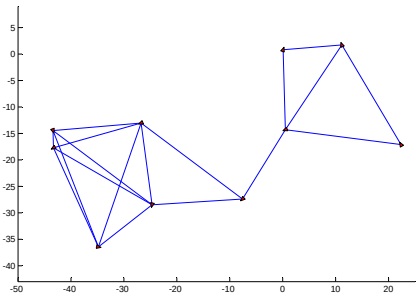
Sia $p = \frac{n}{d} \in \mathbb{Q}$ un numero razionale maggiore di 1 e R la rotazione positiva attorno all'origine dell'angolo $\frac{2\pi}{p}$. Dato $q_1 \neq 0$, definiamo il grafo con nodi $q_{i+1} = Rq_i$ ed archi $e_i = q_{i+1} - q_i$, detto *poligono regolare generalizzato*. Poiché p é razionale, R é periodico con periodo finito. Si distinguono tre tipologie di poligoni (vedi Figura 7):

- se n e d sono coprimi si ottiene un poligono regolare, ovvero con tutti i lati e tutti gli angoli uguali. In questa situazione se $d = 1$, ovvero $p = n$, gli archi del poligono non si intersecano, se invece $d > 1$ si ottiene un poligono a stella, in cui gli archi si intersecano in punti diversi dai vertici;
- se n e d hanno un fattore comune $m > 1$ allora il poligono ha $\tilde{n} = \frac{n}{m}$ vertici distinti e $\tilde{n}$ archi che li attraversano m volte;
- se $d = n$ tutti i vertici sono coincidenti.

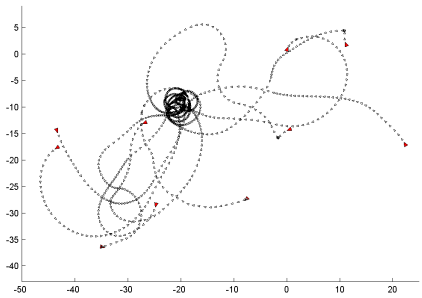
Si puó dimostrare che l'angolo interno ad ogni vertice misura $\pi(1 - \frac{2d}{n})$.

All'equilibrio, se i veicoli sono disposti in modo da inseguirsi circolarmente, le posizioni possono venire descritte da un poligono generalizzato regolare affine a quelli appena descritti. Si trascurano le situazioni in cui n e d sono coincidenti o non coprimi, in quanto questo richiederebbe che piú di un uniciclo occupasse la stessa posizione.

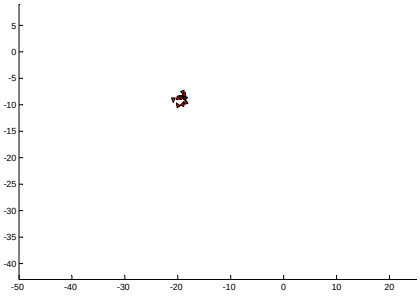
Graficamente si deduce che, per ogni possibile formazione, lo sfasamento della direzione di avanzamento di un veicolo da quella del robot che lo segue é di $\hat{\alpha} = \pm\pi\frac{d}{n}$, in quanto ogni robot punta in una direzione tangente al cerchio che circoscrive il poligono che descrive la posizione dei vertici.



(a) Disposizione iniziale



(b) Traiettorie



(c) Disposizione dei veicoli per $t = 1000$

Fig. 9. Risultato della simulazione al calcolatore per $N = 10$ robot con vincoli anolonomi. Come si può osservare dalla sottofigura (b) le traiettorie percorse dai veicoli sono abbastanza complicate, e la convergenza verso un unico punto è molto più lenta rispetto al caso lineare trattato in sezione III-B.

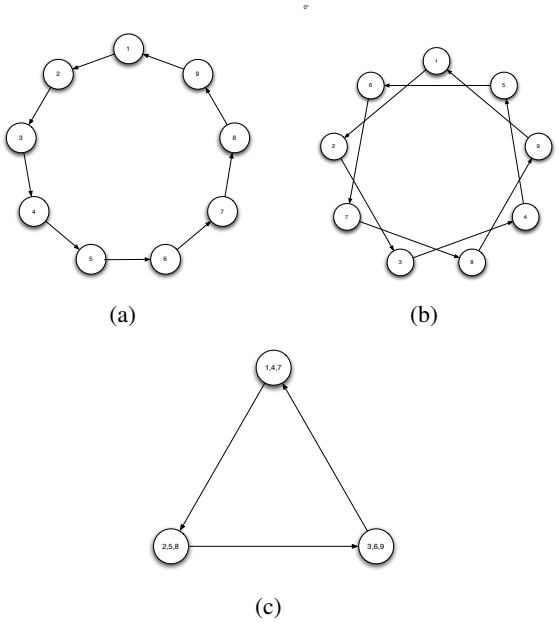


Fig. 10. Possibili configurazioni dei poligoni con $n = 9$. (a): $d = 1$, si ottiene un poligono regolare senza incrocio di archi; (b): $d = 2$, si ottiene un poligono a stella, infatti 9 e 2 sono coprimi; (c): $d = 3$, si ottiene un poligono senza incroci attraversato 3 volte, poiché $\tilde{n} = 3$ é fattore comune tra 9 e 3

Il mantenimento della posizione d'equilibrio risulta garantito se

$$\frac{k_r}{k_\alpha} = \frac{\pi d}{2n} \csc(\hat{\alpha}) = k^*$$

Senza perdita di generalitá si puó scegliere $k_\alpha = 1$ e $k_r = k^*$ per ottenere una formazione di equilibrio. Per una piú approfondita descrizione e per la dimostrazione si rimanda a [4].

In figura 11 è riporata una simulazione dell'algoritmo per l'inseguimento circolare, mentre il codice MATLAB è in appendice III-E.

IV. Accenno al problema del flocking

Il problema del flocking é quello di coordinare il movimento di un grande numero di robots interagenti, in modo da ottenere un movimento d'insieme comune.

Boids è un algoritmo concepito nel 1986 da Craig Reynolds come programma per la simulazione del comportamento di volo in stormi degli uccelli. É un programma per la simulazione artificiale della vita, ma rappresenta uno tra i primi esempi di algoritmo che simula un *emergent behaviour*, quindi, nonostante sia stato applicato principalmente nell'ambito della computer vision, l'algoritmo offre spunti interessanti anche per applicazioni di robotica e di controllo coordinato.

La complessità del comportamento emergente del flock ha origine dall'iterazione di individui indipendenti che aderiscono ad un insieme di semplici regole:

- **Separazione:** ogni individuo cerca di evitare di collidere con i vicini;
- **Allineamento delle velocità:** ogni individuo cerca di far combaciare la propria velocità a quella dei vicini;
- **Coesione:** ogni individuo cerca di avvicinarsi al centro del flock.

Ogni individuo è quindi mosso da due opposti ma bilanciati impulsi: quello di rimanere strettamente vicino al flock, e quello di evitare le collisioni interne al flock stesso.

Le tre regole esposte contribuiscono a determinare la velocitá con cui ogni veicolo si muove. Tale velocitá viene aggiornata di passo in passo sommando le tre componenti indipendentemente calcolate in modo da far rispettare ad ogni individuo le regole base di comportamento.

$$v(t+1) = v(t) + v_1(t) + v_2(t) + v_3(t)$$

Si assume che ogni individuo possa vedere tutti gli individui dello stormo e abbia velocitá massima limitata.

c) Coesione:: $v_1(t)$ è calcolata in modo che il veicolo si avvicini al centro di massa del flock. Si assume che il valore della velocità sia proporzionale alla distanza del veicolo dalle coordinate, tempo varianti, del centro di massa.

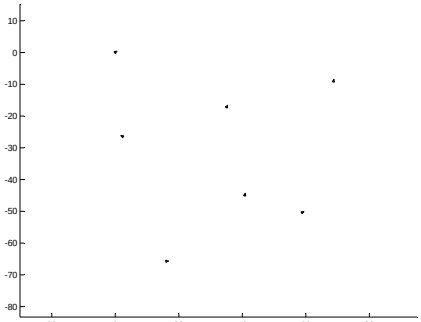
d) Allineamento delle velocità:: Similmente a quanto fatto nel caso precedente, $v_3(t)$ si assume proporzionale alla velocitá media dei veicoli che lo circondano. Questa componente fa in modo che il robot approssimi la propria velocitá a quella media.

e) Separazione:: Ogni veicolo verifica se ce ne sono altri all'interno di una definita distanza minima, `distMin` . Se ciò avviene, la velocità repulsiva $v_2(t)$ si calcola come opposto della somma della distanza dai veicoli più vicini di `distMin` .

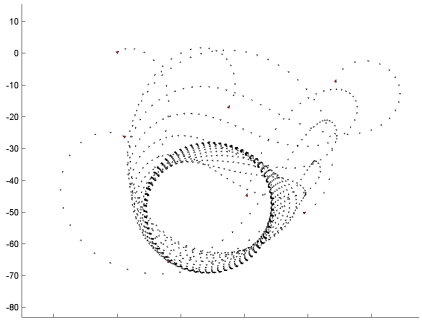
Questo modo di gestire i problemi di collisione è critico per le applicazioni di robotica, infatti ciò che avviene è che più sono vicini due veicoli, tanto meno si respingono. Il motivo per cui in ambiti di computer vision è conveniente usare un modello di questo tipo è che se due individui si trovano molto vicini l'un l'altro, ciò è evidentemente dovuto al fatto che negli istanti immediatamente precedenti stavano volando uno verso l'altro molto velocemente, e forzare una repentina inversione della velocità non apparirebbe realistico.

In appendice si può trovare l'implementazione dell'algoritmo (III-G), che genera i risultati in figura IV-.0.e.

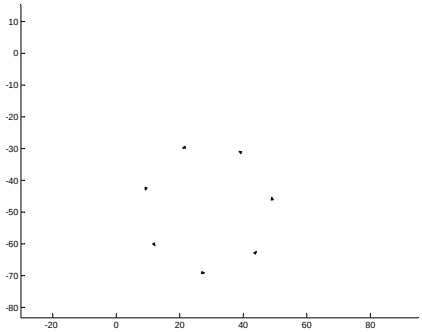
Il movimento risultante del flock si può descrivere come un "ordine caotico", infatti sebbene sia evidente



(a) Disposizione iniziale



(b) Traiettorie



(c) Disposizione dei veicoli per $t = 5000$

Fig. 11. Risultato della simulazione al calcolatore per $N = 7$ robot con vincoli anolonomi e in inseguimento circolare.

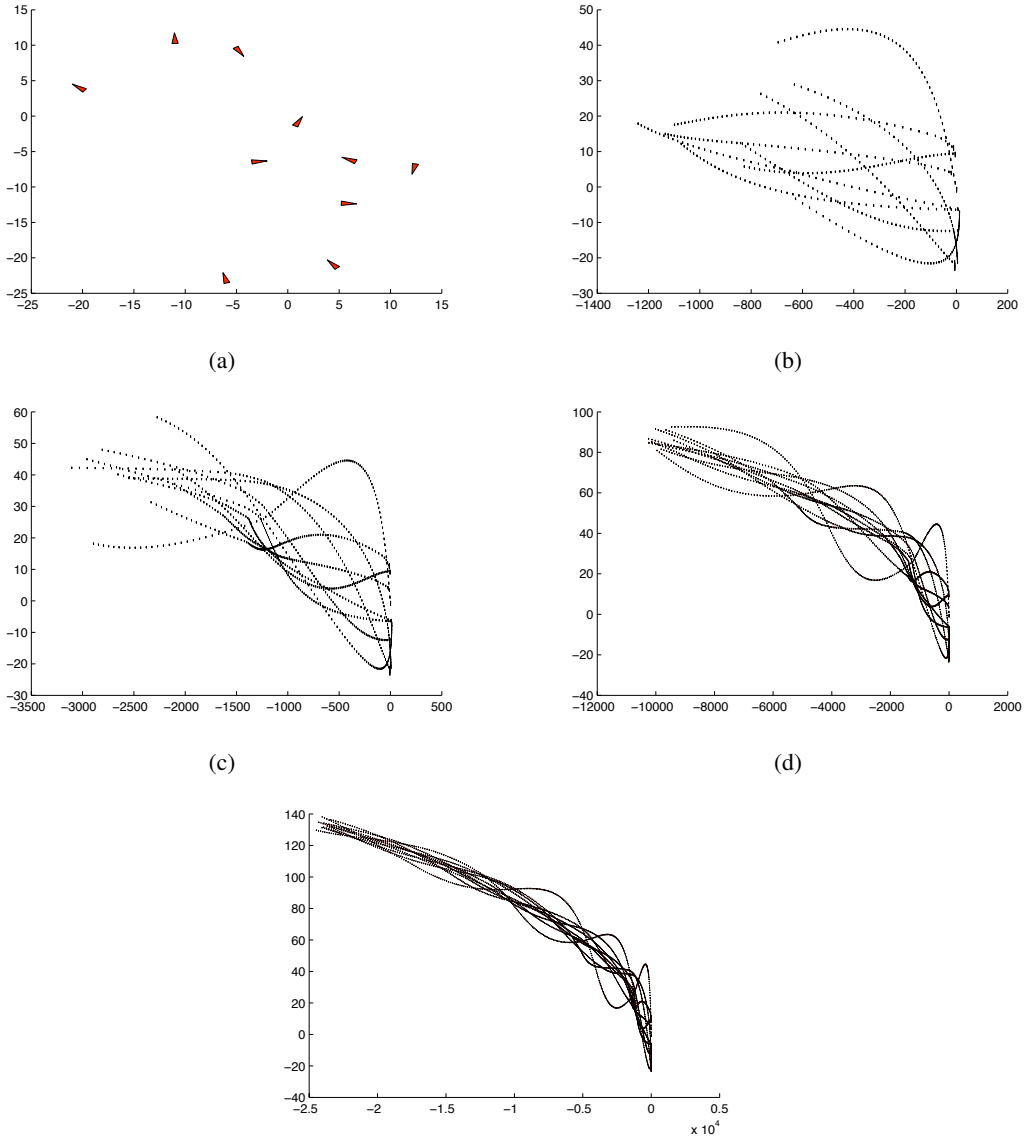


Fig. 12. N=10. Evoluzione del sistema simulato utilizzando l'algoritmo di Reynolds (a): $t = 1$, i veicoli sono distribuiti casualmente e hanno orientamento casuale; (b): $t = 50$, emerge una direzione prevalente, i veicoli diretti in direzione opposta correggono la loro traiettoria; (c), (d), (e): $t = 100$, $t = 200$, $t = 300$, i veicoli si allineano e col passare del tempo diminuiscono anche gli sbandamenti

un comportamento comune di tutti gli individui, ognuno si muove in modo diverso.

