UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria dell'Automazione

Corso di Progettazione di Sistemi di Controllo a.a. 2010/11
Prof. Luca Schenato

Progetto n.3: Analisi di algoritmi di autolocalizzazione per reti di sensori wireless

Cuccato Davide, De Franceschi Igor,
Fauri Davide, Sartor Giorgio

Padova, 25 Febbraio 2011

Indice

Abstract

Le WSN (*Wireless Sensor Network*)sono reti di comunicazione via radio formate da sensori wireless chiamati spesso nodi, o mote, e collocati a distanza variabile tra loro. Tali reti sono usualmente caratterizzate da: comunicazione non direzionabile deterministicamente; dispositivi di dimensioni contenute, ma con risorse di memoria e calcolo limitate; possibili interazioni con l'ambiente circostante; autonomia energetica medio-lunga; contesti operativi eterogenei ed ostili; funzionamento autonomo, non costantemente monitorato, e conseguentemente suscettibile a guasti. Reti siffatte trovano applicazione in vari ambiti tecnologici quali: monitoraggio dell'ambiente, logistico, della previdenza sociale, militare, culturale, turistico. Scopo del lavoro qui presentato é lo studio di modelli, algoritmi e tecniche implementative atti a descrivere le suddette reti in relazione a problemi di autolocalizzazione (supponendo nota la mappa in cui verranno posizionati i mote), comparando diversi approcci già presenti in letteratura. Tra le varie tecniche comunemente utilizzate a tal fine sono state privilegiate quelle basate sulla misura della potenza ricevuta (*Received Signal Strength, RSS*); a fronte dello studio teorico si presentano infine una serie di test sperimentali effettuati sia su una *tranche* di dati reali sia grazie ad ambiente di simulazione Matlab.
Tutto il codice MATLAB prodotto é ampiamente documentato, di facile conversione ad un'applicazione reale e liberamente disponibile su `http://www.mediafire.com/fauri`.

1 Introduzione all'autolocalizzazione

Nelle applicazioni delle reti di sensori *wireless* sopracitate un gran numero di nodi viene posizionato nell'ambiente in maniera casuale e si può assumere in pratica una totale incorrelazione tra ID del nodo e posizione assoluta di questo all'interno della mappa. Simili situazioni si possono verificare ad esempio qualora i sensori vengano posizionati da un operatore esterno che ne ignori i particolari ID sapendo solamente le posizioni spaziali dove essi vanno installati.

Nel nostro contesto quindi un numero N di nodi forma una rete in cui ogni nodo é sprovvisto dell'informazione relativa alla propria posizione assoluta; si deve a questo punto cercare un modo (un algoritmo) per far si che ogni nodo, comunicando con i nodi ad esso vicini, riesca a ricostruire in maniera più esatta possibile la propria posizione.

Ovviamente in assenza di nodi che conoscano la propria posizione rispetto ad un sistema di riferimento spaziale assoluto, il massimo che ci si può aspettare da un tale algoritmo é che riesca a ricostruire le posizioni relative tra i nodi, senza risalire né alla posizione assoluta né all'orientazione della mappa così ricostruita.

L'idea di base che cercheremo di seguire in questo progetto sarà quella di far sì che ogni nodo ricostruisca la propria posizione assoluta comunicando con i nodi ad esso vicino (ovvero dentro il proprio *range* di comunicazione) e con un nodo mobile che conosce la propria posizione assoluta e che invia tale informazione a tutti i nodi della rete.

Esistono in letteratura vari algoritmi per ottenere l'autolocalizzazione di una rete di sensori, ne presentiamo ora una rapida panoramica: Innanzitutto é possibile dividere gli algoritmi in centralizzati, ovvero algoritmi dove l'onere computazionale é affidato ad un processore centrale, e distribuiti, dove i calcoli da eseguire sono ripartiti tra i vari nodi. In un approccio centralizzato si ha una maggior accuratezza ma insorgono problemi di scalabilità e di traffico di pacchetti dalle rete verso il processore, mentre in un approccio distribuito si riscontrano maggiori robustezza e scalabilità a prezzo di un errore di stima mediamente più alto.

Una seconda distinzione può essere fatta tra algoritmi *Range-based* e algoritmi *Map-based*. I primi necessitano di avere informazioni inerenti le distanze relative tra i nodi, mentre i secondi si basano su altre tecniche quali l'analisi di direzione del segnale e del segnale riflesso. L'informazione relativa alla distanza tra i nodi può essere ottenuta in vari modi, in maniera più o meno accurata a seconda dell'approccio scelto; esistono metodi basati sul RSSI (*Received Signal Strength Indicator*), sull'AOA (*Angle of Arrival*), sul TOA (*Time of Arrival*) e sul TDOA (*Time Difference of Arrival*).

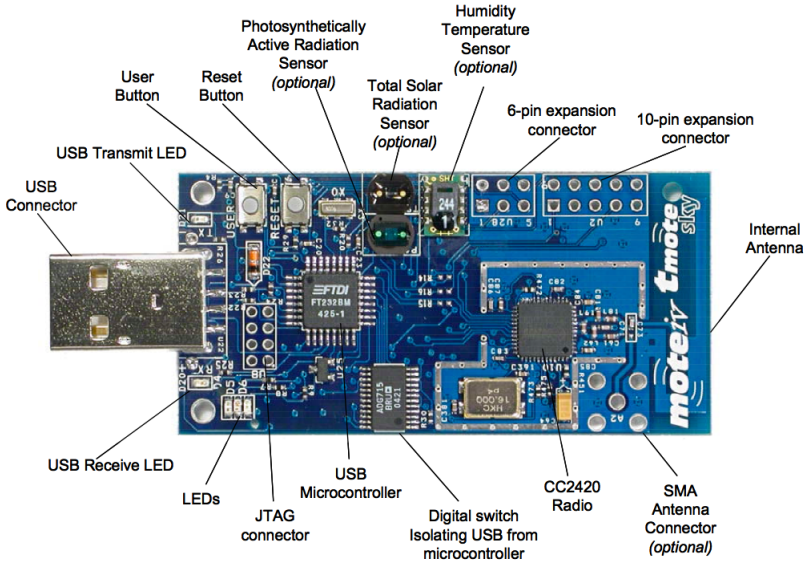
Infine si può distinguere tra algoritmi *Anchor-free* e *Anchor-based*. Un algoritmo *Anchor-free* non prevede la presenza di nodi che conoscono la propria posizione assoluta all'interno della rete (nodi ancora) e, come discusso

prima, potrà solo ricostruire la mappa delle distanze relative tra i nodi. Un algoritmo *Anchor-based* prevede invece la presenza di tali nodi e può quindi ricostruire la mappa assoluta della rete; la precisione con cui la mappa verrà ricostruita sarà ovviamente funzione di quanti nodi ancora sono presenti nella rete.

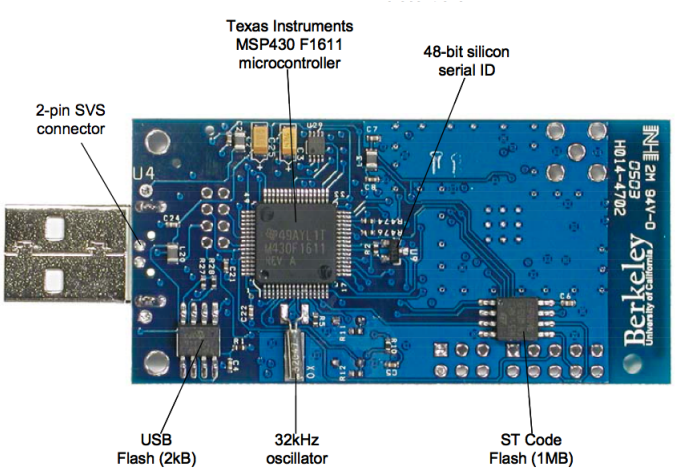
In questo progetto verranno presentati e analizzati vari algoritmi appartenenti alle diverse categorie elencate, simulando tramite MATLAB le loro prestazioni prima su dati generati da ambienti di simulazione e poi su dati sperimentali raccolti nel normale funzionamento della rete. Pregi e difetti che emergeranno nel corso di questa analisi verrano infine riassunti in una sezione conclusiva.

2 Componentistica

Una rete WSN é composta da molteplici nodi sensori (motes) intercomunicanti: non intendendo applicare un approccio gerarchico, non vi é stata necessità di impiegare diverse tipologie di dispositivi.



Tutti i nodi sono così stati implementati tramite Tmote ™sky[**?**], un mote prodotto dalla MoteIV (ora Sentilla) e capace di comunicare via wireless tramite il protocollo IEEE802.15.4. Le trasmissioni avvengono su una banda di 2.4GHz, ad una velocità di trasmissione di 250kbps; l'antenna ha una portata massima (nominale) di 125m in ambienti esterni e 50m all'interno, e opera ad un range di potenza da -25 a 0 dBm, quantizzato in 8 livelli.



» presente il supporto per sensori di umidità, temperatura e luce, un timer interno e molteplici porte ADC interne ed esterne, rispettivamente per monitorare lo stato del mote (batteria, temperatura) o acquisire dati da altri dispositivi collegati. Il microcontrollore é un Texas Instruments MSP430 a 16 bit da 8MHz, con 10KB di RAM e una memoria flash esterna di 1MB; l'alimentazione é fornita da due normali pile AA, essendo il consumo di energia estremamente ridotto.