Nel caso ci siano ostacoli da superare si possono anche verificare fenomeni di rottura e riunione del flock (caso non simulato).

Ogni qual volta un individuo desidera aggiungersi al flock non si verificano problemi di sovraccarico, e gli individui stessi non modificano il loro comportamento in base alle dimensioni del flock, che non ha quindi dei vincoli dimensionali.

Come già ripetuto, la struttura dell'algoritmo è stata spesso usata in computer graphic (*Il Re Leone*, *Batman Return*, ...) in quanto origina una rappresentazione molto realistica del complesso volo di stormi di uccelli, del movimento di banchi di pesci, greggi di animali, movimenti d'insieme che sarebbe molto difficile da creare diversamente. La logica dell'algoritmo di Reyenolds, ovvero che per rappresentare un flock é sufficiente descrivere il comportamento di ogni individuo come la somma di molteplici vincoli, può essere applicato con buoni risultati anche nell'ambito di nostro interesse dei controlli automatici (vedi articolo [6]).

Il principale problema da affrontare per adattare l'algoritmo è la gestione delle collisioni, che in realtà Reynolds affronta solamente dal punto di vista del "realismo visivo".

La gestione delle collisioni si basa sula posizione relativa dei vari individui trascurando la loro velocità. L'alineamento delle velocità, invece, si basa solamente sulle velocità, trascurando completamente la posizione. E' però evidente che un buon maching diminuisce la probabilità di collisione, mantenendo approssimativamente invariata la distanza tra gli individui. La gestione delle collisioni stabilisce la minima distanza di separazione richiesta, il matching delle velocità garantisce che venga mantenuta.

V. Deployment e Exploration

Un'altra importante branca d'utilizzo di robot autonomi è quella che si occupa della loro gestione per l'esplorazione, l'occupazione o la sorveglianza di una superficie. Le componenti principali da considerare per la soluzione di questo genere di problema sono, ancora una volta, la capacità di comunicazione tra robot, la programmazione delle loro facoltà decisionali, il trade-off tra organizzazione centralizzata e autonoma, lo sviluppo di algoritmi robusti al decadimento di prestazioni dovuto alle condizioni reali rispetto a quelle simulative e, ovviamente, il raggiungimento di prestazioni adeguate al compito che viene loro assegnato.

Questo insieme di situazioni è accomunato dalla necessità di riuscire a gestire non soltanto il comportamento reciproco tra robot ma anche il compimento di una missione sia attraverso check-point successivi (il che implica una sorta di pianificazione di traiettoria o comunque il rispetto di vincoli legati all'evoluzione spazio-temporale del gruppo di robot), che attraverso un obiettivo comune noto a tutto il gruppo e da realizzare attraverso il massimo sfruttamento delle possibilità cooperative del team.

Oltre ai vincoli di comunicazione analizzati nei capitoli precedenti, se ne aggiungono di ulteriori dovuti alla caratterizzazione spaziale dell'ambiente in cui si muovono i robot che comportano l'insorgere di ulteriori complicazioni. La distanza euclidea tra robot può non rappresentare la distanza che i veicoli devono percorrere per raggiungersi (questo avviene, ad esempio, quando due veicoli viaggiano vicini ma separati da una parete), inoltre puó verificarsi l'interruzione inaspettata delle comunicazioni se queste sono basate su protocolli che prevedono la necessità di campo libero tra due agenti. Quest'ultimo problema si puó facilmente eludere nella maggior parte degli ambienti, poichè possono essere utilizzate tecniche di comunicazione wireless ormai affidabili, ma nel caso in cui ci sia la necessità per ogni robot di vedere i compagni, utilizzando quindi delle telecamere per riconoscere gli altri veicoli, non si puó evitare di affrontare il problema.

Le difficoltá dovute alla presenza di impedimenti fisici non si possono trascurare, visto che il rilevamento degli ostacoli é spesso lo scopo per il quale questi stessi algoritmi vengono sviluppati .

Come in precedenza, si assume che ogni veicolo sia caratterizzato da un raggio visivo limitato, è quindi necessario che, in ogni situazione in cui si rende necessaria la cooperazione tra robot, gli algoritmi di esplorazione garantiscano innanzitutto il mantenimento delle connessioni.

A. Gestione delle informazioni

Consideriamo il caso in cui lo scopo del team di robot sia quello di costruire la mappa di una zona inesplorata. Il principale problema da gestire é l'aggiornamento e l'unione delle informazioni riguardanti il territorio elaborate da ogni singolo veicolo durante il movimento, in modo da evitare che si continuino ad esplorare zone giá note. Questo problema, detto *map merging*, deve essere gestito indipendentemente dagli algoritmi implementati per l'organizzazione del movimento del team, la creazione della mappa e la comunicazione tra robot o con il server.

È possibile distinguere due grandi tipologie di gestione delle informazioni:

- *Gestione Centralizzata:* é caratterizzata da un *server* che si occupa di ricevere i dati da ogni singolo *client* della rete (in questo caso i veicoli) e ritrasmettere le informazioni elaborate;
- *Gestione Distribuita:* lo scambio di dati e l'elaborazione delle informazioni è gestito autonomamente dalla rete di robot.

Se viene utilizzata una gestione del team di tipo centralizzato, il problema si puó risolvere banalmente: con una determinata frequenza, ogni robot comunica la propria mappa. Il server ha il compito, ad ogni nuovo arrivo di informazioni, di aggiornare la mappa e rimandarla agli agenti. L'unica condizione da garantire in questo caso, perciò, risulta essere quella di mantenere possibile, per ogni robot, la comunicazione con il server.

Se si assume invece una completa autonomia dei veicoli, si può implementare la stessa tecnica assumendo che, istante per istante, ogni robot consideri come server tutti i robot con cui può comunicare. In questa situazione, quindi, con fequenza assegnata ogni robot comunica la sua mappa a tutti i veicoli nella sua area di comunicazione e, allo stesso tempo, elabora ed aggiorna la propria mappa con le nuove informazioni ottenute.

La centralizzazione delle informazioni sgrava i singoli robot dalla necessità di eseguire l'elaborazione di dati generalmente complessi per il continuo aggiornamento

della mappa ma, allo stesso tempo, necessita che l'unità di calcolo sia in comunicazione con ogni robot.

B. Occupazione di un'area nota

L'esplorazione o l'occupazione di un'area non richiede necessariamente l'utilizzo di algoritmi che prevedono la dispersione dei veicoli o un loro movimento pseudo-casuale. Spesso, in condizioni di esplorazione o sorveglianza di una mappa nota, è sufficiente e conveniente imporre ai robot di raggiungere *check point* successivi.

Si illustrano ora alcuni algoritmi che utilizzano un approccio delocalizzato, allo scopo di far raggiungere ad un gruppo di veicoli autonomi una serie di check-point spaziali successivi dove, ad ogni obiettivo raggiunto, un elemento si ferma.

Si assume che ogni check-point corrisponda ad una posizione che deve essere occupata da un robot, per cui il numero di check-point è posto uguale al numero di agenti e la disposizione di queste posizioni è scelta in modo da garantire la comunicazione necessaria tra i veicoli. Non viene discussa l'implementazione del riconoscimento degli obiettivi da parte dei robot, gestibile ad esempio tramite l'utilizzo della rete GPS oppure utilizzando marker visivi se i veicoli sono dotati di telecamere, in quanto esula dall'organizzazione del movimento del team.

1) Group deployment: Un approccio possibile è quello di lasciar muovere casualmente tutti i robot e imporre, ogni volta che un veicolo raggiunge un obiettivo, che lo comunichi agli altri. Alla ricezione del segnale gli altri membri del team rispondono con una conferma. Quando il robot che ha lanciato il primo messaggio, riceve tutte le conferme si ferma, e tutti gli altri veicolo sanno che il check point é stato raggiunto e ci si puó dirigere verso il successivo. Ad ogni check point raggiunto, diminuisce il numero di agenti coinvolti nel movimento, quindi le difficoltá causate dalla necessitá di gestire collisioni ed ingorghi sono evidenti soprattutto all'inizio dell'operazione.

Le probabilitá che giunga una richiesta di stop multipla sono minime, soprattutto in condizioni in cui il movimento dei robot sia 'lento' rispetto ai tempi di comunicazione, e in ogni caso si possono gestire attraverso ulteriori richieste di conferme temporizzate da latenze aleatorie.

Questo genere di assegnazione è simile alla gestione delle reti di comunicazioni di tipo *token bus*, molto utilizzate per la coordinazione di utenti all'interno di reti cablate.

2) Wave deployment: Per risolvere le difficoltá che insorgono con il *group deployment*, si puó utilizzare un diverso algoritmo detto *wave deployment*. I robot non partono contemporaneamente ma uno alla volta, e ogni veicolo transita successivamente sui vari check point. Ogni robot parte solamente quando il veicolo che lo precede supera il primo check point, generando in questo modo un movimento "ad onda". In questo algoritmo il numero di veicoli che si muovono contemporaneamente cresce nel tempo ma non c'é mai un raggruppamento di robot, visto che quando sono in movimento sono sempre distanti l'uno dall'altro se i check point sono stati ordinati in modo che le traiettorie che li collegano non si intersechino.

Lo svantaggio di questo algoritmo é che si rende necessaria l'assegnazione dell'ordine di partenza, quindi é utilizzabile solamente quando è possibile numerare i veicoli prima dell'inizio della loro operazione.

3) Leap-Frog deployment: Questo algoritmo prevede che un robot si sposti dalla base al primo obiettivo. A questo punto un secondo robot lo raggiunge e poi prosegue fino al secondo obiettivo. Una volta raggiunto parte un terzo che raggiunge il primo, il secondo e si posiziona nel terzo check-point e così via. Questo metodo presenta le stesse problematiche di numerazione dell'algoritmo "ad onda", ma a differenza degli algoritmi esposti, prevede che per tutto il tempo dell'operazione sia sempre soltanto uno il veicolo in movimento. Di contro, mentre per gli altri due metodi le comunicazione è prevista solo tra robot che sono in contatto diretto, per la realizzazione di questo algoritmo è necessario che il robot n-esimo, una volta raggiunta la sua posizione, diffonda la comunicazione fino alla base sfruttando il ponte di comunicazione creato dagli altri veicoli, con l'ovvio aumento del tempo di propagazione dell'informazione e della probabilità di perdita di pacchetti.

C. Esplorazione di una mappa ignota

Le tecniche per coordinare il movimento di un gruppo di robot con l'obiettivo di esplorare una zona dai contorni sconusciuti ha bisogno di un approccio differente da quelli appena visti, é ovvia infatti l'impossibilità di posizionare dei check-point da far raggiungere. É importante innanzitutto definire lo scopo che i veicoli devono perseguire, in modo da stabilire delle prioritá nei compiti assegnati ad ogni robot:

- puó essere necessario coprire quanta piú area possibile al fine di sorvegliare la zona o rilevare elementi particolari (ad esempio mine);
- puó essere necessario definire una mappa della zona non conosciuta.

1) Massima copertura d'area: Se lo scopo del team è quello di massimizzare la loro distanza mantendendo il contatto visivo, il problema è molto simile a quello del

rendez-vous, e le soluzioni hanno come denominatore comune l'allontanamento dei robot da tutti i vicini, con l'unico vincolo di non farli uscire da una distanza massima in modo da evitare la perdita delle connessioni. Quello che differenzia gli approcci e ne influenza la complessità, è il modo in cui si gestisce la possibilità dei robot di comunicare e di vedere i veicoli durante l'allontanamento.

Batalin e Sukhatme [7] propongono due semplici metodi, qui descritti in ordine decrescente di complessità.

Il primo, chiamato *Molecular*, impone che ogni robot si allontani il più possibile dai suoi vicini più prossimi. È un algoritmo simile a quello proposto per il rendez-vous, gestito però in modo inverso. Può essere scelto di voler mantenere il grafo connesso (con ovvie complicanze) oppure non considerare questo aspetto, nel caso in cui la connessione tra robot non sia strettamente necessaria, per migliorare le prestazioni.

Una seconda e ancora più elementare tecnica, definita *Basic*, impone semplicemente ad ogni veicolo l'allontanamento dal punto di partenza e non prevede nemmeno il riconoscimento da parte dei robot dei propri compagni, considerati semplicemente degli ostacoli da evitare durante il movimento. É un algoritmo greedy che richiede solamente l'implementazione della gestione delle collisioni.

Tipicamente l'esplorazione ha come parametri di prestazione il tempo necessario alla generazione della stessa, la precisione nella generazione della mappa e la percentuale d'area coperta. Le prestazioni dell'algoritmo *Basic* sono inferiori rispetto a quelle degli altri metodi, che invece piú o meno si equivalgono.

2) Wall-Following: Se l'obiettivo dell'esplorazione è quello di definire una mappa e sono importanti quindi i contorni dell'area da esplorare, un metodo efficace e semplice allo stesso tempo è forzare ogni veicolo a seguire i muri, una volta trovati, percorrendoli sempre nella stessa direzione.

Questa tecnica può essere implementata anche se il robot esploratore non sfrutta la conoscenza sul territorio giá acquisita, utile per evitare di ripercorrere le stesse zone, ma ció comporta un ovvio decadimento di prestazioni e soprattutto il rischio di entrare in un circolo vizioso nel caso in cui, ad esempio, ci si imbatta in una colonna all'interno di una stanza, che verrebbe seguita all'infinito una volta individuata.

Un altro problema è quello della tendenza di "non vedere" l'interno delle zone esplorate, perchè l'algoritmo fa rimanere i veicoli vicino ai muri individuati. Anche questo può essere risolto sfruttando la conoscenza della mappa generata, che una volta definiti tutti i bordi obbliga i robot a muoversi verso il centro per non muoversi ancora in zone già definite, ma un modo sub-ottimo per realizzarlo puó essere anche l'implementazione di una routine che obblighi il robot a muoversi verso il centro della stanza e non piú seguendo il muro, ogni volta che viene individuato un angolo. Questo metodo porta sicuramente a mappature più lente, ma garantisce l'esplorazione anche della zona centrale.

Nel caso ci siano più veicoli che implementano la stessa tecnica, è sufficiente aggiungere un semplice algoritmo di dispersione rispetto ai compagni, come quelli visti precedentemente, da affiancare a questo.

3) Social potential fields: Un altro metodo di esplorazione è quello che si basa sui concetti di forze e campi per decidere il movimento da far intraprendere a ciascun robot. Vengono introdotte tre forze, in modo analogo a quanto visto per gli algoritmi di flocking, la cui somma genera la forza totale che determina direzione, verso e intensità del movimento:

- una forza repulsiva introdotta dagli ostacoli visibili presenti nell'area da esplorare
- una forza repulsiva introdotta dai veicoli più vicini
- una forza attrattiva generata dalla frontiera della zona inesplorata

$$\vec{F}_{totale} = \sum \vec{F}_{ostacoli} + \sum \vec{F}_{robot} + \sum \vec{F}_{frontiera}$$

A seconda del raggio di visibilità dei robot, il numero di forze elementari presenti in ogni sommatoria può essere differente. Queste forze vengono definite come potenziali il cui modulo è proporzionale all'inverso della distanza (euclidea) dell'elemento dal robot, quindi tutte del tipo

$$\vec{F_i} = \frac{-k_j \cdot \vec{d_i}}{\|d_i\|^{\frac{3}{2}}}$$

dove k_j è un guadagno che dipende dal tipo di forza presa in considerazione. Questo tipo di algoritmo, però, non garantisce la copertura massima dell'area inesplorata, infatti potrebbero presentarsi situazioni di minimo delle forze che portano a configurazioni di stallo del team prima di che l'esplorazione sia completa.

D. Problematiche e Future Works

Un problema fondamentale e molto attuale nel settore della coordinazione di veicoli autonomi per l'esplorazione è la sicurezza e la perfetta integrazione con team umani. È necessario, in presenza di situazioni in cui ai robot venga chiesto di svolgere un compito in concomitanza di squadre di esseri umani, che i veicoli non disturbino l'azione degli umani e non possano recar loro danno. Questo è il caso di operazioni militari ma

anche di operazioni di soccorso, dove ai robot viene richiesto di esplorare zone con sensori che l'uomo non ha, ma allo stesso tempo viene richiesta la presenza umana per compiere i salvataggi o i recuperi.

È evidente, quindi, che per avere la certezza che gli algoritmi implementati siano sicuri è necessaria una giustificazine teorica, spesso difficile da derivare. In più, per evitare che il movimento dei robot possa essere d'ostacolo per i team umani, è necessario che le dinamiche dei veicoli siano molto veloci e reattive nell'eseguire eventuali manovre atte a liberare il campo. Questo problema perciò coinvolge la struttura anolonoma degli agenti, caratteristica che ne limita i movimenti e può impedire alcune traiettorie necessarie a non essere d'intralcio.

Diverso, ma altrettanto importante come problema da risolvere in modo semplice ed efficace (soprattutto in mancanza di veicoli con molti tipi di sensori e in condizioni ideali) è quello del risconoscimento dei compagni nella generazione della mappa. Se si usano telecamere l'utilizzo di marker particolari può parzialmente risolvere il problema, ma se la definizione delle frontiere delle zone esplorate è effettuata in modi diversi (laser o radar, ad esempio), questa operazione diventa difficoltosa, perchè deve essere realizzata attraverso la comunicazione tra robot della loro posizione reciproca o attraverso informazioni centralizzate della posizione di ogni robot, con ovvi problemi di temporizzazioni, ritardi e possibilità di errori da gestire.

In generale, il problema dell'esplorazione è molto aperto e presenta moltissime soluzioni, generalmente molto specifiche sia per quanto riguarda le specifiche dei robot da utilizzare, sia per le operazioni che svolgono, sia per il contesto in cui vengono calati; questo porta all'osservazione di risultati sulla base di simulzioni ed esperimenti, spesso poco supportati da garanzie di teoremi e dimostrazioni, che renderebbero molto più robusti gli algoritmi.

E. Simulazioni

Per cercare di visualizzare con maggiore chiarezza la struttura dei problemi descritti si è cercato di realizzare un semplice esempio di *exploration* da parte di un singolo robot all'interno di una zona ignota, con lo scopo di definire la mappa della superficie esplorata. L'algoritmo utilizzato può essere riassunto facilmente come 'se vedi un muro seguilo, altrimenti cercalo'. Più in dettaglio, si è cercato di implementare un algoritmo decisionale per il veicolo che lo porta a compiere una passeggiata aleatoria nel caso in cui nel suo raggio visivo non ci sono muri, mentre segue la traiettoria di un muro nel caso questo entri nel suo raggio visivo.

Si è subito notato come fossero necessarie, anche in un esempio così elementare, alcune accortezze per il corretto raggiungimento dell'obiettivo: è stato necessario infatti imporre un *verso di percorrenza* al movimento del veicolo, cioè obbligare il robot a eseguire l'esplorazione in verso orario o antiorario, ma non a caso. In una realizzazione in cui di volta in volta veniva scelta a caso la direzione in cui percorrere ogni singolo muro, infatti, si è subito notato che nella maggior parte dei casi questo portava allo stagnamento dell'esplorazione in particolari zone, dovuto alla 'media nulla' della direzione di esplorazione.

Un altro aspetto che è stato necessario trattare è stato la gestione degli angoli. Per la semplicità dell'algoritmo, infatti, la presenza di due muri nel raggio visivo creava ambiguità e difficiltà dell'esplorazione, empasse superata gestendo separatamente questi casi, imponendo al veicolo un movimento ad allontanarsi rispetto all'angolo di una distanza aleatoria.

La realizzazione in MATLAB di questo esempio è molto specifica rispetto alla mappa trattata, in quanto la realizzazione di un ambiente robusto e flessibile esulava dai propositi che questa simulazione voleva mostrare. Una realizzazione di questo algoritmo che ne permettesse il funzionamento in una qualsiasi mappa, infatti, avrebbe introdotto moltissimi problemi di programmazione completamente avulsi dalle problematiche di esplorazione trattate. In un caso reale, infatti, sarebbe solamente necessaria la definizione di una struttura di individuazione dei muri (attraverso ad esempio una telecamera o laser o radar); una volta realizzata questa facoltà del veicolo, l'algoritmo impone solamente di seguire la direzione del muro se è nel campo visivo (decidendo a priori un verso di percorrenza sempre uguale) o effettuare un movimento aleatorio se non si vedono muri.

Per la definizione della mappa, ancora una volta la semplicità è estrema: passo per passo, quando l'agente si trova nella situazione in cui vede un muro, semplicemente marca il punto in cui vede il muro in una griglia di due dimensioni. Per quanto riguarda la capacità di riconoscere la propria posizione all'interno di questa griglia del robot, può essere utilizzato il GPS o, molto più semplicemnte, ricavata la posizione passo per passo attraverso il movimento delle ruote considerando come origine il punto in cui parte l'esplorazione.

Sempre secondo l'idea di mostrare principalmente come un'idea semplice potesse comunque essere efficace senza introdurre restrizioni non necessarie, è stato scelto di non inserire i vincoli anolonomi, che, seppur mutuabili direttamente dalle simulazioni trattate per le altre sezioni, in questo esempio non avrebbero cambiato di molto il risultato delle simulazioni.

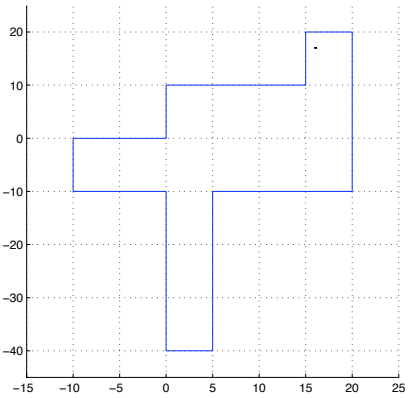


Fig. 13. Struttura della zona da esplorare e condizione iniziale del veicolo (alto a destra)

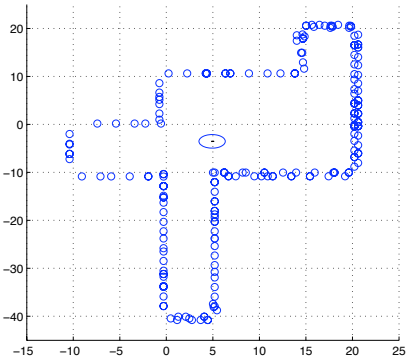


Fig. 14. Definizione della mappa da parte del veicolo dopo 400 passi

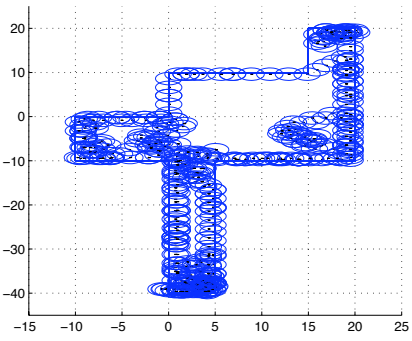


Fig. 15. Zona 'vista' dal veicolo nei 400 passi di esplorazione

APPENDIX I
DIMOSTRAZIONE DELLA CONVERGENZA NEL BARICENTRO DEL PROTOCOLLO (4)

Il protocollo (4) puó essere espresso in forma matriciale:

$$\dot{q} = -Lq \ , \ q \in \mathbb{C}$$

dove L é la matrice laplaciana del grafo, quindi, come noto, $\mathbf{1}$ é autovettore relativo alll'autovalore 0. Questo significa che $\mathbf{1}$ é generatore dello spazio nullo $\mathcal{N}$.

Preso un qualsiasi stato iniziale $q(0)$, questo può essere fattorizzato nella forma:

$$q(0) = c\mathbf{1} + \omega, \quad c \in \mathcal{N}, \ \omega \perp \mathcal{N}$$

Quindi l'evoluzione del sistema può essere scritta come segue:

$$q(t) = e^{-Lt}q(0) = ce^{-Lt}\mathbf{1} + e^{-Lt}\omega = c\mathbf{1} + e^{-Lt}\omega$$

Il grafo G è fortemente connesso, ne segue che l'autovalore nullo è isolato e tutti gli altri autovalori di $-L$ sono a parte reale strettamente negativa. Allora:

$$\lim_{t\to\infty} q(t) = c\mathbf{1}$$

quindi tutti i veicoli convergono esponenzialmente in c.

Il grafo che descrive le interazioni tra i robot è per ipotesi non orientato, di conseguenza $a_{ij} = a_{ji}$, e perció la somma degli stati di tutti i nodi è una quantità invariante:

$$\sum_i \dot{q}_i = 0 \quad \Rightarrow \quad \sum_i q_i = c\mathbf{1} \tag{17}$$

Applicando la (17) due volte, al tempo $t = 0$ e $t = \infty$ si ottiene

$$c = \frac{1}{N}\sum_i q_i(0) \tag{18}$$

ovvero c, stazionario nel tempo, è il baricentro delle posizioni iniziali $q(0)$.

APPENDIX II
DIMOSTRAZIONI DELLA CONNESSIONE E DELLA CONVERGENZA DEL PROTOCOLLO (14)

In questa sezione si dimostrerà che il protocollo di rendez-vous (14) introdotto in sezione III-B garantisce la connessione del grafo, e sotto alcune ipotesi anche la convergenza dei robot mobili verso un unico punto.

A. Ipotesi sui veicoli e Algoritmo a tempo continuo

Si suppone che il movimento di ogni veicolo possa essere descritto da un integratore lineare a tempo continuo

$$\dot{q}_i(t) = u_i(t) \tag{19}$$

dove la variabile complessa $q_i = x_i + jy_i$ rappresenta la posizione dell'i-esimo veicolo. Ogni robot ha un raggio di visibilità limitato, per cui non può ottenere informazioni su quei veicoli la cui distanza supera una costante $D > 0$.

L'algoritmo prevede che ripetutamente ogni robot:

1) calcoli la posizione relativa di quei robot che gli sono visibili;
2) individui i robot a massima distanza entro il suo raggio di visibilità:
   a) se questi sono più di uno, allora si pone:
   $$\dot{q}_i(t) = 0, \tag{20}$$
   b) mentre se esiste un solo robot, v_i^M, a massima distanza da v_i ed entro il raggio di visibilità di v_i si pone:
   $$\dot{q}_i(t) = q_i^M(t) - q_i(t) \tag{21}$$
   dove q_i^M è la posizione del veicolo v_i^M;

B. Mantenimento della connessione

Si dimostrerà che l'algortimo a tempo continuo appena esposto garantisce la connessione della rete costituita dai veicoli mobili.

Sia $G(t) = (V, E(t))$ il grafo non orientato e connesso che descrive la rete costituita dai robot. Per dimostrare che la connessione del grafo non viene persa è sufficiente dimostrare che ogni nodo v_i non perde mai la connessione con ogni suo vicino $v_j \in N_i$.