Il mote possiede infine una porta USB, tramite cui é possibile caricare e scaricare i dati raccolti e i vari programmi da eseguire: questi ultimi sono stati scritti in nesC[?] (*Network Embedded Systems C*), un "dialetto" del più famoso C, specificatamente progettato per sistemi dedicati che adottano una logica ad eventi, e ottimizzato per i ristretti limiti di memoria dei dispositivi.

L'elaborazione é gestita da TinyOS[?], un sistema operativo non bloccante scritto in nesC, che governa cioé in maniera totalmente asincrona le operazioni di I/O, prediligendo quindi un'alta concorrenza (*concurrency*) a discapito di alcuni limiti nelle operazioni computazionalmente complesse; le interfacce più utilizzate sono GUI in Java, o semplicemente shell testuali. Sia nesC che TinyOS sono stati sviluppati dalla University of California, Berkeley e sono open-source secondo la Berkeley Software Distribution License.

3 Data Preprocessing

3.1 Ottimizzazione dei dati

I dati gentilmente forniti dall'ing. Filippo Zanella sui quali si é lavorato durante tutto il progetto provengono da una seduta di "ascolto" della durata di un weekend[1] eseguita al Dipartimento di Ing. dell'Informazione di Pado-

[1] La decisione di eseguirlo durante il weekend deriva dal fatto di voler ridurre al minimo le interferenze, come anche quelle dovute al transito delle persone

va. In un'ala di questa sede erano posizionati 45 sensori wireless secondo la seguente mappa: Utilizzando un apposito software, l'ing. Zanella ha la-

Figura 1: Mappa dei nodi wireless all'interno del DIE(Department of Information Engineering).

sciato che i nodi ricevessero e spedissero pacchetti per un intero weekend e per ognuno di essi ha creato una tabella con i soli pacchetti ricevuti, del tipo:

ID	RSSI	RSS	LQI	Ch	Power	Counter	Time
4	-53	-18	106	6	31	1	1290198322
4	-53	-18	103	6	31	1	1290198322
3	-68	-33	104	6	31	1	1290198322
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

dove:

- ID : identification number of the sensor
- RSSI: received signal strength inidcator
- RSS: received signal strength
- LQI: link quality indicator
- Ch: transmission channel
- Power: transmission power
- Counter: packet counter
- Time: packet timestamp

L'*ID* di ogni nodo é legato al suo nome sulla mappa secondo la seguente tabella:

ID	Nome
1	n6
2	n5
3	m6
4	m5
9	n4
11	n2
12	n1
13	m4
14	m3
15	m2
17	n10
18	n9
20	n7
21	m10
24	m7
25	c10
26	c9
28	c7
29	c6
31	c4
33	p1
34	p2
35	p3

ID	Nome
37	p5
38	p6
40	p8
41	p16
49	s4
51	s2
53	p20
54	p19
55	p18
56	p17
57	p24
58	p23
59	p22
60	p21
61	d1
62	d2
63	d3
64	d4
66	mm4
68	c1
71	mm5
72	c2

Per eseguire l'autolocallizzazione di tutti i nodi tramite alcuni nodi ancora non sono necessarie tutte queste informazioni. Si é scelto quindi di utilizzare solo l'*ID*, l'*RSSI* e l'*LQI*, gli unici che servissero effettivamente allo scopo.
Le tabelle erano contenute all'interno di un file di testo. Inizialmente é stato necessario utilizzare un'utile tool di Matlab capace di trasformare una tabella in una matrice creando una struttura chiamata *dataDEI*.
Successivamente la struttura é stata divisa in due (*dataDEI1*, *dataDEI2*) per separare il processo di identificazione da quello di simulazione(vedi sezione 3.2).
Come si può notare, la tabella e quindi la matrice così creata non é in un formato "computazionalemente" efficiente. Il passo successivo sarà quindi quello di creare una nuova struttura dati, chiamata questa volta *dataIN1* e *dataIN2*, che contenga i dati riorganizzati in modo differente.
In particolare, ogni blocco di dati(per esempio *dataIN1*) sarà costituito da un vettore colonna di 45 entries(una per ogni nodo). Ogni entry sarà costituita da una struttura dati(*struct* in Matlab) che contenga i campi *rssi*, *lqi*

e *indici*. Il campo *rssi* contiene a sua volta una matrice di 45 righe e numero di colonne dato dal numero massimo di pacchetti ricevuti.
Se si prende come esempio la seconda entry(corrispondente quindi al secondo nodo), la matrice del suo campo struttura *rssi* conterrà nella prima riga i valori RSSI di tutti i pacchetti che ha ricevuto il nodo 2 inviati dal nodo 1.
Il campo *lqi* é organizzato esattamente allo stesso modo mentre il campo *indici* é un vettore colonna di 45 righe in cui sono contenuti il numero di pacchetti ricevuti da ogni nodo(questo numero é infatti molto variabile a causa di perdite di pacchetto,ecc...).
Questo processo di ottimizzazione é stato ottenuto mediante la creazione di un apposito script Matlab chiamato *dataOptimization.m*.
In questo modo si é ottenuta una struttura dati molto più semplice da utilizzare ma soprattutto molto più efficiente a livello computazionale.

3.2 Identificazione del canale

Per eseguire l'autolocalizzazione é necessario riuscire a trasformare l'*rssi* ricevuto in una distanza seguendo una certa legge. Questa legge dovrebbe essere diversa per ogni nodo in quanto ognuno di essi é inserito in situazioni ambientali differenti[2] .
Lo scopo della prima parte di questo progetto é quindi quello di cercare di ottenere una buona identificazione del canale di trasmissione, che sia diverso per ogni nodo e che fornisca una funziona per trasformare l'RSSI ricevuto in una distanza.
Come anticipato in precedenza, l'identificazione deve essere sempre eseguita con un set di dati differente da quello della simulazione. I motivi sono principalmente due e sono molto semplici.
Il primo é che usare per la simulazione un set di dati sul quale é stata appena eseguita l'identificazione porta a un risultato *sicuramente* buono e quindi poco attendibile.
Il secondo motivo deriva dal confronto con la realtà. Se si dovesse svolgere un compito *real time* é palese che l'identificazione si svolgerebbe sui primi dati a disposizione mentre i successivi algoritmi utilizzerebbero dati completamente nuovi e in continuo cambiamento ma sicuramente differenti dai primi.
Per questi motivi l'identificazione del canale di trasmissione é stata fatta utilizzando esclusivamente il set di dati *dataIN1* mentre tutti gli altri algoritmi sono stati applicati al set di dati *dataIN2*.
Nell'autolocalizzazione dei nodi si deve tenere conto anche di qualcos'altro. In particolare del fatto che non si conosce la posizione esatta di tutti i nodi. Per costruire una mappa che dall'RSSI restituisca una distanza é necessario disporre sia di uno che dell'altra. Si capisce quindi che essa deve essere

[2]Due nodi di una stessa stanza o due nodi di stanze differenti necessitano di una legge diversa a causa della presenza di un muro tra di essi che modifica la trasmissione.

costruita a partire solo dai *nodi ancora*, quei nodi per i quali si conosce esattamente anche la posizione.
Guardando il grafico di figura (2) si può intravedere la grossa difficoltà che introduce questo progetto.

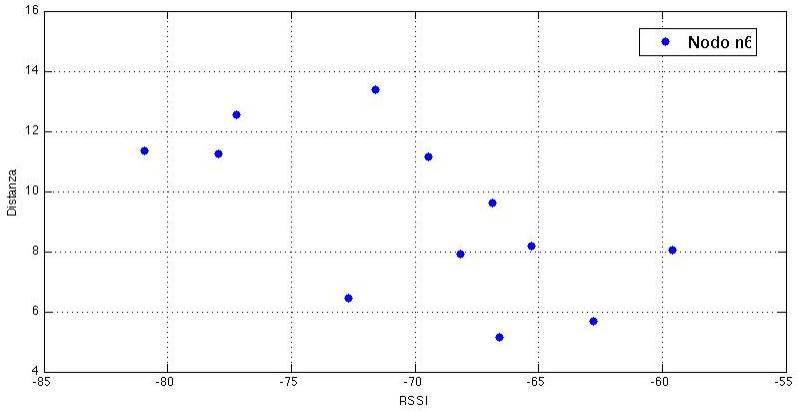


Figura 2: Valori di RSSI ricevuti dal nodo *n6* con le rispettive distanze.

Sono riportati sull'asse delle ascisse i valori di RSSI ricevuti dal nodo 1 e inviati da alcuni nodi ancora con cui esso comunica. Nelle ordinate sono invece rappresentate le distanze dal nodo 1 a cui erano posti i vari nodi ancora. Si può notare come la varianza dei dati sia decisamente elevata. Se si considera come esempio il nodo ancora distante circa 6 metri dal nodo 1 si vede dal grafico che l'RSSI corrispondente sia circa -72dB. Ma per quasi lo stesso valore di RSSI vi é anche un altro nodo ancora distante però 14 metri circa[3]!
E' intuitivo capire che costruire una mappa da RSSI a distanze abbastanza corretta sia quasi impossibile con questo tipo di dati.
Il metodo scelto per eseguire l'identificazione é creare una funzione di interpolazione che approssimi nel miglior modo possibile l'andamento della distanza al variare dell'RSSI.
I 3 grafici seguenti(3, 4, 5) rappresentano 3 tipi di interpolazioni differenti eseguite utilizzando il tool *polyfit* di Matlab. Essi contengono i rilevamenti di tutti i nodi ancora e le relative interpolazioni.