Per ipotesi, $\forall v_i \in V$ e $\forall v_j \in N_i$:

$$d(v_i, v_j) \leq D \tag{22}$$

$$\frac{d}{dt} d(v_i, v_j) \lesseqgtr 0 \tag{23}$$

1) Se $\frac{d}{dt} d(v_i, v_j) \leq 0$, il nodo v_j non si allontana dal nodo v_i, per cui la connessione tra i due nodi viene conservata.
2) Se $\frac{d}{dt} d(v_i, v_j) > 0$ allora il nodo v_j si allontana da v_i. Per la (21) ciò significa che esiste un nodo $v_r \in V$, $v_r \neq v_i$ tale che:
   $$D \geq d(v_j, v_r) > d(v_i, v_j) \tag{24}$$
   e di conseguenza $d(v_i, v_j) < D$.
   Se $d(v_i, v_j)$ continua a crescere, per la continuità della traiettoria di v_j, esiste un istante di tempo $\bar{t}$ in cui vale:
   $$d(v_i(\bar{t}), v_j(\bar{t})) = D \tag{25}$$
   e ciò implica che:
   $$d(v_j(\bar{t}), v_r(\bar{t})) = D \tag{26}$$
   in quanto, per ipotesi, il veicolo v_j si avvicina al suo veicolo a massima distanza v_r, distanza che non può eccedere D per la (22). Ne segue che all'istante $\bar{t}$, il robot v_j vede due veicoli, v_i e v_r, alla massima distanza, e dunque rimane fermo.
   A questo punto il veicolo v_j, o è il veicolo a massima distanza per v_i, che quindi vi si avvicina, oppure esiste un altro veicolo $v_q \in N_i$ a distanza $d(v_i, v_q) = D$, e di conseguenza anche v_i rimane fermo alla distanza D da v_j.
   Ne segue che $d(v_i, v_j) \leq D$, $\forall t > 0$, $\forall v_j \in N_i$.

C. Convergenza dell'algoritmo

Si dimostrerà che sotto certe ipotesi l'algoritmo garantisce la convergenza in un unico punto.

1) Indicheremo con v_i^M il/i veicolo/i entro la sfera di visibilità del veicolo v_i, a massima distanza $d_i^M(t)$ da v_i. Vale a dire $\forall v_i \in V$ e $\forall v_j \in N_i$:
   $$d_i^M(t) = d(v_i, v_i^M) \geq d(v_i, v_j) \tag{27}$$
   Indicheremo poi con $D^M(t)$ il massimo delle distanze $d_i^M(t)$:
   $$D^M(t) = \max_i \left(d_i^M(t) \right) \tag{28}$$
2) Se all'istante t_0 esiste nel grafo una formazione di m veicoli disposti secondo una *figura regolare* in cui il lato ha lunghezza pari a $D^M(t_0)$, allora per ogni $t \geq t_0$ la convergenza in un unico punto non viene raggiunta. In particolare la figura regolare risulta invariante nel tempo, mentre i veicoli che non ne fanno parte si concentrano in uno o più punti, mai coincidenti con i veicoli che formano la figura regolare, a seconda delle condizioni iniziali.
3) Si ha la converegnza in un unico punto se e solo se nessuna delle condizioni di cui al precedente punto *2* vale.

1) Dimostrazione del punto 3: Per dimostrare che si raggiunge la convergenza in un unico punto è sufficiente dimostrare che $D^M(t)$ converge a zero, vale a dire:

$$D^M(t) \longrightarrow 0 \tag{29}$$

La funzione $D^M(t)$ non è continua ma si dimostra che è monotona decrescente a tratti. Infatti, ogniqualvolta

un veicolo v_x entra nella sfera di visibilità di un altro veicolo v_y all'istante di tempo $\bar{t}$, è evidente che $d(v_x(\bar{t}), v_y(\bar{t})) = D$, perciò in $D^M(\bar{t})$ compare una discontinuità se $D^M(\bar{t} - \epsilon) < D$. Tuttavia il numero di discontinuità in $D^M(t)$ è limitato perchè per ipotesi la cardinalità N dell'insieme dei robot V è limitata, e per la dimostrazione del mantenimento della connessione, quando un veicolo entra nel raggio visivo di un altro veicolo vi rimane indefinitamente. Dunque per dimostrare la (29) è sufficiente dimostrare che nei tratti in cui è continua la funzione $D^M(t)$ è monotona decrescente.

Siano v_i e v_j due veicoli tali che:

$$d(v_i, v_j) = D^M \tag{30}$$

Ne segue che necessariamente:

$$\frac{d}{dt} d(v_i, v_j) \leq 0 \tag{31}$$

Se la disequazione in (31) è strettamente negativa per ogni coppia v_i, v_j che soddisfa la (30), allora la (29) è dimostrata, e l'algoritmo converge.

Se, viceversa, esiste almeno una coppia di veicoli, v_1 e v_m, tali per cui:

$$\frac{d}{dt} d(v_1, v_m) = 0 \tag{32}$$

significa che i veicoli v_1 e v_m sono necessariamente fermi[1]. Questo accade se v_1 vede un altro veicolo, diciamo v_2, alla distanza D^M, e se v_m vede un altro veicolo, diciamo v_{m-1}, alla distanza D^M.

Se almeno uno dei veicoli v_2 o v_{m-1}, diciamo v_2, vede un solo veicolo alla massima distanza D^M, vale a dire v_1, allora la condizione (32) è solo transitoria, dal momento che il veicolo v_2 si avvicinerebbe al veicolo v_1, sboccandolo.

Questa situazione non si verifica solo se entrambi i veicoli v_2 e v_{m-1} vedono un altro veicolo, diciamo v_3 e v_{m-2} rispettivamente , alla distanza D^M, rimanendo quindi fermi. Si capisce che anche in questo caso si può ripetere il ragionamento condotto al paragrafo precedente.

Ne segue che i veicoli v_1 e v_m rimangono indefinitivamente alla distanza reciproca D^M solo se esiste un circuito chiuso di m veicoli a distanza reciproca D^M, cioè se nel grafo esiste una formazione di m veicoli disposti secondo una figura regolare in cui il lato ha lunghezza pari a D^M.

[1]Infatti i veicoli v_1 e v_m non possono mantenere la distanza reciproca D^M se sono in movimento. Se questo accade significa che i veicoli in questione sono attirati da altri veicoli entro la loro sfera di visibilità e che stanno a distanza maggiore di D^M. Ma questo è impossibile per la definizione di D^M.

Siccome per ipotesi del punto *3* questa situazione non si verifica, allora la (29) è verificata e l'algoritmo converge in un unico punto.

Viceversa, se tutti i veicoli convergono in un unico punto, allora nel grafo non sono presenti figure regolari di m lati con lunghezza del lato pari a D^M.

2) Dimostrazione del punto 2: L'invarianza di eventuali figure regolari di m veicoli con lato di lunghezza pari a D^M è già stata dimostrata precedentemente.

Si dimostra ora che la lunghezza ℓ del lato deve essere necessariamente pari a D^M e non inferiore. Infatti se ciò non fosse, ad un certo punto almeno uno dei veicoli che compongono la figura regolare, diciamo v_f, vedrebbe nel suo campo visivo un veicolo v_r che non fa parte della figura regolare a distanza $d(v_f, v_r) > \ell$ e ne sarebbe attratto, rompendo così la figura.

I veicoli che non fanno parte della figura regolare non sono vincolati a rimanere fermi. Dalle simulzioni svolte al calcolatore si osserva che convergono verso uno o più punti, a seconda delle condizioni iniziali.

APPENDIX III
CODICE MATLAB

A. Funzioni comuni agli algoritmi di rendez-vous

Vengono qui riportati i listati delle funzioni comuni a tutti gli algoritmi di rendez-vous.

1) `crea_i_veicoli.m`: questa funzione crea N veicoli distribuiti casualmente sul piano (x, y). I veicoli non si sovrappongono e disegnano un grafo connesso.

Ciascun veicolo è rappresentato come una struttura contenente diversi campi, che regolano il funzionamento degli algortimi.

```
% CREA_I_VEICOLI
% Questa funzione crea N veicoli in posizioni
% aleatorie. I veicoli non si sovrappongono
% e disegnano un grafo connesso.
%
% Massimo Bortolato, Matteo Bustreo, Nicola Natali


function V = crea_i_veicoli(N, d, diam, delta)


q(1)            = rand + i*rand;

for k = 1 : N

    V(k).x      = real(q(k));       % robot x-pos
    V(k).y      = imag(q(k));       % robot y-pos
    V(k).theta  = 2*pi*rand;        % angolo di sterzo
    V(k).r      = 0;                % distanza veicolo (ins. circolare e anolonomi)
    V(k).diam   = diam;             % diametro di ingombro del robot
    V(k).d      = d;                % raggio di visibilita del robot
    V(k).delta  = delta;            % max movimento del robot
    V(k).el     = [];               % robot associato al robot
    V(k).dist   = [];               % distanza degli altri robot dal robot
    V(k).alfa   = [];               % angolatura dei robot relativa al robot
    V(k).invista= [];               % robot visibili dal robot
    V(k).direz  = 0;                % movimento intrapreso
    V(k).fasi   = 0;                % no. di cicli compiuti dal robot

    OK          = false;
    while OK == false
        r           = max(diam, d * rand);
        a           = 2 * pi * rand;
        tmp         = q(k) + r * exp(i*a);
        dist2       = (q - tmp) .* conj(q - tmp);
        dist2       = dist2 < (diam^2);
        if sum(dist2) == 0
            OK      = true;
        end
    end
    q(k+1)      = tmp;

end
```

2) `disegna_i_veicoli.m`: questa funzione disegna i veicoli nelle proprie posizioni. I veicoli possono essere rappresentati sia in forma circolare, sia triangolare per gli algoritmi che fanno uso di vincoli anolonomi.

```
function disegna_i_veicoli(V, colore, forma, scala)

if nargin < 3
    forma = 'circolare';
    scala = 1;
elseif nargin < 4
    scala = 1;
end
```

```
N       = size(V, 2);

if strcmp(forma, 'circolare')
ACCU    = 30;    % accuratezza nel disegnare il cerchio
hold on;
for k = 1 : N
        q       = V(k).x + i * V(k).y;
        r       = V(k).diam / 2 * scala;
        h       = q + r * exp(i*(0:pi/ACCU:2*pi));
        patch(real(h), imag(h), colore);
end
end

if strcmp(forma, 'triangolare')
    hold on;
    for k = 1 : N
        x = V(k).x;
        y = V(k).y;
        theta = V(k).theta;
        a = [-0.3536 -0.3536]';
b = [.5 0]';
c = [-0.3536 0.3536]';
P = scala*[a b c];
R = [ cos(theta) -sin(theta); sin(theta) cos(theta)];
Prot = R*P;
Prot_trasl = Prot + [ones(1,3)*x; ones(1,3)*y];
patch(Prot_trasl(1,:),Prot_trasl(2,:),colore);
    end
end
```

3) `disegna_grafo.m`*:* questa funzione disegna il grafo di visibilità dei veicoli.

```
% DISEGNA_GRAFO
% Questa funzione disegna il grafo
% di visibilità della rete di robot.
%
% Massimo Bortolato, Matteo Bustreo, Nicola Natali

function A = disegna_grafo(V)


N       = size(V, 2);
A       = false(N, N);

for k = 1 : N
    q(k)    = V(k).x + i * V(k).y;
end
for k = 1 : N
    dq      = q - q(k);
    dist    = sqrt(dq .* conj(dq));
    vedo    = dist <= V(k).d;
    vedo(k) = 0;
    A(:,k)  = vedo';
end
for k = 1 : N
    for h = k : N
        if A(k, h) == true
            line([V(k).x V(h).x], [V(k).y V(h).y]);
        end
    end
end
```

B. Rendez-Vous per una rete tempo-invariante di punti materiali

Questo algoritmo è costituito da un file: `main_algoritmo1.m`

```
% ALGORITMO 1
% Protocollo per la convergenza nel baricentro
% per una rete tempo-invariante di punti materiali
%
% Massimo Bortolato, Matteo Bustreo, Nicola Natali

clear all;
close all;
clc;

N       = 10;           % numero dei robot
d       = 50;           % raggio di visibilita dei robot
diam    = 1;            % diametro di ingombro dei robot
delta   = 1;            % max spostamento incrementale
dt      = .01;          % quanto temporale

% creo N veicoli in posizioni aleatorie. L'unico vincolo
% è che che il grafo sia connesso.
V       = crea_i_veicoli(N, d, diam, delta);

% adesso verranno calcolate le matrici
% di adiacenza e la matrice di Perron
% del grafo costituito dalla rete
% di robot mobili.
A       = zeros(N, N);      % matrice di adiacenza
L       = zeros(N, N);      % matrice laplaciana
D       = zeros(N, N);      % matrice dei gradi
for k = 1 : N
    q(k)    = V(k).x + i * V(k).y;
end
for k = 1 : N
    dq      = q - q(k);
    dist    = sqrt(dq .* conj(dq));
    vedo    = dist <= V(k).d;
    A(:,k)  = vedo';

    D(k,k)  = sum(vedo);
end
L       = D - A;            % matrice laplaciana
P       = eye(N) - dt * L;  % matrice di Perron
q       = q.';

figure; axis equal;
disegna_grafo(V);
disegna_i_veicoli(V, 'w');
ax      = axis;
pause;
clf;

for t = 1 : 10000

    % aggiornamento delle posizioni
    q      = P * q;
for k = 1 : N
        V(k).x  = real(q(k));
        V(k).y  = imag(q(k));
end

    % disegno dei grafici
    if t == 10 | t == 40 | t == 150
        clf;
        disegna_i_veicoli(V, 'w');
        axis(ax);
```

```
        drawnow;
        pause;
    end

end

clf;
disegna_i_veicoli(V, 'w');
axis(ax);
```

C. Rendez-Vous per robot puntiformi con visibilità limitata

Questo algoritmo è costituito da due file: `algoritmo_2.m` che implementa passo-passo il protocollo (14), e `main_algoritmo2.m` che crea i veicoli e li disegna.

```
% ALGORITMO 2
% Protocollo per la convergenza in un unico punto
% per una rete tempo-variante di robot mobili
%
% Massimo Bortolato, Matteo Bustreo, Nicola Natali

clear all;
close all;
clc;


N       = 20;           % numero dei robot
d       = 50;           % raggio di visibilita dei robot
diam    = 1;            % diametro di ingombro dei robot
delta   = 1;            % max spostamento incrementale

V       = crea_i_veicoli(N, d, diam, delta);

figure; axis equal;
disegna_grafo(V);
disegna_i_veicoli(V, 'w');
ax      = axis;
pause;
clf;

for t = 1 : 500

    V = algoritmo_2(V);

    if t == 10 | t == 40 | t == 150
        clf;
        disegna_i_veicoli(V, 'w');
        axis(ax);
        drawnow;
        pause;
    end

end

clf;
disegna_i_veicoli(V, 'w');
axis(ax);

*****
```

```
function V = algoritmo_2(V, option, value, value2)

if nargin < 2
    option = 'none';
    value  = 0;
    value2 = 0;
end

N   = size(V, 2);

% si calcolano le distanze relative tra
% un robot e tutti gli altri, e si individua
% il robot a massima distanza
for k = 1 : N
    q(k,1)    = V(k).x + i * V(k).y;
end
for k = 1 : N
    dq      = q - q(k);
    dist    = sqrt(dq .* conj(dq));
    alfa    = angle(dq);
    vedo    = dist <= V(k).d;
    vedo(k) = 0;
    dist    = dist .* vedo;
    if sum(vedo) == 0
        el  = [];
    else
        el  = find(dist == max(dist));
        if length(el) > 1
            el  = [];
        end
    end
    if strcmp(option, 'noise')
        e_a       = -value2 + 2*value2*rand(N,1);
        e_d       = -value + 2*value*rand(N,1);
        dist      = dist .* (1 + e_d/100);
        alfa      = alfa + e_a;
    end
    V(k).el       = el;
    V(k).dist     = dist;
    V(k).invista  = vedo;
    V(k).alfa     = alfa;
end

% si calcola la direzione in cui si deve muovere
% ogni robot, e l'intensità del movimento
W   = V;
for k = 1 : N

    if strcmp(option, 'synchrony') & rand > (value/100)
        W(k).direz = 0;
        continue;
    end

    if length(V(k).el) > 0
        if V(k).dist(V(k).el) > V(k).delta
            direz       = V(k).delta * exp(i*V(k).alfa(V(k).el));
        else
            direz       = V(k).dist(V(k).el) * exp(i*V(k).alfa(V(k).el));
        end
        W(k).direz  = direz;
        W(k).x    = V(k).x + real(W(k).direz);
        W(k).y    = V(k).y + imag(W(k).direz);
        W(k).fasi = V(k).fasi + 1;
    end

end
```

```
V   = W;
```

D. Rendez-Vous con veicoli anolonomi

```
% ALGORITMO 3
% Protocollo per la convergenza in un unico punto
% per una rete tempo-variante di robot mobili
%
% Massimo Bortolato, Matteo Bustreo, Nicola Natali

clear all;
close all;
clc;


N       = 20;           % numero dei robot
d       = 25;           % raggio di visibilita dei robot
diam    = 1;            % diametro di ingombro dei robot
delta   = 1;            % max spostamento incrementale
dt      = .01;

V       = crea_i_veicoli(N, d, diam, delta);

figure; axis equal;
disegna_i_veicoli(V, 'r', 'triangolare');
ax      = axis;
pause;
clf;

for t = 1 : 10000

    V = algoritmo_3(V, dt);

    if mod(t, 2) == 0
        clf;
        disegna_i_veicoli(V, 'r', 'triangolare');
        axis(ax);
        drawnow;
    end

end


*****




function V = algoritmo_3(V, dt)

N       = size(V, 2);

kr      = .5;%pi/(2*N) * csc(pi/N);
ka      = 1;

for k = 1 : N
    x(k) = V(k).x;
    y(k) = V(k).y;
end
for k = 1 : N
    dx = x - x(k);
    dy = y - y(k);
    dist2 = dx.^2 + dy.^2;
```

```
    vedo = dist2 <= (V(k).d)^2;
    vedo(k) = 0;
    if sum(vedo) == 0
        V(k).el = [];
    else
        dist2(~vedo) = 0;
        el = find(dist2 == max(dist2));
        if length(el) > 1
         el = [];
        end
        V(k).el = el;
    end
end

for k = 1 : N
    if length(V(k).el) == 0
        V(k).r          = 0;
        V(k).alfa       = 0;
    else
        dx          = V(V(k).el).x - V(k).x;
        dy          = V(V(k).el).y - V(k).y;
        V(k).r      = sqrt(dx^2+dy^2);
        alfa        = atan2(dy, dx) - V(k).theta;
        alfa        = mod(alfa, 2*pi);
        if alfa > pi
            rem         = mod(alfa, pi);
            alfa        = -pi + rem;
        end
        V(k).alfa   = alfa;
    end
end

for k = 1 : N
    v        = kr * V(k).r;
    if (V(k).r > .95 * V(k).d) & (abs(V(k).alfa) > pi/3)
        v = 0;
    end
    w        = ka * V(k).alfa;
    V(k).x   = V(k).x + dt * (cos(V(k).theta) * v);
    V(k).y   = V(k).y + dt * (sin(V(k).theta) * v);
    V(k).theta  = V(k).theta + dt * w;
end
```

E. Inseguimento circolare di veicoli anolonomi

```
% ALGORITMO 4
% Protocollo per l'inseguimento circolare
%
% Massimo Bortolato, Matteo Bustreo, Nicola Natali

clear all;
close all;
clc;


N       = 10;            % numero dei robot
d       = 50;           % raggio di visibilita dei robot
diam    = 1;            % diametro di ingombro dei robot
delta   = 1;            % max spostamento incrementale
dt      = .01;          % quanto temporale

V       = crea_i_veicoli(N, d, diam, delta);

figure; axis equal;
```

```
disegna_i_veicoli(V, 'r', 'triangolare');
ax      = axis;
pause;
clf;

for t = 1 : 10000

    V = algoritmo_4(V, dt);

    if mod(t, 4) == 0
        clf;
        disegna_i_veicoli(V, 'r', 'triangolare');
        axis(ax);
        drawnow;
    end

end


*****



function V = algoritmo_4(V, dt)


N       = size(V, 2);

% assegnazione dell'ordine di inseguimento
for k = 1 : N-1
    V(k).el     = k+1;
end
V(N).el     = 1;

% definizione dei guadagni critici
kr      = pi/(2*N) * csc(pi/N);
ka      = 1;

% calcolo delle coordinate relative
for k = 1 : N
    dx          = V(V(k).el).x - V(k).x;
    dy          = V(V(k).el).y - V(k).y;
    V(k).r      = sqrt(dx^2+dy^2);
    alfa        = atan2(dy, dx) - V(k).theta;
    alfa        = mod(alfa, 2*pi);
    if alfa > pi
        rem         = mod(alfa, pi);
        alfa        = -pi + rem;
    end
    V(k).alfa   = alfa;
end

% aggiornamento delle posizioni
for k = 1 : N
    v        = kr * V(k).r;         % velocità tangenziale
    w        = ka * V(k).alfa;      % velocità angolare
    V(k).x   = V(k).x + dt * (cos(V(k).theta) * v);
    V(k).y   = V(k).y + dt * (sin(V(k).theta) * v);
    V(k).theta  = V(k).theta + dt * w;
end
```

F. File usati per derivare gli indici di prestazione

Vengono qui riportati i listati fei file MATLAB utilizzati per derivare gli indici delle prestazioni dell'algortimo (14) in caso di veicoli asincroni e con rumore di misura.

```
% ANALISI DI SINCRONISMO
% Protocollo per la convergenza in un unico punto
% per una rete tempo-variante di robot mobili.
% Analisi delle prestazioni con veicoli asincroni.
%
% Massimo Bortolato, Matteo Bustreo, Nicola Natali

clear all;
close all;
clc;


N       = 10;           % numero dei robot
d       = 50;           % raggio di visibilita dei robot
diam    = 1;            % diametro di ingombro dei robot
delta   = 1;            % max spostamento incrementale

nomefile= 'analisi.txt';
option  = 'synchrony';
fid     = fopen(nomefile, 'a');

W       = crea_i_veicoli(N, d, diam, delta);

figure; axis equal;
disegna_grafo(W);
disegna_i_veicoli(W, 'w');
pause;


value   = 100;
V       = W;
OK      = true;
while OK
    V   = algoritmo_2(V);
    OK  = assieme(V);
end
for k = 1 : N
    f(k) = V(k).fasi;
end
F       = sum(f);
fprintf(fid, 'Opzione: %s %.1f\tVeicoli: %i\tFasi ...
tot: %i\n', option, value, N, F);
fprintf(fid, 'Media fasi per veicolo: %.2f\n\n', F/N);

clear F, f, V;
for value = [80 60 40 20]
    for u = 1 : 10
V       = W;
OK      = true;
while OK
            V   = algoritmo_2(V, option, 0, value);
            OK  = assieme(V);
end
for k = 1 : N
            f(k) = V(k).fasi;
end
F(u)    = sum(f);
fprintf(fid, 'Opzione: %s %.1f\tVeicoli: ...
            %i\tFasi tot: %i\n', option, value, N, F(u));
    end
    fprintf(fid, 'Media fasi per veicolo: %.2f\n\n', mean(F)/N);
    clear F, f, V;
```

```
end

fclose(fid);



****


% ANALISI CON RUMORE
% Protocollo per la convergenza in un unico punto
% per una rete tempo-variante di robot mobili.
% Analisi delle prestazioni con errori di misura.
%
% Massimo Bortolato, Matteo Bustreo, Nicola Natali

clear all;
close all;
clc;


N       = 50;           % numero dei robot
d       = 50;           % raggio di visibilita dei robot
diam    = 1;            % diametro di ingombro dei robot
delta   = 1;            % max spostamento incrementale

nomefile= 'analisi.txt';
option  = 'noise';
fid     = fopen(nomefile, 'a');

W       = crea_i_veicoli(N, d, diam, delta);

figure; axis equal;
disegna_grafo(W);
disegna_i_veicoli(W, 'w');
pause;

value   = 0;
value2  = 0;
V       = W;
OK      = true;
while OK
    V   = algoritmo_2(V);
    OK  = assieme(V);
end
for k = 1 : N
    f(k) = V(k).fasi;
end
F       = sum(f);
fprintf(fid, 'Opzione: %s %.1f %.1f\tVeicoli: %i\t...
Fasi tot: %i\n', option, value, value2*180/pi, N, F);
fprintf(fid, 'Media fasi per veicolo: %.2f\n\n', F/N);

clear F, f, V;
value2  = 0;
for value = [5 10 30 50]
    value2 = value * pi/180;
    for u = 1 : 10
V       = W;
OK      = true;
while OK
            V   = algoritmo_2(V, option, value, value2);
            OK  = assieme(V);
end
for k = 1 : N
            f(k) = V(k).fasi;
```

```
end
F(u)    = sum(f);
fprintf(fid, 'Opzione: %s %.1f %.1f\t...
Veicoli: %i\tFasi tot: %i\n', option, value, value2*180/pi, N, F(u));
    end
    fprintf(fid, 'Media fasi per veicolo: %.2f\n\n', mean(F)/N);
    clear F, f, V;
end

fclose(fid);
```

G. *Flocking*

```
T = 160;
dt = 1/15;
scale = 0.3;
N = 10;

rateV1 = 500;
distMin = 10;
rateV3 = 40;
velMax = 8;

place = 10^2 - 10^2*i;

v1 = zeros(N,1);
v2 = zeros(N,1);
v3 = zeros(N,1);

%% INIZIALIZZAZIONE %%

% generazione casuale delle condizioni iniziali
x = 10*randn(N,1) + 10i*randn(N,1);
v = 1*randn(N,1) + 1i*randn(N,1);

px = real(x);
py = imag(x);
vx = real(v);
vy = imag(v);
vel = atan2(vy, vx);


%% EVOLUZIONE %%

for t = 1:T

for j = 1 : N
plot_car(px(j), py(j), vel(j), scale);
end

centro = sum(x)/N;
vMedia = sum(v)/N;

for k = 1:N
% avvicinamento al centro di massa
v1(k) = (centro - x(k))/rateV1;

% mantenimento distanza minima
v2(k) = 0;
```

```
for w = 1:N
if abs(x(k)-x(w))<distMin
    v2(k) = v2(k) - abs(x(k)-x(w));
end
end

% assestamento velocità
v3(k) = (vMedia - v(k))/rateV3;

v4(k) = 0;
% avvicinamento ad un prefissato punto
% v4(k) = (place - x(k))/100;

% aggiornamento velocità
v(k) = v(k)+v1(k)+v2(k)+v3(k)+v4(k);

% limitazione della velocità
if v(k)>velMax
    v(k) = v(k)/abs(v(k))*velMax;
end

% aggiornamento posizione
x(k) = x(k)+v(k);
end

px = real(x);
py = imag(x);
vx = real(v);
vy = imag(v);
vel = atan2(vy, vx);

drawnow;
pause(0.05);
end;
```

H. Exploration

1) seguiMuri.m: È il file principale della simulazione. Vengono utilizzate le istanze già descritte nelle altre simulazioni per semplicità e coerenza, anche se in questo esempio si prende in considerazione solamente la posizione del veicolo mentre gli altri parametri vengono lasciati a zero.

```
T           = 1000000;
N           = 1;
ray         = 1;
scale       = .05;

V.d      = ray;
V.th     = 0;
a        = 0;
q        = 0;
V.x      = 16;
V.y      = 17;
V.r      = 0;
V.alfa   = 0;
```

Vengono ora definiti il passo di spostamento e la zona da esplorare. Si è scelto di rappresenzare la mappa come un poligono, per cui è stato sufficiente descriverne i vertici. Il codice non è robusto al cambiamento di questi vertici, perchè la definizione dei 'muri' non è automatica ma è stata descritta in modo sequenziale.

```
spost = 1;

limiti = [-10-10i -10i -40i 5-40i 5-10i 20-10i
20+20i 15+20i 15+10i 10i 0 -10 -10-10i];
mappa = 0;
```