Nel primo caso l'interpolazione é di grado 1 e quindi é rappresentata da rette. Tutte le rette hanno pendenza negativa seguendo l'andamento corretto che dovrebbe avere la distanza all'aumentare del valore di RSSI.

[3]Il nodo 1 rappresenta il nodo *n6* mentre gli altri due nodi sono rispettivamente *c6* e *d4*.

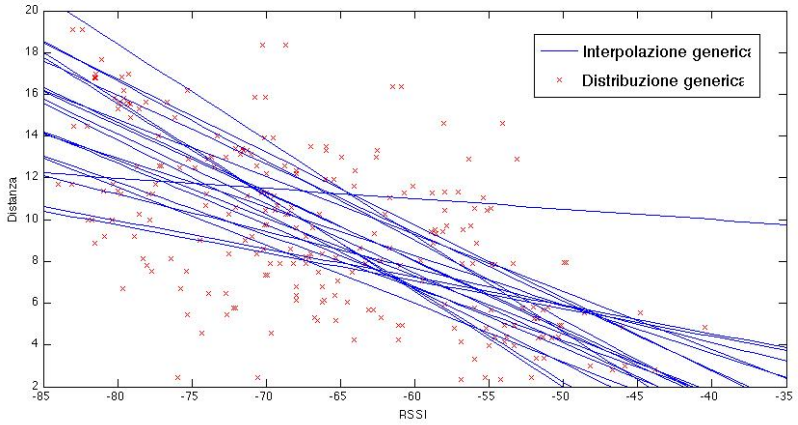


Figura 3: Interpolazione di grado 1.

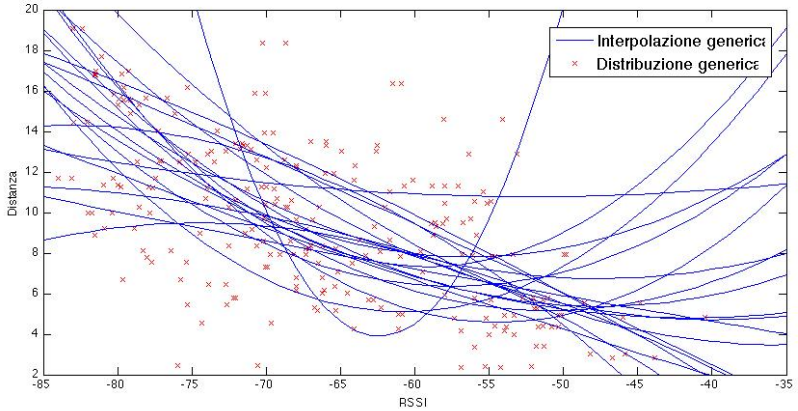


Figura 4: Interpolazione di grado 2.

Il secondo caso riporta una interpolazione di grado 2 eseguita quindi con delle parabole. Quasi tutte presentano concavità verso il basso ma alcune di esse hanno la concavità verso il basso. Queste ultime non rappresentano un comportamento verosimile dell'andamento che dovrebbe avere la funzione.
Il terzo caso presenta delle cubiche per l'interpolazione di grado 3. La situazione peggiora rispetto alle parabole e quindi le cubiche(come anche le interpolazioni di grado superiore) non possono in alcun modo essere considerate per una approssimazione corretta della funzione.
Si anticipa già che i migliori risultati sono stati ottenuti con l'interpolazione di grado 1. L'approssimazione lineare é infatti quella che limita gli errori il più possibile restando sempre molto limitata all'interno del intero range di RSSI al contrario di quanto avviene per parabole e cubiche che in alcuni casi, vicino ai limiti inferiori e superiori di RSSI, si comportano "male".

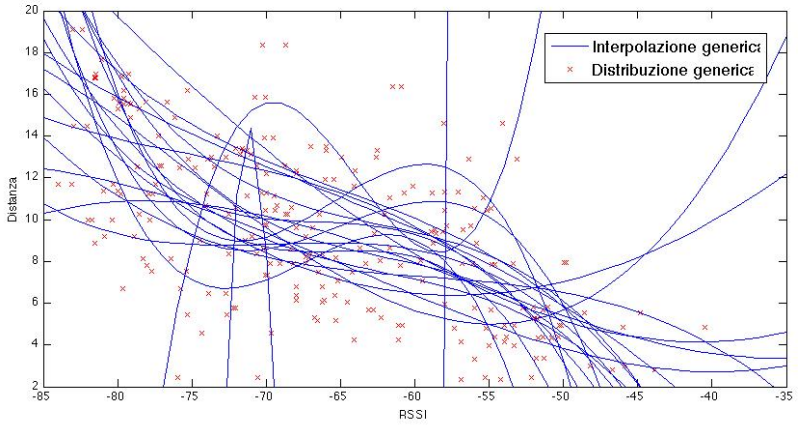


Figura 5: Interpolazione di grado 3.

E' necessario precisare che nell'identificazione del canale per sensori wireless é usale fare l'interpolazione delle misure con la distanza espressa in scala logaritmica infatti "in teorià' l'RSSI dovrebbe variare in maniera logaritmica rispetto alla distanza. E' emerso nel corso di varie simulazioni che nella nostra situazione le misure sono così disperse da essere rappresentate male da una scala logaritmica e comunque peggio che da una scala invariata. Di conseguenza **non** si é fatta nessun modifica alla scala della distanza per eseguire l'identificazione.
La funzione *polyfit* di Matlab fornisce i coefficienti polinomiali che interpolano l'insieme di punti utilizzando il criterio dei minimi quadrati.
Il problema può essere visto infatti come la stima di un modello di regressione lineare del tipo:

$$y = S\Theta$$

in cui si cerca di stimare il parametro Θ dando in ingresso il vettore di RSSI e in un uscita il vettore delle distanze.
Si é provato a questo scopo ad utilizzare l'approccio classico alla stima dei modelli utilizzando la funzione *iddata* e *oe* (per il modello *Output-Error*) che restituiscono la stima del parametro Θ e sono stati trovati risultati pressoché identici a quelli di *polyfit* nella sua versione più semplice(cioé quella che restituisce solo i parametri della retta interpolatrice).
Tuttavia *polyfit* é una funzione che utilizzata propriamente nella sua versione più complessa restituisce anche altri parametri oltre ai coefficienti polinomiali, utili ad una migliore interpolazione[4]. Per questo motivo e anche per una maggiore velocità computazionale si é preferito usare *polyfit*.
E' stato possibile in questo modo costruire una retta interpolatrice per ogni nodo ancora. Una volta ottenuti tutti i parametri é possibile, tramite l'RSSI

[4]Per maggiori informazioni consultare l'help di Matlab.

ricevuto, ricostruire la distanza.
Vista la varianza della distanza al variare dell'RSSI i risultati che si potranno ottenere non potranno essere sicuramente precisi.
Per cercare di ridurre gli errori si é scelto di filtrare leggermente i dati e di scegliere solo le misurazioni che avessero un $RSSI > -90$ e un $LQI > 90$. Tutte queste operazioni vengono eseguite dallo script Matlab *CHidentification*.

3.3 Calcolo delle distanze

Tutte le informazioni sul canale fin qui ottenute vengono utilizzate dallo script *DISestimation* per calcolare le distanze. Per questo scopo si usa il nuovo set di dati *dataIN2* e tramite la funzione *polyval* si ricostruiscono le distanze.
La funzione Matlab *polyval* prende in ingresso sia i parametri calcolati in precedenza con la funzione *polyfit* sia i nuovi dati di RSSI e restituisce la distanza tra i nodi.
In particolare, la funzione *DISestimation* crea una matrice quadrata delle distanze dove ad ogni posto *(i,j)* corrisponde la distanza dal nodo i al nodo j.
Come specificato in precedenza, non si conoscono tutte le distanze e le posizioni precise di tutti i nodi ma solo di un sottoinsieme di essi chiamati nodi ancora. Da questo segue che la matrice delle distanze non potrà essere completa e presenterà dei buchi nelle posizioni *(i,j)* per esempio in cui i sia un nodo ancora mentre j no. In questa situazione si inserisce il valore *Inf* che nelle operazioni successive verrà opportunamente scartato.

4 Approccio *Range Based*

4.1 Introduzione

Un approccio utilizzato in letteratura per risolvere il problema di autolocalizzazione é quello che viene detto *range-based*. Vanno sotto questo nome tutti quegli algoritmi che richiedono in ingresso informazione sulle distanze o sulle direzioni relative tra i nodi.

Ad esempio in letterattura si possono trovare metodi basati su:

- RSSI: *Received Signal Strength Indicator*. L'RSSI é un indice relativo alla potenza con cui un segnale trasmesso da un nodo i viene ricevuto da un nodo j; esistono poi modelli che legano il valore di questo indice alla distanza relativa tra i due nodi. Questi modelli peró non tengono tipicamente conto di ostacoli che possono frapporsi tra i nodi e dei fenomeni di rifrazione e riflessione delle onde elettromagnetiche, risultando quindi molto imprecisi.

- TOA: *Time of Arrival*. Il TOA é la differenza temporale tra l'istante in cui il nodo i comincia a trasmettre e quello in cui il nodo j riceve il segnale. Anche in questo caso esistono modelli che legano il valore del TOA alla distanza tra i nodi che risultano meno affetti dal rumore rispetto al metodo dell'RSSI. Sono tuttavia modelli complicati e anche l'hardware necessario a rilevare il TOA risulta essere piú complesso;

- DOA: *Direction of Arrival*. Il DOA é un indice legato alla direzione di arrivo del segnale ricevuto. I metodi che da questa informazione risalgono alla posizione dei nodi sono molto efficenti, ma hanno un carico computazionale elevato e sono molto sensibili ai fenomeni di riflessione e rifrazione.

Un'ulteriore distinzione puó essere effettuata tra algoritmi *anchor-free* e algoritmi *anchor-based*. Nel primo caso i nodi non sono in possesso di alcuna informazione riguardo la loro posizione assoluta, ma possono solo ricavare informazioni locali (tramite comunicazione con i nodi adiacenti); é chiaro che un algoritmo di questo genere potrá ricavare una mappa della rete senza peró poterla posizionare ed orientare correttamente nello spazio ambiente. Nel secondo caso si presuppone invece che un certo numero di nodi presenti nella rete conosca esattamente la propria posizione rispetto ad un sistema di riferimento; in questo modo la rete puó essere mappata esattamente nell'ambiente circostante. Le prestazioni di questo tipo di algoritmi sono affette dal numero di nodi ancora. Ovviamente i vantaggi derivanti dall'approccio *anchor-based* sono pagati in termini di complessitá computazionale e di hardware.

In questo contesto é stato scelto un approccio basato sull'*RSSI* dal momento che i nodi presenti al dei sono dotati di una capacitá computazionale limitata e restituiscono solo questo indice (assieme all'LQI).

4.2 Baricentro

Lo script *bari.m* calcola la posizione del nodo incognito semplicemente calcolando il baricentro dei 3 nodi ancora più vicini. Non può essere considerato propriamente un metodo Range Based in quanto non utilizza direttamente la distanza per calcolare la posizione del nodo incognito ma la utilizza per trovare quelli con la distanza minore.
In figura (6, 7) sono riportati rispettivamente l'andamento dell'errore medio al variare delle ancore e il grafico dell'autolocalizzazione per un numero di ancora pari alla metà di quelle totali.

4.3 Combinazione Convessa della distanza

Lo script *conv.m* appositamente creato esegue una semplice combinazione convessa pesata con le distanza sui 3 nodi ancora più vicini.
In particolare ogni nodo incognito riordina il vettore dei nodi ancora con cui

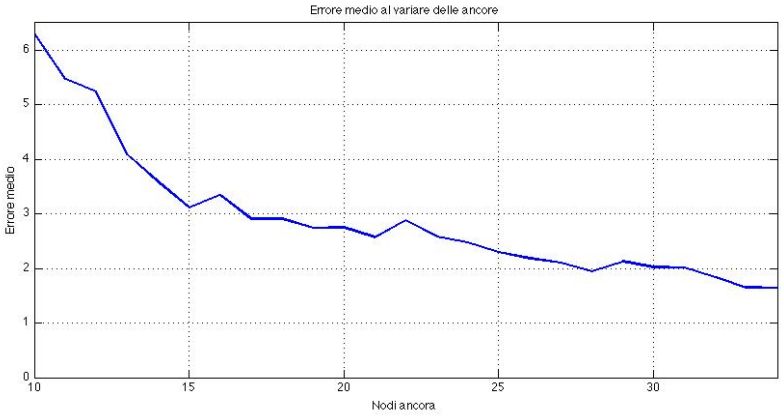


Figura 6: Errore medio al variare dei nodi ancora per il metodo *bari.m*

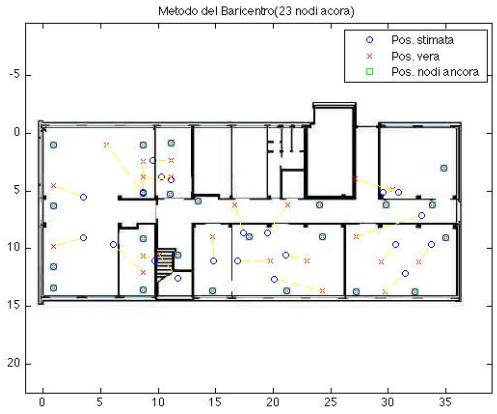


Figura 7: Autolocalizzazione con il metodo del baricentro

comunica in base alla loro distanza e sceglie i 3 più vicini. Successivamente esegue una combinazione convessa dei 3 nodi ancora scelti pesata con le distanze. A questo proposito le distanze sono state normalizzate in modo che la somma dei coefficienti della combinazione avessero somma unitaria[5]. In figura (8, 9) sono riportati rispettivamente l'andamento dell'errore medio al variare delle ancore e il grafico dell'autolocalizzazione per un numero di ancore pari alla metà di quelle totali.

[5]Deve essere così per definizione di combinazione convessa.

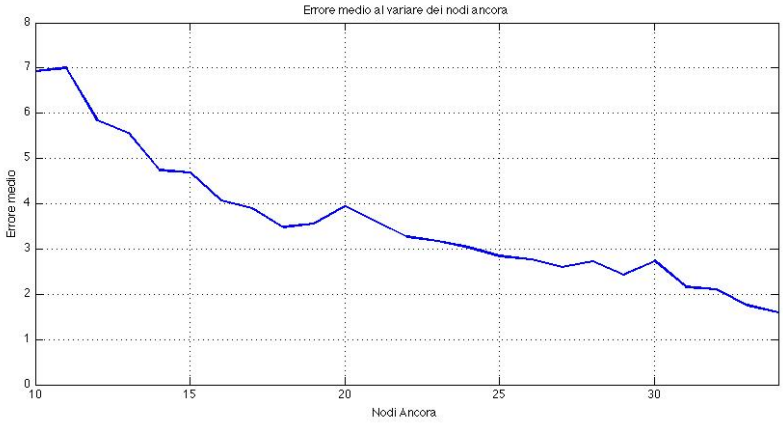


Figura 8: Errore medio al variare dei nodi ancora per il metodo *conv.m*

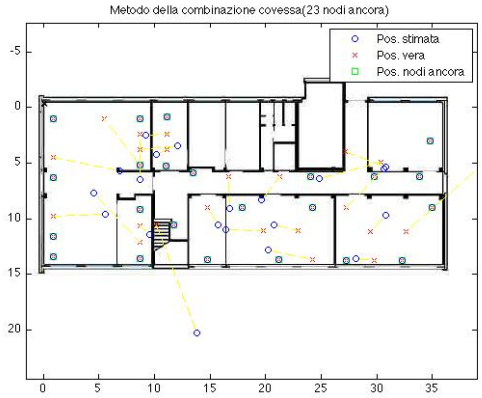


Figura 9: Autolocalizzazione con il metodo della combinazione convessa.

4.4 Metodo dei minimi quadrati

4.4.1 Introduzione

Un primo approccio seguito per la risoluzione del problema di autolocalizzazione é il metodo dei minimi quadrati; tale metodo é di fatto un algoritmo di stima statica su modelli lineari molto semplice e facilmente implementabile.

Si consideri un sistema di misura di una certa grandezza $\vartheta \in \mathbb{R}^p$ organizzato nella forma:

$$y = s(\vartheta) + w$$

dove $s(\cdot)$ é un'arbitraria funzione di ϑ descrittiva del sistema di misura e w é un rumore additivo, somma di tutti i disturbi che affliggono il processo di misura (inclusa quindi la deviazione della caratteristica reale dello strumento di misura dalla caratteristica ideale $s(\cdot)$); benché sembri ragionevole

assumere w gaussiano, come afferma il *teorema del limite centrale*, non si fa tale ipotesi ma si suppone umicamente che $E[w] = 0$ e $var(w) < +\infty$.

Si supponga poi di avere a disposizione più misure di questa forma, ottenute anche con strumenti diversi:

$$y_t = s_t(\vartheta) + w_t \quad , \quad t = 1, 2 \cdots N$$

e si consideri un'approssimazione lineare (serie di *Taylor* troncata al primo ordine) della funzione $s(\cdot)$:

$$y_t = s_t\vartheta + w_t \quad , \quad t = 1, 2 \cdots N$$

dove si é supposto nullo il termine affine eventualmente derivante dall'approssimazione (se così non fosse basta sottrarre tale valore di *offset* dalle misure disponibili). L'insieme di misure può quindi essere organizzato in un vettore:

$$Y = [y(1), y(2) \cdots y(N)]^T$$

che soddisfa l'equazione

$$Y = S\vartheta + W \tag{1}$$

dove: $S = [s_1^T, s_2^T \cdots s_N^T]^T \in \mathbb{R}^{N \times p}$ e $W = [w_1^T, w_2^T \cdots w_N^T]^T \in \mathbb{R}^N$.