Viene inizializzata la zona da esplorare nel grafico.

```
figure;
disegna_i_veicoli(V, scale);
plot(limiti);
axis([-15 25 -45 25]);
grid on;
pause;
```

La prima parte dell'algoritmo vero e proprio si occupa della gestione degli angoli. Si prende in considerazione caso per caso e viene imposto al robot uno spostamento verso l'interno della mappa pari allo spostamento base più una quantità aleatoria. Inoltre quando viene 'visto' un angolo, questo viene segnato nella mappa: questa altro non è se non un vettore in cui vengono inseriti tutti i punti riconosciuti come muri.

```
for t = 1:T
  if V.x<-10+ray && V.y<-10+ray
    mappa(length(mappa)+1)=V.x-ray + (V.y-ray)*i;
    V.y = V.y + spost+rand;
    V.x = V.x + spost+rand;
  elseif V.x<-10+ray && V.y>0-ray
    mappa(length(mappa)+1)=V.x-ray + (V.y+ray)*i;
    V.x = V.x + spost+rand;
    V.y = V.y - spost-rand;
  elseif V.x<0+ray && V.y>10-ray
    mappa(length(mappa)+1)=V.x-ray + (V.y+ray)*i;
    V.x = V.x + spost+rand;
    V.y = V.y - spost-rand;
  elseif V.x<15+ray && V.y>20-ray
    mappa(length(mappa)+1)=V.x-ray + (V.y+ray)*i;
    V.x = V.x + spost+rand;
    V.y = V.y - spost-rand;
  elseif V.x>20-ray && V.y>20-ray
    mappa(length(mappa)+1)=V.x+ray + (V.y+ray)*i;
    V.x = V.x - spost-rand;
    V.y = V.y - spost+rand;
  elseif V.x>20-ray && V.y<-10+ray
    mappa(length(mappa)+1)=V.x+ray + (V.y-ray)*i;
    V.x = V.x - spost-+rand;
    V.y = V.y + spost+rand;
  elseif V.x>5-ray && V.y<-40+ray
    mappa(length(mappa)+1)=V.x+ray + (V.y-ray)*i;
    V.x = V.x - spost-rand;
    V.y = V.y + spost+rand;
  elseif V.x<0+ray && V.y<-40+ray
    mappa(length(mappa)+1)=V.x-ray + (V.y-ray)*i;
    V.x = V.x + spost+rand;
    V.y = V.y + spost+rand;
```

Vengono ora prese in considerazione le altre condizioni. Sempre per semplicità di realizzazione la zona da esplorare, a partire dai vertici, viene descritta attraverso iperpiani. Come si può notare, un'ultieriore semplificazione è la presenza di muri esclusivamente paralleli all'asse delle ascisse o a quello delle ordinate. Anche questa caratteristica, comunque, toglie ben poco alla generalità del problema semplificando di gran lunga però il codice.

In questa sezione dell'algoritmo, si considera la posizione del robot e, se si riscontra la vicinanza ad un muro, si lancia un subroutine specificando la posizione del veicolo e la posizone del muro rispetto al veicolo in senso cartesiano.

Se invece il robot non è in prossimità di muri, si lancia un altra subroutine.

```
  else

    if V.x<-10+ray
      [V.x V.y mappa] = muro(V,1,spost,mappa,ray);
    elseif V.x>20-ray
      [V.x V.y mappa] = muro(V,2,spost,mappa,ray);
    elseif V.x<ray && V.y<-10
      [V.x V.y mappa] = muro(V,1,spost,mappa,ray);
    elseif V.x<ray && V.y>0
      [V.x V.y mappa] = muro(V,1,spost,mappa,ray);
    elseif V.x>5-ray && V.y<-10
      [V.x V.y mappa] = muro(V,2,spost,mappa,ray);
    elseif V.x<15+ray && V.y>15
      [V.x V.y mappa] = muro(V,1,spost,mappa,ray);
    else
      V.x = noMuroX(V,spost);
    end

    if V.y<-40+ray
      [V.x V.y mappa] = muro(V,3,spost,mappa,ray);
    elseif V.y>20-ray
      [V.x V.y mappa] = muro(V,4,spost,mappa,ray);
    elseif V.y>10-ray && V.x<15
      [V.x V.y mappa] = muro(V,4,spost,mappa,ray);
    elseif V.y>0-ray && V.x<0
      [V.x V.y mappa] = muro(V,4,spost,mappa,ray);
    elseif V.y<-10+ray && V.x<0
      [V.x V.y mappa] = muro(V,3,spost,mappa,ray);
    elseif V.y<-10+ray && V.x>5
      [V.x V.y mappa] = muro(V,3,spost,mappa,ray);
    else
      V.y = noMuroY(V,spost);
    end

  end
```

Questa parte dell'algoritmo disegna la mappa, passo per passo.

```
  disegna_i_veicoli(V, scale);
  plot_ellipse(V.x,V.y, 2*ray,'b');

  if length(mappa)>1
    for i = 2:length(mappa)
      plot(mappa(i),'o');
    end
  end

  drawnow;
  clf;
  axis([-15 25 -45 25]);
  grid on;
end
```

2) muro.m: Questa function gestisce il comportamento del robot in presenza di muro visibile. A secondo di dove si trova il muro, viene aggiornata la mappa e imposto al veicolo di muoversi nella direzione del muro.

Si è scelto di imporre un verso di percorrenza orario, ma era indifferente la scelta opposta, realizzabile semplicemente invertendo i segni di spostamento.

```
function [X Y mappa] = muro(V,a,spost,mappa,ray)

temp = randn>0;
if a == 1
    Y = V.y + spost;
    X =V.x;
    mappa(length(mappa)+1)=V.x-ray + V.y*i;
end
if a == 2
    Y = V.y - spost;
    X =V.x;
    mappa(length(mappa)+1)=V.x+ray + V.y*i;
end
if a == 3
    X = V.x - spost;
    Y =V.y;
    mappa(length(mappa)+1)=V.x + (V.y-ray)*i;
end
if a == 4
    X = V.x + spost;
    Y =V.y;
    mappa(length(mappa)+1)=V.x + (V.y+ray)*i;
end
```

3) noMuroX.m: Questa function (e in modo completamente simmetrico anche noMuroY.m) gestisce il caso in cui non ci sia nel raggio di visibilità del robot un muro. Viene semplicemente imposta uno spostamento aleatorio di media nulla.

```
function X = noMuroX(V,spost)

sp = 2*randn;
if sp > spost
    sp = spost;
elseif sp < -spost
    sp = -spost;
end
X = V.x + sp;
```

4) noMuroY.m:

```
function Y = noMuroY(V,spost)

sp = 2*randn;
if sp > spost
    sp = spost;
elseif sp < -spost
    sp = -spost;
end
Y = V.y + sp;
```

APPENDIX IV
LISTATI DELLE SIMULAZIONI DI PRESTAZIONE DEL PROTOCOLLO (14) INTRODOTTO IN SEZIONE III-B

.

A. Simulazioni con robot asincroni

```
Opzione: synchrony 100.0 Veicoli: 10 Fasi tot: 630
Media fasi per veicolo: 63.00

Opzione: synchrony 80.0 Veicoli: 10 Fasi tot: 626
Opzione: synchrony 80.0 Veicoli: 10 Fasi tot: 624
Opzione: synchrony 80.0 Veicoli: 10 Fasi tot: 645
Opzione: synchrony 80.0 Veicoli: 10 Fasi tot: 653
Opzione: synchrony 80.0 Veicoli: 10 Fasi tot: 634
Opzione: synchrony 80.0 Veicoli: 10 Fasi tot: 644
Opzione: synchrony 80.0 Veicoli: 10 Fasi tot: 641
Opzione: synchrony 80.0 Veicoli: 10 Fasi tot: 641
Opzione: synchrony 80.0 Veicoli: 10 Fasi tot: 672
Opzione: synchrony 80.0 Veicoli: 10 Fasi tot: 631
Media fasi per veicolo: 64.11

Opzione: synchrony 60.0 Veicoli: 10 Fasi tot: 652
Opzione: synchrony 60.0 Veicoli: 10 Fasi tot: 646
Opzione: synchrony 60.0 Veicoli: 10 Fasi tot: 652
Opzione: synchrony 60.0 Veicoli: 10 Fasi tot: 674
Opzione: synchrony 60.0 Veicoli: 10 Fasi tot: 654
Opzione: synchrony 60.0 Veicoli: 10 Fasi tot: 618
Opzione: synchrony 60.0 Veicoli: 10 Fasi tot: 679
Opzione: synchrony 60.0 Veicoli: 10 Fasi tot: 690
Opzione: synchrony 60.0 Veicoli: 10 Fasi tot: 663
Opzione: synchrony 60.0 Veicoli: 10 Fasi tot: 644
Media fasi per veicolo: 65.72

Opzione: synchrony 40.0 Veicoli: 10 Fasi tot: 663
Opzione: synchrony 40.0 Veicoli: 10 Fasi tot: 603
Opzione: synchrony 40.0 Veicoli: 10 Fasi tot: 677
Opzione: synchrony 40.0 Veicoli: 10 Fasi tot: 716
Opzione: synchrony 40.0 Veicoli: 10 Fasi tot: 676
Opzione: synchrony 40.0 Veicoli: 10 Fasi tot: 577
Opzione: synchrony 40.0 Veicoli: 10 Fasi tot: 604
Opzione: synchrony 40.0 Veicoli: 10 Fasi tot: 689
Opzione: synchrony 40.0 Veicoli: 10 Fasi tot: 689
Opzione: synchrony 40.0 Veicoli: 10 Fasi tot: 619
Media fasi per veicolo: 65.13

Opzione: synchrony 20.0 Veicoli: 10 Fasi tot: 704
Opzione: synchrony 20.0 Veicoli: 10 Fasi tot: 718
Opzione: synchrony 20.0 Veicoli: 10 Fasi tot: 656
Opzione: synchrony 20.0 Veicoli: 10 Fasi tot: 646
Opzione: synchrony 20.0 Veicoli: 10 Fasi tot: 656
Opzione: synchrony 20.0 Veicoli: 10 Fasi tot: 668
Opzione: synchrony 20.0 Veicoli: 10 Fasi tot: 634
Opzione: synchrony 20.0 Veicoli: 10 Fasi tot: 700
Opzione: synchrony 20.0 Veicoli: 10 Fasi tot: 650
Opzione: synchrony 20.0 Veicoli: 10 Fasi tot: 669
Media fasi per veicolo: 67.01
```

```
Opzione: synchrony 100.0 Veicoli: 50 Fasi tot: 8300
Media fasi per veicolo: 166.00

Opzione: synchrony 80.0 Veicoli: 50 Fasi tot: 8651
Opzione: synchrony 80.0 Veicoli: 50 Fasi tot: 8368
Opzione: synchrony 80.0 Veicoli: 50 Fasi tot: 8468
Opzione: synchrony 80.0 Veicoli: 50 Fasi tot: 8097
Opzione: synchrony 80.0 Veicoli: 50 Fasi tot: 8500
Opzione: synchrony 80.0 Veicoli: 50 Fasi tot: 8431
Opzione: synchrony 80.0 Veicoli: 50 Fasi tot: 8302
Opzione: synchrony 80.0 Veicoli: 50 Fasi tot: 8512
Opzione: synchrony 80.0 Veicoli: 50 Fasi tot: 8481
Opzione: synchrony 80.0 Veicoli: 50 Fasi tot: 8941
Media fasi per veicolo: 169.50

Opzione: synchrony 60.0 Veicoli: 50 Fasi tot: 8474
Opzione: synchrony 60.0 Veicoli: 50 Fasi tot: 8325
Opzione: synchrony 60.0 Veicoli: 50 Fasi tot: 8476
Opzione: synchrony 60.0 Veicoli: 50 Fasi tot: 8332
Opzione: synchrony 60.0 Veicoli: 50 Fasi tot: 8478
Opzione: synchrony 60.0 Veicoli: 50 Fasi tot: 8239
Opzione: synchrony 60.0 Veicoli: 50 Fasi tot: 8414
Opzione: synchrony 60.0 Veicoli: 50 Fasi tot: 8469
Opzione: synchrony 60.0 Veicoli: 50 Fasi tot: 8361
Opzione: synchrony 60.0 Veicoli: 50 Fasi tot: 8390
Media fasi per veicolo: 167.92

Opzione: synchrony 40.0 Veicoli: 50 Fasi tot: 8664
Opzione: synchrony 40.0 Veicoli: 50 Fasi tot: 8749
Opzione: synchrony 40.0 Veicoli: 50 Fasi tot: 8762
Opzione: synchrony 40.0 Veicoli: 50 Fasi tot: 8494
Opzione: synchrony 40.0 Veicoli: 50 Fasi tot: 8768
Opzione: synchrony 40.0 Veicoli: 50 Fasi tot: 8477
Opzione: synchrony 40.0 Veicoli: 50 Fasi tot: 8529
Opzione: synchrony 40.0 Veicoli: 50 Fasi tot: 8685
Opzione: synchrony 40.0 Veicoli: 50 Fasi tot: 8628
Opzione: synchrony 40.0 Veicoli: 50 Fasi tot: 8719
Media fasi per veicolo: 172.95

Opzione: synchrony 20.0 Veicoli: 50 Fasi tot: 8633
Opzione: synchrony 20.0 Veicoli: 50 Fasi tot: 8970
Opzione: synchrony 20.0 Veicoli: 50 Fasi tot: 8463
Opzione: synchrony 20.0 Veicoli: 50 Fasi tot: 8611
Opzione: synchrony 20.0 Veicoli: 50 Fasi tot: 8247
Opzione: synchrony 20.0 Veicoli: 50 Fasi tot: 9012
Opzione: synchrony 20.0 Veicoli: 50 Fasi tot: 8698
Opzione: synchrony 20.0 Veicoli: 50 Fasi tot: 8700
Opzione: synchrony 20.0 Veicoli: 50 Fasi tot: 8161
Opzione: synchrony 20.0 Veicoli: 50 Fasi tot: 8522
Media fasi per veicolo: 172.03
```

```
Opzione: synchrony 100.0 Veicoli: 100 Fasi tot: 13300
Media fasi per veicolo: 133.00

Opzione: synchrony 80.0 Veicoli: 100 Fasi tot: 13247
Opzione: synchrony 80.0 Veicoli: 100 Fasi tot: 13843
Opzione: synchrony 80.0 Veicoli: 100 Fasi tot: 13697
Opzione: synchrony 80.0 Veicoli: 100 Fasi tot: 13557
Opzione: synchrony 80.0 Veicoli: 100 Fasi tot: 13634
Opzione: synchrony 80.0 Veicoli: 100 Fasi tot: 13071
Opzione: synchrony 80.0 Veicoli: 100 Fasi tot: 14204
Opzione: synchrony 80.0 Veicoli: 100 Fasi tot: 13567
Opzione: synchrony 80.0 Veicoli: 100 Fasi tot: 13251
Opzione: synchrony 80.0 Veicoli: 100 Fasi tot: 13495
Media fasi per veicolo: 135.57

Opzione: synchrony 60.0 Veicoli: 100 Fasi tot: 13435
Opzione: synchrony 60.0 Veicoli: 100 Fasi tot: 13822
Opzione: synchrony 60.0 Veicoli: 100 Fasi tot: 13527
Opzione: synchrony 60.0 Veicoli: 100 Fasi tot: 13607
Opzione: synchrony 60.0 Veicoli: 100 Fasi tot: 12932
Opzione: synchrony 60.0 Veicoli: 100 Fasi tot: 14233
Opzione: synchrony 60.0 Veicoli: 100 Fasi tot: 13630
Opzione: synchrony 60.0 Veicoli: 100 Fasi tot: 13161
Opzione: synchrony 60.0 Veicoli: 100 Fasi tot: 13119
Opzione: synchrony 60.0 Veicoli: 100 Fasi tot: 13393
Media fasi per veicolo: 134.86

Opzione: synchrony 40.0 Veicoli: 100 Fasi tot: 13425
Opzione: synchrony 40.0 Veicoli: 100 Fasi tot: 13621
Opzione: synchrony 40.0 Veicoli: 100 Fasi tot: 13371
Opzione: synchrony 40.0 Veicoli: 100 Fasi tot: 13668
Opzione: synchrony 40.0 Veicoli: 100 Fasi tot: 13586
Opzione: synchrony 40.0 Veicoli: 100 Fasi tot: 13168
Opzione: synchrony 40.0 Veicoli: 100 Fasi tot: 14002
Opzione: synchrony 40.0 Veicoli: 100 Fasi tot: 13502
Opzione: synchrony 40.0 Veicoli: 100 Fasi tot: 13408
Opzione: synchrony 40.0 Veicoli: 100 Fasi tot: 13304
Media fasi per veicolo: 135.06

Opzione: synchrony 20.0 Veicoli: 100 Fasi tot: 13985
Opzione: synchrony 20.0 Veicoli: 100 Fasi tot: 14041
Opzione: synchrony 20.0 Veicoli: 100 Fasi tot: 13460
Opzione: synchrony 20.0 Veicoli: 100 Fasi tot: 13259
Opzione: synchrony 20.0 Veicoli: 100 Fasi tot: 13267
Opzione: synchrony 20.0 Veicoli: 100 Fasi tot: 13835
Opzione: synchrony 20.0 Veicoli: 100 Fasi tot: 13274
Opzione: synchrony 20.0 Veicoli: 100 Fasi tot: 14049
Opzione: synchrony 20.0 Veicoli: 100 Fasi tot: 14091
Opzione: synchrony 20.0 Veicoli: 100 Fasi tot: 13710
Media fasi per veicolo: 136.97
```