Il problema che ci si pone é quello di trovare stime efficienti di ϑ, note che siano le misure Y, la matrice S e la matrice R, caratterizzante la varianza di W pari appunto a $\sigma^2 R$. Si suppone in questo contesto che la matrice S sia a rango colonne pieno e che la matrice R sia definita positiva. La prima ipotesi é senz'altro ragionevole e sta ad indicare il fatto che le misure sono prese in maniera indipendente e portano la maggior informazione possibile; la seconda invece serve per garantire l'esistenza di R^{-1} che comparirà nelle formule[6].

Il criterio dei minimi quadrati pesati indica come stima di ϑ la quantità:

$$\hat{\vartheta} = Argmin_{\vartheta \in \mathbb{R}^p} \left\{ \|Y - S\vartheta\|_Q^2 \right\} \tag{2}$$

dove Q é una matrice della forma $Q = diag(q_1, q_2 \cdots q_N)$ che ha come elementi della diagonale degli opportuni pesi (strettamente positivi) da assegnare alle componenti di Y in base a quanto accurata si ritiene essere la misura, e $\|\cdot\|_Q$ é la norma indotta da tale matrice. Per motivi di generalità si include già da subito il caso in cui Q é una qualunque matrice simmetrica e definita positiva. Si noti che é stato implicitamente assunto che esista un minimo di tale quantità e che esso sia unico.

Ricercando tale minimo minimo con l'usuale metodo di uguagliare a zero la derivata (vettoriale), si trova:

$$\hat{\vartheta} = (S^T Q S)^{-1} S^T Y \tag{3}$$

[6]Se tale ipotesi non fosse verificata basta sostituire a R^{-1} una pseudoinversa, ad esempio la *pseudoinversa di Moore-Penrose*. In questo modo però non é più garantita l'unicità del risultato.

che é l'espressione dello stimatore ai minimi quadrati[7].

Osservazione 4.1 *A rigore matematico si dovrebbe poi verificare che tale soluzione sia effettivamente un minimo calcolando la matrice Hessiana delle derivate seconde e controllando che sia almeno semidefinita positiva. Tale calcolo si può tralasciare notando che la funzione* $\|\cdot\|_Q^2$ *é pari, non negativa e strettamente crescente in un intorno di* $0_{\mathbb{R}^N}$*. Essendo verosimilmente* $\hat{\vartheta}$ *prossimo a* ϑ *la quantità* $Y - S\hat{\vartheta}$ *é prossima a* W *che ha media nulla; si può quindi affermare che in un intorno della soluzione trovata la funzione in esame ammette un solo punto stazionario che é necessariamente un minimo.*

Si precisa tuttavia che tale discussione é di carattere puramente qualitativo e non va interpretata come una vera e propria dimostrazione.

Si può ora pensare di utilizzare la matrice Q come parametro di ottimizzazione al fine di ottenere una stima migliore; a tal fine si calcola il momento secondo della quantità $\hat{\vartheta} - \vartheta$, ovvero la varianza dello stimatore:

$$\begin{array}{rcl} var(\hat{\vartheta}) & = & E[(S^TQS)^{-1}S^TY - \vartheta)(S^TQS)^{-1}S^TY - \vartheta)^T] \\ & = & S^TQS)^{-1}S^T \cdot R \cdot (S^TQS)^{-1}S^T)^T \end{array} \tag{4}$$

e se ne ricerca un minimo al variare di Q. La soluzione é nota come *Teorema di Gauss-Markov* e afferma che la matrice Q che minimizza la varianza di $\hat{\vartheta}$ esiste, é unica ed é pari a R^1.

Pertanto il miglior stimatore ai minimi quadrati risulta essere la quantità:

$$\hat{\vartheta} = (S^TR^{-1}S)^{-1}S^TR^{-1}Y \tag{5}$$

Osservazione 4.2 *Si può dimostrare che, nel caso in cui il rumore additivo sia gaussiano, tale stimatore coincide con lo stimatore di massima verosimiglianza di* ϑ*.*

Ci si può infine porre il problema di calcolare una stima della quantità σ^2. Tralasciando i passaggi matematici analoghi al calcolo precedente, si trova:

$$\hat{\sigma}^2 = \frac{1}{N}||Y - S\hat{\vartheta}||_{R^{-1}}^2, \tag{6}$$

che é lo stimatore di σ^2 a minima varianza. Tuttavia tale stimatore non é corretto essendo

$$E[\hat{\sigma}^2] = \sigma^2\frac{N-p}{N}.$$

Uno stimatore corretto, anche se non a minima varianza, é dato dalla formula:

$$\hat{\sigma}^2 = \frac{1}{N-p}||Y - S\hat{\vartheta}||_{R^{-1}}^2,$$

Per una trattazione più completa si rimanda a [25].

[7] L'esistenza della matrice $(S^TQS)^{-1}$ é garantita dal fatto che Q é definita positiva e S ha rango colonne pieno

4.4.2 Applicazione al problema di autolocalizzazione

Indipendentemente dall'approccio tipico della teoria degli errori usato nella sezione precedente, il metodo dei minimi quadrati può essere usato per stimare la quantità vettoriale ϑ in un qualunque problema della forma (1). Si tratta dunque ora di formulare in questa forma il problema di autolocalizzazione.

Si consideri il nodo j, la cui posizione (x, y) é incognita. Si supponga che tale nodo possa comunicare con n nodi ancora, pertanto sia a conoscenza della posizione (x_i, y_i) di ognuno di tali nodi e della distanza d_i che lo separa da essi (in questo contesto ricavata tramite *RSSI*). Si modellizzi l'errore di misura della distanza come una variabile aleatoria v_i a media nulla e varianza finita. É allora possibile costruire il sistema non lineare di n equazioni:

$$\begin{cases} (x - x_1)^2 + (y - y1)^2 &= d_1^2 + v_1 \\ \vdots & \\ (x - x_n)^2 + (y - yn)^2 &= d_n^2 + v_n \end{cases} \tag{7}$$

Svolgendo i quadrati e sottraendo l'ultima equazione alle rimanenti $n-1$ si arriva ad ottenere un sistema lineare in (x, y), in quanto i termini in x^2 e y^2 si cancellano:

$$\begin{cases} x_1^2 - x_n^2 - 2(x_1 - x_n)x + y_1^2 - y_n^2 - 2(y_1 - y_n)y &= d_1^2 - d_n^2 + w_1 \\ \vdots & \\ x_{n-1}^2 - x_n^2 - 2(x_{n-1} - x_n)x + y_{n-1}^2 - y_n^2 - 2(y_{n-1} - y_n)y &= d_{n-1}^2 - d_n^2 + w_{n-1} \end{cases} \tag{8}$$

dove é stato implicitamente definito $w_i = v_i - v_n$.

É immediato notare che tale sistema é proprio della forma (1) con le posizioni:

$$\begin{aligned} \vartheta &= \begin{bmatrix} x \\ y \end{bmatrix} \\ S &= \begin{bmatrix} -2(x_1 - x_n) & -2(y_1 - y_n) \\ \vdots & \vdots \\ -2(x_{n-1} - x_n) & -2(y_{n-1} - y_n) \end{bmatrix} \\ Y &= \begin{bmatrix} d_1^2 - d_n^2 - (x_1^2 - x_n^2) - (y_1^2 - y_n^2) \\ \vdots \\ d_{n-1}^2 - d_n^2 - (x_{n-1}^2 - x_n^2) - (y_{n-1}^2 - y_n^2) \end{bmatrix} \end{aligned}$$

Applicando la (5) si trova una stima della posizione del nodo j. Ripetendo tale procedimento per ogni nodo di posizione incognita, si può ricavare una stima della posizione dei nodi di tutta la rete.

Si noti che nulla é stato detto riguardo la matrice R della varianza di rumore. Effettivamente in questo contesto ricavare informazioni sul rumore

che affligge le misure é tutt'altro che semplice, se non impossibile, a causa delle interferenze elettromagnetiche che variano di ora in ora e dei fenomeni di riflessione e rifrazione. Gli elementi della matrice R sono stato quindi usati come parametri di *tuning*, al fine di ottenere una stima migliore.

Osservazione 4.3 *Sicuramente l'arguto lettore avrà notato che nulla garantisce che la matrice S abbia rango colonne pieno (pari a 2); si analizzi meglio questa condizione. Avere una matrice S di rango 2 signigica avere almeno 2 equazioni nel sistema (8) ovvero, contando anche l'equazione relativa al nodo n che viene sottratta alle rimanenti, avere almeno 3 nodi ancora. Perché poi queste due equazioni siano linearmente indipendenti succedere che le due equazioni corrispondenti in (7) e l'equazione n-esima siano linearmente indipendenti tra di loro. Ciò equivale di fatto ad imporre che ogni nodo da localizzare riesca a comunicare con almeno tre nodi ancora non allineati, condizione che appare intuitiva.*

Per una trattazione più completa si rimanda a [26].

4.4.3 Risultati sperimentali

Il metodo dei minimi quadrati risulta essere un metodo molto veloce e non richiede più iterazioni per convergere al risultato; di contro presenta un elevato errore medio ed é molto sensibile al rumore. Il problema dei minimi quadrati, sinfatti, si sa essere un problema numericamente malcondizionato, quindi una variazione δY nel vettore dei termini noto o una variazione δA nella matrice dei coefficenti provoca una variazione $\delta\vartheta$ di entità elevata:

$$\delta\vartheta >> max\{\delta Y, \delta A\}.$$

Tale proprietà (non desiderabile) é purtroppo intrinseca nel problema ed é causa appunto di un elevato errore nel calcolo delle posizioni.

In figura (10) é riportata la mappa dei nodi che vengono localizzati dall'algoritmo. Le $\times$ rappresentano le posizioni vere dei nodi mentre i $\circ$ indicano le posizioni stimate; i nodi ancora sono contraddistinti da un quadrato verde.

Questa stima é stata calcolata con un numero di nodi ancora pari a 20; definendo la quantità:

$$err = \left[\sqrt{(x^{(1)} - \hat{x}^{(1)}_{MQ})^2 + (y^{(1)} - \hat{y}^{(1)}_{MQ})^2}, \quad \cdots \quad \sqrt{(x^{(M)} - \hat{M}^{(N)}_{MQ})^2 + (y^{(M)} - \hat{y}^{(M)}_{MQ})^2} \right] \tag{9}$$

dove M é il numero di nodi non ancora da localizzare, si può considerare la norma di tale vettore come un indice sull'errore medio che viene commesso nel processo di autolocalizzazione. In questo caso il risultato trovato é di 3.2251 m.

Un primo risultato intuitivo é che l'errore medio così calcolato diminuisca all'aumentare del numero di nodi ancora. Come si può osservare in figura

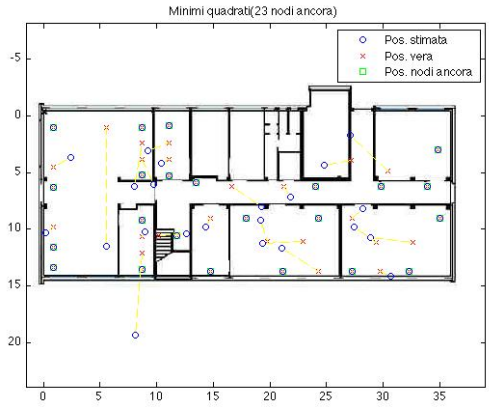


Figura 10: Mappa della rete localizzata tramite minimi quadrati

(11) tale andamento monotono decrescente é effettivamente rispettato. Il primo dato é relativo ad un numero di nodi ancora pari a 8, appena sufficente (assieme ad una sapiente disposizione spaziale dei nodi ancora stessi) a garantire che ogni nodo da localizzare riesca almeno a comunicare con tre nodi ancora non allineati. Si noti che la velocità con cui l'errore tende a zero non é elevata quanto a priori si potrebbe immaginare; questo fatto é dovuto alla bassa qualità dei dati relativi alle distanze ricavati a partire dall'RSSI, che é un metodo dai risultati poco attendibili.

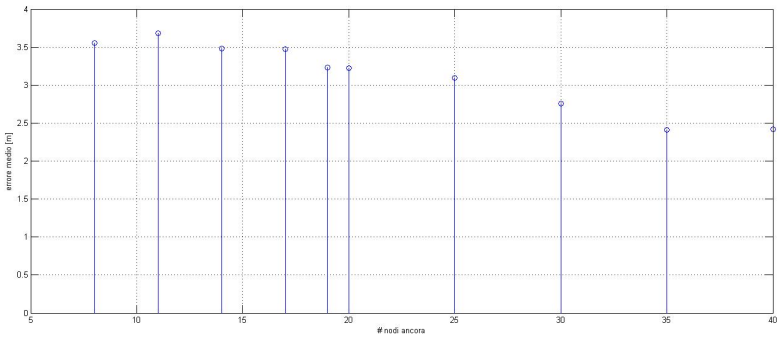


Figura 11: Andamento dell'errore medio in funzione della quantità di nodi ancora

Un altro aspetto non intuitivo é il fatto che inizialmente il grafico sia pressocché piatto: anche aumentando il numero di nodi ancora l'errore non cala significativamente se i nodi da localizzare sono superiori al 50% del totale. La cusa di questo insolito comportameneto va questa volta cercata nella densità dei nodi disposti al DEI: quando la rete é stata predisposta é stata

pensata per applicazioni di *localizzazione* e *tracking*, quindi il numero e la disposizione dei nodi é ottimale per tali applicazioni. Nel problema di autolocalizzazione invece alcuni nodi passano da ancore a nodi da localizzare, diminuendo di fatto la densità dei nodi nello spazio; é chiaro quindi che in questo caso la rete, ottima nel caso precedente, risulta sottodimensionata e non da risultati accettabili a meno che non si aumenti il numero di nodi oltre una soglia minima tale da garantire una buona autolocalizzazione. Sperimentalmente si può notare che il numero di ancore necessario é pari a 20-25, ovvero circa metà nodi; un tale numero é impensabile in applicazioni di utilizo pratico, ma si ribadisce il fatto che in queste applicazioni si ha una densità di nodi più elevata pertanto la percentuale di nodi ancora risulta minore.

L'ultimo aspetto da analizzare riguarda la matrice R caratterizzante la varianza del rumore. Essendo che ogni nodo sfrutta solo l'informazione proveniente dai nodi ancora e non dagli altri nodi, l'errore che si commette nel localizzare il nodo i é indipendente da quello commesso nel localizzare il nodo j, pertanto la matrice R sarà una matrice diagonale. Verosimilmente gli errori commessi per ogni nodo non saranno troppo diversi tra loro pertanto gli elementi non nulli di R saranno dello stesso ordine di grandezza. Per individuare quale matrice fosse più opportuno usare si é quindi usata la formula:

```
Q = q*eye(length(b)) + diag(rand(length(b),1)*q/10) ;
```

dove il parametro q é stao fatto variare nel range $0.001 \div 1000$. L'errore di stima che ne consegue (a parità di nodi ancora) non cambia significativamente e questo fatto trova giustificazione nel fatto che la matrice R compare sia in forma diretta che in forma inversa R^{-1}, pertanto (data la sua forma diagonale e la confrontabilità degli elementi non nulli) il contributo da essa portato é pressocché nullo. Per velocizzare i calcoli é stata fatta quindi la posizione $R = I$.

4.5 Filtro di Kalman

4.5.1 Introduzione

Si consideri un sistema descritto dalla dinamica:

$$\begin{cases} x_{k+1} &= F_k x_k + B_k u_k + w_k \\ z_k &= H_k x_k + v_k \end{cases} \tag{10}$$

dove w_k e v_k sono variabili aleatorie di media nulla e varianza Q_k e R_k rispettivamente. Lo stato inziale x_0 é anch'esso una v.a. di media $\hat{x}_0$ e varianza P_0. Tutte e tre queste v. a. sono supposte Gaussiane.

Il filtro di *Kalman* é un algoritmo iterativo che permette di stimare lo stato x_k a partire dal processo di osservazione y_k nota che sia la dinamica del

sistema e si articola in due fasi: predizione e filtraggio. Detta P_k la varianza dello stato x_k si ha, per il passo di predizione

$$\begin{array}{rcl} \hat{x}_{k|k-1} & = & F_k \hat{x}_{k-1|k-1} + Bu_k \\ P_{k|k-1} & = & F_k P_{k-1|k-1} F_k^T + Q_k \end{array} \tag{11}$$

e per il passo di filtraggio

$$\begin{array}{rcl} \hat{x}_{k|k} & = & \hat{x}_{k|k-1} + K_k(z_k - H_k \hat{x}_{k|k-1}) \\ P_{k|k} & = & P_{k|k-1} + K_k(H_k P_{k|k-1} H_k^T) K_k^T \end{array} \tag{12}$$

dove $K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T)^{-1}$.

É stato ampiamente dimostrato in letteratura che, dopo un breve transitorio iniziale, la stima $\hat{x}_{k|k}$ converge al valore x_k in maniera molto precisa. Queste qualità vengono però a mancare nel caso di sistemi non lineari o di variabili aleatorie a distribuzione multimodale, ma tali aspetti verranno approfonditi in seguito.

Esiste poi una formulazione alternativa delle equazioni del filtro di *Kalman*, che prende il nome di *filtro di Kalman in forma d'informazione*; le equazioni che lo descrivono hanno le stesse qualità del filtro classico, ma presentano un minor carico computazionale quanto maggiore é la dimensione del processo di osservazione rispetto al processo di stato. Definite le quantità:

$$\Psi_k = P_k^{-1}, y_k = \Psi_k x_k = P_k^{-1} x_k \tag{13}$$

si può riscrivere il passo di predizione come:

$$\begin{array}{l} \Psi_{k|k-1} = [F_k \Psi_{k-1|k-1}^{-1} F_k^T + Q_k]^{-1} \\ L_{k|k-1} = \Psi_{k|k-1} F_k \Psi_{k-1|k-1}^{-1} \\ \hat{y}_{k|k-1} = L_{k|k-1} \hat{y}_{k-1|k-1} + \Psi_{k|k-1} B_k u_k \end{array} \tag{14}$$

e quello di filtraggio:

$$\begin{array}{l} \Psi_{k|k} = \Psi_{k|k-1} + \Phi_k \\ \hat{y}_{k|k} = \hat{y}_{k|k-1} + i_k \\ \Phi_k = H_k^T R_k^{-1} H_k \\ i_k = H_k^T R_k^{-1} z_k \end{array} \tag{15}$$

La quantità i_k prende il nome di *variabile d'informazione* e rappresenta appunto la quantità di informazione nuova che é portata dalla misura z_k.