B. Simulazioni con errori di misura

Il primo parametro del campo `noise` è l'errore sulla distanza, il secondo l'errore sulla direzione.

```
Opzione: noise 0.0 0.0 Veicoli: 10 Fasi tot: 400
Media fasi per veicolo: 40.00

Opzione: noise 0.0 5.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 5.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 5.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 5.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 5.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 5.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 5.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 5.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 5.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 5.0 Veicoli: 10 Fasi tot: 400
Media fasi per veicolo: 40.00
```

```
Opzione: noise 0.0 10.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 10.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 10.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 10.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 10.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 10.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 10.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 10.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 10.0 Veicoli: 10 Fasi tot: 400
Opzione: noise 0.0 10.0 Veicoli: 10 Fasi tot: 400
Media fasi per veicolo: 40.00

Opzione: noise 0.0 30.0 Veicoli: 10 Fasi tot: 420
Opzione: noise 0.0 30.0 Veicoli: 10 Fasi tot: 410
Opzione: noise 0.0 30.0 Veicoli: 10 Fasi tot: 420
Opzione: noise 0.0 30.0 Veicoli: 10 Fasi tot: 420
Opzione: noise 0.0 30.0 Veicoli: 10 Fasi tot: 420
Opzione: noise 0.0 30.0 Veicoli: 10 Fasi tot: 420
Opzione: noise 0.0 30.0 Veicoli: 10 Fasi tot: 410
Opzione: noise 0.0 30.0 Veicoli: 10 Fasi tot: 410
Opzione: noise 0.0 30.0 Veicoli: 10 Fasi tot: 420
Opzione: noise 0.0 30.0 Veicoli: 10 Fasi tot: 420
Media fasi per veicolo: 41.70

Opzione: noise 0.0 50.0 Veicoli: 10 Fasi tot: 440
Opzione: noise 0.0 50.0 Veicoli: 10 Fasi tot: 490
Opzione: noise 0.0 50.0 Veicoli: 10 Fasi tot: 440
Opzione: noise 0.0 50.0 Veicoli: 10 Fasi tot: 440
Opzione: noise 0.0 50.0 Veicoli: 10 Fasi tot: 470
Opzione: noise 0.0 50.0 Veicoli: 10 Fasi tot: 460
Opzione: noise 0.0 50.0 Veicoli: 10 Fasi tot: 470
Opzione: noise 0.0 50.0 Veicoli: 10 Fasi tot: 460
Opzione: noise 0.0 50.0 Veicoli: 10 Fasi tot: 460
Opzione: noise 0.0 50.0 Veicoli: 10 Fasi tot: 460
Media fasi per veicolo: 45.90




Opzione: noise 0.0 0.0 Veicoli: 25 Fasi tot: 2825
Media fasi per veicolo: 113.00

Opzione: noise 0.0 5.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 5.0 Veicoli: 25 Fasi tot: 2825
Opzione: noise 0.0 5.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 5.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 5.0 Veicoli: 25 Fasi tot: 2825
Opzione: noise 0.0 5.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 5.0 Veicoli: 25 Fasi tot: 2825
Opzione: noise 0.0 5.0 Veicoli: 25 Fasi tot: 2825
Opzione: noise 0.0 5.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 5.0 Veicoli: 25 Fasi tot: 2825
Media fasi per veicolo: 113.50

Opzione: noise 0.0 10.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 10.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 10.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 10.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 10.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 10.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 10.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 10.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 10.0 Veicoli: 25 Fasi tot: 2850
Opzione: noise 0.0 10.0 Veicoli: 25 Fasi tot: 2850
Media fasi per veicolo: 114.00

Opzione: noise 0.0 30.0 Veicoli: 25 Fasi tot: 2975
Opzione: noise 0.0 30.0 Veicoli: 25 Fasi tot: 3000
Opzione: noise 0.0 30.0 Veicoli: 25 Fasi tot: 2975
Opzione: noise 0.0 30.0 Veicoli: 25 Fasi tot: 2975
Opzione: noise 0.0 30.0 Veicoli: 25 Fasi tot: 2975
Opzione: noise 0.0 30.0 Veicoli: 25 Fasi tot: 2975
Opzione: noise 0.0 30.0 Veicoli: 25 Fasi tot: 3025
Opzione: noise 0.0 30.0 Veicoli: 25 Fasi tot: 3000
Opzione: noise 0.0 30.0 Veicoli: 25 Fasi tot: 3000
Opzione: noise 0.0 30.0 Veicoli: 25 Fasi tot: 2975
Media fasi per veicolo: 119.50

Opzione: noise 0.0 50.0 Veicoli: 25 Fasi tot: 3225
Opzione: noise 0.0 50.0 Veicoli: 25 Fasi tot: 3200
Opzione: noise 0.0 50.0 Veicoli: 25 Fasi tot: 3200
Opzione: noise 0.0 50.0 Veicoli: 25 Fasi tot: 3225
Opzione: noise 0.0 50.0 Veicoli: 25 Fasi tot: 3275
Opzione: noise 0.0 50.0 Veicoli: 25 Fasi tot: 3275
Opzione: noise 0.0 50.0 Veicoli: 25 Fasi tot: 3225
Opzione: noise 0.0 50.0 Veicoli: 25 Fasi tot: 3225
Opzione: noise 0.0 50.0 Veicoli: 25 Fasi tot: 3250
Opzione: noise 0.0 50.0 Veicoli: 25 Fasi tot: 3275
Media fasi per veicolo: 129.50
```

```
Opzione: noise 0.0 0.0 Veicoli: 50 Fasi tot: 4450
Media fasi per veicolo: 89.00

Opzione: noise 0.0 5.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 5.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 5.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 5.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 5.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 5.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 5.0 Veicoli: 50 Fasi tot: 4400
Opzione: noise 0.0 5.0 Veicoli: 50 Fasi tot: 4400
Opzione: noise 0.0 5.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 5.0 Veicoli: 50 Fasi tot: 4450
Media fasi per veicolo: 88.80

Opzione: noise 0.0 10.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 10.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 10.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 10.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 10.0 Veicoli: 50 Fasi tot: 4350
Opzione: noise 0.0 10.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 10.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 10.0 Veicoli: 50 Fasi tot: 4450
Opzione: noise 0.0 10.0 Veicoli: 50 Fasi tot: 4400
Opzione: noise 0.0 10.0 Veicoli: 50 Fasi tot: 4400
Media fasi per veicolo: 88.60

Opzione: noise 0.0 30.0 Veicoli: 50 Fasi tot: 4650
Opzione: noise 0.0 30.0 Veicoli: 50 Fasi tot: 4650
Opzione: noise 0.0 30.0 Veicoli: 50 Fasi tot: 4600
Opzione: noise 0.0 30.0 Veicoli: 50 Fasi tot: 4550
Opzione: noise 0.0 30.0 Veicoli: 50 Fasi tot: 4550
Opzione: noise 0.0 30.0 Veicoli: 50 Fasi tot: 4700
Opzione: noise 0.0 30.0 Veicoli: 50 Fasi tot: 4650
Opzione: noise 0.0 30.0 Veicoli: 50 Fasi tot: 4600
Opzione: noise 0.0 30.0 Veicoli: 50 Fasi tot: 4600
Opzione: noise 0.0 30.0 Veicoli: 50 Fasi tot: 4700
Media fasi per veicolo: 92.50

Opzione: noise 0.0 50.0 Veicoli: 50 Fasi tot: 5200
Opzione: noise 0.0 50.0 Veicoli: 50 Fasi tot: 5050
Opzione: noise 0.0 50.0 Veicoli: 50 Fasi tot: 5200
Opzione: noise 0.0 50.0 Veicoli: 50 Fasi tot: 5200
Opzione: noise 0.0 50.0 Veicoli: 50 Fasi tot: 5100
Opzione: noise 0.0 50.0 Veicoli: 50 Fasi tot: 5100
Opzione: noise 0.0 50.0 Veicoli: 50 Fasi tot: 5200
Opzione: noise 0.0 50.0 Veicoli: 50 Fasi tot: 4900
Opzione: noise 0.0 50.0 Veicoli: 50 Fasi tot: 5050
Opzione: noise 0.0 50.0 Veicoli: 50 Fasi tot: 5200
Media fasi per veicolo: 102.40


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Opzione: noise 0.0 0.0 Veicoli: 10 Fasi tot: 370
Media fasi per veicolo: 37.00

Opzione: noise 5.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 5.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 5.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 5.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 5.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 5.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 5.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 5.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 5.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 5.0 0.0 Veicoli: 10 Fasi tot: 370
Media fasi per veicolo: 37.00

Opzione: noise 10.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 10.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 10.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 10.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 10.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 10.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 10.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 10.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 10.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 10.0 0.0 Veicoli: 10 Fasi tot: 370
Media fasi per veicolo: 37.00

Opzione: noise 30.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 30.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 30.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 30.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 30.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 30.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 30.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 30.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 30.0 0.0 Veicoli: 10 Fasi tot: 370
```

```
Opzione: noise 30.0 0.0 Veicoli: 10 Fasi tot: 370
Media fasi per veicolo: 37.00

Opzione: noise 50.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 50.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 50.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 50.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 50.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 50.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 50.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 50.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 50.0 0.0 Veicoli: 10 Fasi tot: 370
Opzione: noise 50.0 0.0 Veicoli: 10 Fasi tot: 370
Media fasi per veicolo: 37.00



Opzione: noise 0.0 0.0 Veicoli: 25 Fasi tot: 1800
Media fasi per veicolo: 72.00

Opzione: noise 5.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 5.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 5.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 5.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 5.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 5.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 5.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 5.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 5.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 5.0 0.0 Veicoli: 25 Fasi tot: 1800
Media fasi per veicolo: 72.00

Opzione: noise 10.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 10.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 10.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 10.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 10.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 10.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 10.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 10.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 10.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 10.0 0.0 Veicoli: 25 Fasi tot: 1800
Media fasi per veicolo: 72.00

Opzione: noise 30.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 30.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 30.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 30.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 30.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 30.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 30.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 30.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 30.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 30.0 0.0 Veicoli: 25 Fasi tot: 1800
Media fasi per veicolo: 72.00

Opzione: noise 50.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 50.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 50.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 50.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 50.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 50.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 50.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 50.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 50.0 0.0 Veicoli: 25 Fasi tot: 1800
Opzione: noise 50.0 0.0 Veicoli: 25 Fasi tot: 1800
Media fasi per veicolo: 72.00



Opzione: noise 0.0 0.0 Veicoli: 50 Fasi tot: 5800
Media fasi per veicolo: 116.00

Opzione: noise 5.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 5.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 5.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 5.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 5.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 5.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 5.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 5.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 5.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 5.0 0.0 Veicoli: 50 Fasi tot: 5800
Media fasi per veicolo: 116.00

Opzione: noise 10.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 10.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 10.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 10.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 10.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 10.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 10.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 10.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 10.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 10.0 0.0 Veicoli: 50 Fasi tot: 5800
Media fasi per veicolo: 116.00

Opzione: noise 30.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 30.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 30.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 30.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 30.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 30.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 30.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 30.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 30.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 30.0 0.0 Veicoli: 50 Fasi tot: 5800
Media fasi per veicolo: 116.00

Opzione: noise 50.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 50.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 50.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 50.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 50.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 50.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 50.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 50.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 50.0 0.0 Veicoli: 50 Fasi tot: 5800
Opzione: noise 50.0 0.0 Veicoli: 50 Fasi tot: 5800
Media fasi per veicolo: 116.00



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



Opzione: noise 0.0 0.0 Veicoli: 10 Fasi tot: 510
Media fasi per veicolo: 51.00

Opzione: noise 5.0 5.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 5.0 5.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 5.0 5.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 5.0 5.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 5.0 5.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 5.0 5.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 5.0 5.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 5.0 5.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 5.0 5.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 5.0 5.0 Veicoli: 10 Fasi tot: 510
Media fasi per veicolo: 51.00

Opzione: noise 10.0 10.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 10.0 10.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 10.0 10.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 10.0 10.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 10.0 10.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 10.0 10.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 10.0 10.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 10.0 10.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 10.0 10.0 Veicoli: 10 Fasi tot: 510
Opzione: noise 10.0 10.0 Veicoli: 10 Fasi tot: 510
Media fasi per veicolo: 51.00

Opzione: noise 30.0 30.0 Veicoli: 10 Fasi tot: 530
Opzione: noise 30.0 30.0 Veicoli: 10 Fasi tot: 540
Opzione: noise 30.0 30.0 Veicoli: 10 Fasi tot: 530
Opzione: noise 30.0 30.0 Veicoli: 10 Fasi tot: 530
Opzione: noise 30.0 30.0 Veicoli: 10 Fasi tot: 530
Opzione: noise 30.0 30.0 Veicoli: 10 Fasi tot: 540
Opzione: noise 30.0 30.0 Veicoli: 10 Fasi tot: 520
Opzione: noise 30.0 30.0 Veicoli: 10 Fasi tot: 530
Opzione: noise 30.0 30.0 Veicoli: 10 Fasi tot: 540
Opzione: noise 30.0 30.0 Veicoli: 10 Fasi tot: 530
Media fasi per veicolo: 53.20

Opzione: noise 50.0 50.0 Veicoli: 10 Fasi tot: 580
Opzione: noise 50.0 50.0 Veicoli: 10 Fasi tot: 580
Opzione: noise 50.0 50.0 Veicoli: 10 Fasi tot: 570
Opzione: noise 50.0 50.0 Veicoli: 10 Fasi tot: 570
Opzione: noise 50.0 50.0 Veicoli: 10 Fasi tot: 600
Opzione: noise 50.0 50.0 Veicoli: 10 Fasi tot: 590
Opzione: noise 50.0 50.0 Veicoli: 10 Fasi tot: 570
Opzione: noise 50.0 50.0 Veicoli: 10 Fasi tot: 600
Opzione: noise 50.0 50.0 Veicoli: 10 Fasi tot: 590
Opzione: noise 50.0 50.0 Veicoli: 10 Fasi tot: 570
Media fasi per veicolo: 58.20



Opzione: noise 0.0 0.0 Veicoli: 25 Fasi tot: 3575
Media fasi per veicolo: 143.00

Opzione: noise 5.0 5.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 5.0 5.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 5.0 5.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 5.0 5.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 5.0 5.0 Veicoli: 25 Fasi tot: 3575
```

```
Opzione: noise 5.0 5.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 5.0 5.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 5.0 5.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 5.0 5.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 5.0 5.0 Veicoli: 25 Fasi tot: 3575
Media fasi per veicolo: 143.00

Opzione: noise 10.0 10.0 Veicoli: 25 Fasi tot: 3600
Opzione: noise 10.0 10.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 10.0 10.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 10.0 10.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 10.0 10.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 10.0 10.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 10.0 10.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 10.0 10.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 10.0 10.0 Veicoli: 25 Fasi tot: 3575
Opzione: noise 10.0 10.0 Veicoli: 25 Fasi tot: 3600
Media fasi per veicolo: 143.20

Opzione: noise 30.0 30.0 Veicoli: 25 Fasi tot: 3750
Opzione: noise 30.0 30.0 Veicoli: 25 Fasi tot: 3700
Opzione: noise 30.0 30.0 Veicoli: 25 Fasi tot: 3725
Opzione: noise 30.0 30.0 Veicoli: 25 Fasi tot: 3825
Opzione: noise 30.0 30.0 Veicoli: 25 Fasi tot: 3725
Opzione: noise 30.0 30.0 Veicoli: 25 Fasi tot: 3725
Opzione: noise 30.0 30.0 Veicoli: 25 Fasi tot: 3800
Opzione: noise 30.0 30.0 Veicoli: 25 Fasi tot: 3800
Opzione: noise 30.0 30.0 Veicoli: 25 Fasi tot: 3750
Opzione: noise 30.0 30.0 Veicoli: 25 Fasi tot: 3725
Media fasi per veicolo: 150.10

Opzione: noise 50.0 50.0 Veicoli: 25 Fasi tot: 4025
Opzione: noise 50.0 50.0 Veicoli: 25 Fasi tot: 3975
Opzione: noise 50.0 50.0 Veicoli: 25 Fasi tot: 4175
Opzione: noise 50.0 50.0 Veicoli: 25 Fasi tot: 4050
Opzione: noise 50.0 50.0 Veicoli: 25 Fasi tot: 4175
Opzione: noise 50.0 50.0 Veicoli: 25 Fasi tot: 4150
Opzione: noise 50.0 50.0 Veicoli: 25 Fasi tot: 4075
Opzione: noise 50.0 50.0 Veicoli: 25 Fasi tot: 4075
Opzione: noise 50.0 50.0 Veicoli: 25 Fasi tot: 4025
Opzione: noise 50.0 50.0 Veicoli: 25 Fasi tot: 4100
Media fasi per veicolo: 163.30
```

```
Opzione: noise 0.0 0.0 Veicoli: 50 Fasi tot: 8400
Media fasi per veicolo: 168.00

Opzione: noise 5.0 5.0 Veicoli: 50 Fasi tot: 8400
Opzione: noise 5.0 5.0 Veicoli: 50 Fasi tot: 8550
Opzione: noise 5.0 5.0 Veicoli: 50 Fasi tot: 8400
Opzione: noise 5.0 5.0 Veicoli: 50 Fasi tot: 8500
Opzione: noise 5.0 5.0 Veicoli: 50 Fasi tot: 8500
Opzione: noise 5.0 5.0 Veicoli: 50 Fasi tot: 8550
Opzione: noise 5.0 5.0 Veicoli: 50 Fasi tot: 8500
Opzione: noise 5.0 5.0 Veicoli: 50 Fasi tot: 8550
Opzione: noise 5.0 5.0 Veicoli: 50 Fasi tot: 8450
Opzione: noise 5.0 5.0 Veicoli: 50 Fasi tot: 8600
Media fasi per veicolo: 170.00

Opzione: noise 10.0 10.0 Veicoli: 50 Fasi tot: 8550
Opzione: noise 10.0 10.0 Veicoli: 50 Fasi tot: 8550
Opzione: noise 10.0 10.0 Veicoli: 50 Fasi tot: 8400
Opzione: noise 10.0 10.0 Veicoli: 50 Fasi tot: 8450
Opzione: noise 10.0 10.0 Veicoli: 50 Fasi tot: 8350
Opzione: noise 10.0 10.0 Veicoli: 50 Fasi tot: 8450
Opzione: noise 10.0 10.0 Veicoli: 50 Fasi tot: 8550
Opzione: noise 10.0 10.0 Veicoli: 50 Fasi tot: 8550
Opzione: noise 10.0 10.0 Veicoli: 50 Fasi tot: 8450
Opzione: noise 10.0 10.0 Veicoli: 50 Fasi tot: 8550
Media fasi per veicolo: 169.70

Opzione: noise 30.0 30.0 Veicoli: 50 Fasi tot: 8800
Opzione: noise 30.0 30.0 Veicoli: 50 Fasi tot: 8750
Opzione: noise 30.0 30.0 Veicoli: 50 Fasi tot: 8900
Opzione: noise 30.0 30.0 Veicoli: 50 Fasi tot: 8900
Opzione: noise 30.0 30.0 Veicoli: 50 Fasi tot: 8850
Opzione: noise 30.0 30.0 Veicoli: 50 Fasi tot: 8950
Opzione: noise 30.0 30.0 Veicoli: 50 Fasi tot: 8950
Opzione: noise 30.0 30.0 Veicoli: 50 Fasi tot: 8800
Opzione: noise 30.0 30.0 Veicoli: 50 Fasi tot: 9000
Opzione: noise 30.0 30.0 Veicoli: 50 Fasi tot: 8750
Media fasi per veicolo: 177.30

Opzione: noise 50.0 50.0 Veicoli: 50 Fasi tot: 9700
Opzione: noise 50.0 50.0 Veicoli: 50 Fasi tot: 9600
Opzione: noise 50.0 50.0 Veicoli: 50 Fasi tot: 9600
Opzione: noise 50.0 50.0 Veicoli: 50 Fasi tot: 9450
Opzione: noise 50.0 50.0 Veicoli: 50 Fasi tot: 9600
Opzione: noise 50.0 50.0 Veicoli: 50 Fasi tot: 9450
Opzione: noise 50.0 50.0 Veicoli: 50 Fasi tot: 9700
Opzione: noise 50.0 50.0 Veicoli: 50 Fasi tot: 9550
Opzione: noise 50.0 50.0 Veicoli: 50 Fasi tot: 9500
Opzione: noise 50.0 50.0 Veicoli: 50 Fasi tot: 9650
Media fasi per veicolo: 191.60
```

CONTENTS

REFERENCES

[1] R. Olfati-Saber, J. A. Fax, R. M. Murray: *Consensus and cooperation in networked multi-agent systems*, 2006.

[2] H. Ando, Y. Oasa, I. Suzuki, M. Yamashita: *Distributed memoryless point convergence algorithm for mobile robots with limited visibility*, IEEE Trans. on Robotics and Automation, vol. 15, no. 5, ottobre 1999.

[3] C. Raynolds: *Flocks, Herds, and Schools: A distribuited Behavioral Model*, Computer Graphics, July 1987

[4] J. A. Marshall, M. E. Broucke, B. A. Francis: *Pursuit Formations of Unicycles*, Automatica, vol. 41, no. 12, December 2005

[5] J. A. Marshall, M. E. Broucke, B. A. Francis: *Formations of Vehicles in Cyclic Pursuit*, Automatic control, Vol. 49, no. 11, November 2004

[6] D. E. Chang, S. C. Shadden, J. E. Mardsen, R. Olfati-Saber: *Collision Avoidance for Multiple Agent Systems* , Proceedings of the 42nd IEEE Conference on Decision and Control, Maui, Hawaii USA, December 2003

[7] Batalin, M.A., Sukhatme, G.S.: *Spreading out: A local approach to multi-robot coverage*. Proc. of 6th International Symposium on Distributed Autonomous Robotic Systems, Fukuoka, Japan (2002) 373–382

[8] G. Fang, G. Dissanayake and H. Lau, *A Behaviour-Based Optimisation Strategy for Multi-Robot Exploration*. IEEE Conference on Robotics, Automation and Mechatronics (RAM'04), Singapore, December 2004

[9] K. Ito, *Simple Robots in a Complex World: Collaborative Exploration Behavior using Genetic Programming*. ???

[10] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, *Coordinated multi-robot exploration*. IEEE Transactions on Robotics, 2005.

[11] R. Simmons, D. Apfelbaum, D. Fox, R. Goldman, K. Zita Haigh, D. Musliner, M. Pelican, and S. Thrun, *Coordinated Deployment of Multiple, Heterogeneous Robots*. In Proceedings of the Conference on Intelligent Robots and Systems (IROS), Takamatsu Japan, October 2000

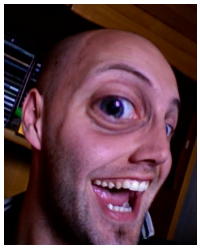
[12] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, *Coordination for multi-robot exploration and mapping*. In Proc. of the National Conference on Artificial Intelligence (AAAI), 2000, pp. 851–858

[13] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. *A practical, decision-theoretic approach to multi-robot mapping and exploration*. In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pages 3232–3238, Las Vegas, NV, USA, 2003

Massimo Bortolato won the bronze medal in the "Chiesetta" slalom ski race in Caviola di Falcade, Belluno, Italy, 1986.

He won the gold medal in the karate regional competition in Padova, Italy, 1989, and in 1992 he won the silver medal in the "Zero Branco" football tournament in Zero Branco, Treviso, Italy.

At present, he is engaged in several dangerous climbs all over the world, from Japan to Kenia.



Matteo Bustreo is a rockstar and a virtuous guitarist. In 2002 he won the Rugby Italian U21 Championship with Petrarca Padova, playing no match. He broke his nose 3 times, his arm once, and lost memory for a *ginocchiata* once. His yoga experiments makes his *pancia* to *crescere*.

Nicola Natali is a looser.

Sandra is the Muse that inspired the group. She is Massimo's girlfriend, but also everybody's friend!

During our rendez-vous experiments, her presence gave rise to some difficulties because all robots chose her as centroid.