La stima dello stato x_k può poi essere ricavata tramite l'inversa della (13).

4.5.2 Il filtro di Kalman esteso

Un'ipotesi molto restrittiva per l'applicazione del filtro di *Kalman* é la linearità del sistema in esame. Per questo motivo in letteratura sono stati

studiati vari metodi per applicare le equazioni (11 - 12) o (14 - 15) ad un sistema non lineare della forma:

$$\begin{cases} x_k &= f(x_{k-1}, u_k) + w_k \\ z_k &= h(x_k) + v_k \end{cases}$$

dove w_k e v_k sono le consuete variabili aleatorie gaussiane di media nulla e varianza Q_k e R_k rispettivamente.

Il risultato più intuitivo, che prende il nome di *filtro di Kalman esteso*, si basa sull'approssimazione al primo ordine delle funzioni $f(\cdot)$ e $h(\cdot)$. Dette J_x^f e J_x^h le matrici Jacobiane di tali funzioni, calcolate nello stato x (e in u_k per J_x^f), si possono implementare le seguenti equazioni per il passo di predizione:

$$\begin{aligned} \hat{x}_{k|k-1} &= f(\hat{x}_{k-1|k-1}, u_k) \\ P_{k|k-1} &= J_x^f P_{k-1|k-1} J_x^{f^T} \end{aligned} \tag{16}$$

e il passo di stima:

$$\begin{aligned} K_k &= P_{k|k-1} J_x^{h^T} [J_x^h P_{k|k-1} J_x^{h^T} + R_k]^{-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k(z_k - h(\hat{x}_{k|k-1})) \\ P_{k|k} &= P_{k|k-1} - K_k [J_x^h P_{k|k-1} J_x^{h^T} + R_k] K_k^T. \end{aligned} \tag{17}$$

In questo caso però non esistono risultati teorici che garantiscono la convergenza della stima al valore vero dello stato x_k e potrebbe anzi succedere che il processo $\hat{x}_{k|k}$ diverga. In molte situazioni pratiche, tuttavia, il filtro di *Kalman* esteso riesce a garantire prestazioni ottimali.

Come la sua controparte non approssimata, anche il filtro di *Kalman* esteso ammette la versione in forma d'informazione, detta appunto *filtro di Kalman esteso in forma d'informazione*. Assumendo le posizioni (13) si può ricavare:

$$\begin{aligned} \Psi_{k|k-1} &= [J_x^f \Psi_{k-1|k-1}^{-1} J_x^{f^T} + Q_k]^{-1} \\ \hat{y}_{k|k-1} &= \Psi_{k|k-1} f(\hat{x}_{k-1|k-1}, u_k) \end{aligned} \tag{18}$$

e:

$$\begin{aligned} \Psi_{k|k} &= \Psi_{k|k-1} + \Phi_k \\ \hat{y}_{k|k} &= \hat{y}_{k|k-1} + i_k \\ \Phi_k &= J_x^{h^T} R_k^{-1} J_x^h \\ i_k &= J_x^{h^T} R_k^{-1} [z_k - h(\hat{x}_{k|k-1}) + J_x^h \hat{x}_{k|k-1}] \end{aligned} \tag{19}$$

Anche in questo caso le equazioni in forma di informazione sono convenienti quando il processo di misura ha dimensione maggiore di quella del processo di stato.

4.5.3 Applicazione al problema di autolocalizzazione

Per poter applicare le equazioni del filtro di *Kalman* (in una delle sue forme viste nelle sezioni precedenti) si deve prima ricavare un modello di stato

per la posizione dei nodi di tutta la rete. Sia quindi $x_k^{(i)} = [p_{x,k}^{(i)}, p_{y,k}^{(i)}]^T$ la posizione del nodo (i) all'istante k[8]; allora lo stato dell'intera rete é descritto dal vettore:

$$x_k = [x_k^{(1)}, \cdots, x_k^{(N)}]^T$$

che possiamo supporre soggetto alla dimanica:

$$x_k = x_{k-1} + w_k, \tag{20}$$

con w_k rumore bianco gaussiano a media nulla e varianza Q_k.

Si definisca poi la quantità $z_k^{(i,j)}$ come la distanza tra il nodo i e il nodo j, misurata dal nodo i. La dinamica di tale processo può essere modellizata come:

$$\begin{aligned} z_k^{(i,j)} &= h^{(i,j)}(x_k^{(i)}, x_k^{(j)}) + v_k^{(i)} \\ & \quad ||x_k^{(i)} - x_k^{(j)}|| + v_k^{(i)} \\ & \quad \sqrt{(p_{x,k}^{(i)} - p_{x,k}^{(j)})^2 + (p_{y,k}^{(i)} - p_{y,k}^{(j)})^2} + v_k^{(i)} \end{aligned} \tag{21}$$

Raggruppando tutti i possibili processi scalari di questo tipo in un vettore z_k, si può ottenere il completo processo delle osservazioni. Si noti fin da subito che tale vettore avrà dimensioni che dipendono da N e dalla topologia della rete, in quanto ogni nodo comporta la presenza in z_k di un numero di componenti tanti quanti sono i nodi con cui esso può comunicare; essendo questi tipicamente strettamente maggiori di due (anche per questioni di localizzabilità della rete) si ha che $dim(x_k) << dim z_k$ cosicché le equazioni più convenienti da implementare sono quelle del filtro in forma di informazione.

Un'obiezione a questa considerazione potrebbe essere quella di osservare che, in linea teorica, $z_k^{(i,j)} = z_k^{(j,i)}$ dunque il vettore z_k risulterebbe avere metà componenti. Questo sarebbe sicuramente vero in un canale privo di rumore o perfettamente simmetrico, ma dal momento che, in generale, $v_k^{(i)} \neq v_k^{(j)}$ si ha che $z_k^{(i,j)} \neq z_k^{(j,i)}$ e per non perdere l'informazione portata da una delle due misure conviene considerare entrambi i termini.

Per stimare lo stato della rete ora basterebbe implementare le equazioni (18)-(19), ma si può notare che la parte di dinamica dello stato é lineare e quindi il filtro risulta una versione *ibrida* tra la versione estesa e quella classica. Il passo di predizione é infatti quello di un normale filtro di *Kalman* in forma di informazione:

$$\begin{aligned} \Psi_{k|k-1} &= [\Psi_{k-1|k-1}^{-1} + Q_k]^{-1} \\ L_{k|k-1} &= \Psi_{k|k-1}\Psi_{k-1|k-1}^{-1} \\ \hat{y}_{k|k-1} &= L_{k|k-1}\hat{y}_{k-1|k-1} \end{aligned} \tag{22}$$

[8]Ovviamente, dato che nel problema di autolocalizzazione i nodi sono fermi si avrà $x_k^{(i)} = x^{(i)} \forall k$, ma si considera il caso tempo-variante per questioni di generalità.

mentre il passo di filtraggio deriva delle equazioni (19):

$$\begin{array}{lcl} \Psi_{k|k} & = & \Psi_{k-1|k-1} + |Phi_k \\ \hat{y}_{k|k} & = & \hat{y}_{k|k-1} + i_k \\ \Phi_k & = & J_x^{h^T} R_k^{-1} J_x^h \\ i_k & = & J_x^{h^T} R_k^{-1} [z_k - h(\hat{x}_{k|k-1}) + J_x^h \hat{x}_{k|k-1} \end{array} \tag{23}$$

La matrice Jacobiana J_x^h va calcolata ovviamente nel punto $\hat{x}_{k|k-1}$. Essendo $h(\cdot)$ una funzione che per ogni componente $z_k^{(i,j)}$ ha solamente quattro argomenti molte derivate parziali saranno nulle; in particolare la riga corrispondente alla componente $z_k^{(i,j)}$ risulta essere della forma:

$$J^{h(i,j)} = \left[\begin{array}{cccc} \mathbf{0} & J_{x,i}^{h(i,j)} & \mathbf{0} & J_{x,j}^{h(i,j)} \end{array} \quad \mathbf{0} \right],$$

dove i termini non nulli, relativi alle colonne corrispondenti ai nodi i e j, sono dati dalla seguente formula:

$$J_{x,i}^{h(i,j)} = \left[\begin{array}{cc} \frac{p_x^{(i)} - p_x^{(j)}}{d} & \frac{p_y^{(i)} - p_y^{(j)}}{d} \end{array} \right] = -J_{x,j}^{h(i,j)}.$$

d é ovviamente la distanza euclidea tra i due nodi, ricavabile tramite la:

$$d = \sqrt{(p_x^{(i)} - p_x^{(j)})^2 + ((p_y^{(i)} - p_y^{(j)})^2}$$

Per una trattazione più completa si rimanda a [27].

4.5.4 Risultati sperimentali

Sperimentalmente il filtro di *Kalman* si é dimostrato un buon metodo per ridurre l'errore su una stima calcolata con un qualsiasi altro metodo. Per poter portare avanti i conti, infatti, si rende necessaria una condizione iniziale ragionevolmente plausibile, e allo scopo é stata scelta una stima derivante dal metodo dei minimi quadrati. La matrice di varianza é stata scelta di forma diagonale in conseguenza dell'indipendenza degli errori di stima della posizione di nodi diversi dervante dal metodo dei minimi quadrati. Gli elementi corrispondenti alla posizione (in x e y) dei nodi ancora sono stati ovviamente posti pari a zero, mentre per i rimanenti elementi é stato adottato il criterio descritto di seguito. Tramite la (6) si calcola per ogni nodo una stima della varianza d'errore e la si spartisce in parti uguali tra varianza dell'errore sulla coordinata x e la coordinata y, assumendo quindi indipendenza statistica tra i due errori. Si costruisce in questo modo una matrice diagonale 90×90 con $2M$ elementi non nulli a due a due uguali tra loro e consecutivi[9].

[9]Chiaramente una tale matrice non é definita positiva, pertanto non é invertibile. Non sorgono tuttavia problemi nei calcoli successivi, in quanto MATLAB utilizza la pseudoinversa di *Moore-Penrose*, che é unica

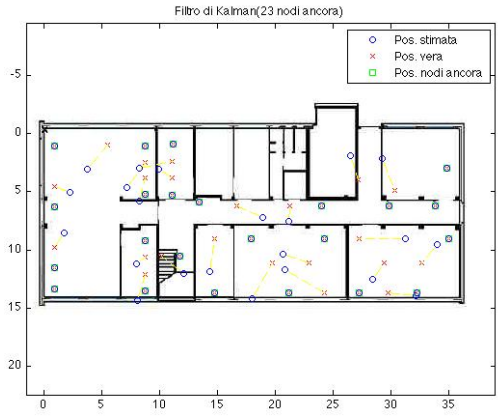


Figura 12: Mappa della rete localizzata tramite filtro di Kalman esteso

Un'altra particolarità del caso in esame é il fatto che é disponibile una singola osservazione del processo di misura, e non un intero tratto di traiettoria. Il problema é stato aggirato considerando il processo z_k come un processo costante nel tempo, usando ad ogni iterazione lo stesso valore iniziale.

Infine si é modificato il codice in modo da tener conto della conoscenza esatta della posiione dei nodi ancora: alla fine di ogni passo di filtraggio la stima della posizione dei nodi ancora (che inevitablmente si sposta leggermente dal valore corretto) é riportata deterministicamente al valore vero.

In figura (12) é riportato il risultato della stima a partire dal valore ottenuto con il metodo dei minimi quadrati riportati in figura (8).

Questa stima é ottenuta dopo 100 iterazioni delle equazioni (19)-(18) e presenta un errore medio, calcolato in modo analogo al caso precedente, di 2.5349 m; si ha in altre parole un abbassamento dell'errore di più di mezzo metro. L'andamento dell'errore medio in funzione delle iterazioni é presentato in figura (13); si noti come già dopo 50 passi il filtro sia praticamente a regime e la stima non cambi più.

In generale si può notare come il filtro di *Kalman* migliori la stima iniziale. I risultati sono riportati in tabella (1) e si riferiscono ad una stima iniziale ai minimi quadrati con gli stessi nodi ancori ancora usate nei test del capitolo precedente

Si noti che quando il numero di nodi é troppo basso il filtro peggiora addirittura la stima; probablmente la causa di ciò é dovuta alla poca informazione che portano le misure: essendoci pochi nodi ancora le misure sono in numero molto basso e per di più affette da rumore. É quindi naturale che il valore a cui converge il filtro di *Kalman* non sia una buona stima dello stato del sistema. Il fatto che addirittura peggiori la stima iniziale é probabilmente dovuto alla natura non lineare del sistema, che fa cadere i risultati teorici di convergenza verso un valore ottimo o sub-ottimo.

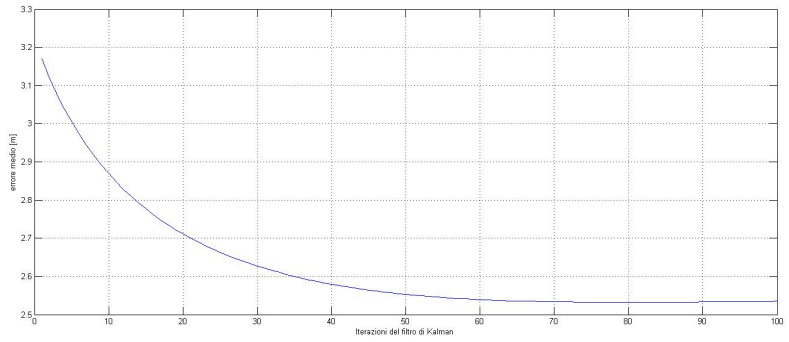


Figura 13: Andamento dell'errore medio in funzione del numero di iterazioni

	Errore medio [m]	Miglioramento netto [m]
8 nodi ancora	3.8400	-0.2892
11 nodi ancora	3.5175	0.1657
14 nodi ancora	3.0109	0.4683
17 nodi ancora	2.9049	0.5650
19 nodi ancora	2.3831	0.8465
20 nodi ancora	2.5349	0.6902
25 nodi ancora	2.5348	0.5574
30 nodi ancora	2.4240	0.3360
35 nodi ancora	2.2500	0.1635
40 nodi ancora	2.6020	-0.1833

Tabella 1: Risultati per un filtro di Kalman

Lo stesso comportamento si ha anche per un numero di nodi elevato. Qui probabilmente la causa é il rumore presente nelle misure che diventa predominante; infatti, a causa di questo elevato rumore additivo, le distanze ricavate diventano tra loro incompatibili e l'algoritmo non riesce a produrre una buona approssimazione dello stato della rete.

Nei casi intermedi invece, quando il numero di nodi ancora é ottimale, l'algoritmo migliora in maniera sostanziale la stima.

L'ultimo aspetto da analizzare riguarda la scelta delle matrici di varianza del rumore. I risultati finora esposti sono relativi a matrici della forma:

$$R = \sigma_R^2 I \quad , \quad Q = \sigma_Q^2 I,$$

con $\sigma_R^2 = 1000$ e $\sigma_Q^2 = 0.001$.

La scelta di matrici diagonali é ancora legata all'indipendenza degli errori commessi nella stima delle posizioni di nodi diversi mentre i valori dati ai due σ^2 sono stati scelti in maniera empirica. Si sono effettuate infatti varie

simulazioni (come riportato nella tabella (2) sottostante) e tali valori sono risultati ottimali.

	$\sigma_Q^2 = 0.001$	$\sigma_Q^2 = 0.01$	$\sigma_Q^2 = 0.1$	$\sigma_Q^2 = 1$	$\sigma_Q^2 = 10$	$\sigma_Q^2 = 100$
$\sigma_R^2 = 0.001$	2.7399	2.7515	2.7547	2.7552	2.7553	2.7553
$\sigma_R^2 = 0.01$	2.7126	2.7399	2.7514	2.7547	2.7552	2.7553
$\sigma_R^2 = 0.1$	2.7123	2.7126	2.7399	2.7514	2.7547	2.7552
$\sigma_R^2 = 1$	2.7007	2.7123	2.7126	2.7399	2.7514	2.7547
$\sigma_R^2 = 10$	2.6761	2.6490	2.7121	2.7126	2.7399	2.7514
$\sigma_R^2 = 100$	2.6110	2.6233	2.6806	2.7118	2.7126	2.7399
$\sigma_R^2 = 1000$	2.5349	2.5376	2.5698	2.6660	2.7115	2.7126

Tabella 2: Errore medio in funzione delle varianze di rumore

Si può suddividere la tabella in tre aree:

- $\frac{\sigma_Q^2}{\sigma_R^2} \geq 10^3$: dove l'errore medio resta costante;
- $0.1 \leq \frac{\sigma_Q^2}{\sigma_R^2} < 10^3$: dove l'errore medio dipende unicamente dal rapporto tra i due σ^2;
- $\frac{\sigma_Q^2}{\sigma_R^2} < 0.1$: dove non si nota più nessuna dipendenza elementare

In generale tuttavia si può notare come l'errore cali al calare di σ_Q^2 e all'aumentare di σ_R^2, ed é appunto questo fatto che ha portato alla scelta dei valori usati nella localizzazione della rete. Tale comportamento ha però un limite e infatti si ha la seguente situazione:

$$\begin{aligned} \sigma_R^2 = 10^4, \sigma_Q^2 = 10^{-3} \quad &\Rightarrow \quad err = 2.8653 \\ \sigma_R^2 = 10^3, \sigma_Q^2 = 10^{-4} \quad &\Rightarrow \quad err = 2.5347, \end{aligned}$$

dove appare evidente che un ulteriore aumento di σ_R^2 porta ad un aumento dell'errore medio, mentre un calo di σ_Q^2 non porta miglioramenti sostanziali.

Il diminuire dell'errore al diminuire di σ_Q^2 era facilmente intuibile, in quanto la matrice Q rappresenta, in un certo senso, quanto é grande il rumore di modello; essendo la rete fissa la dinamica corretta sarebbe $x_{k+1} = x_k$, senza rumore additico, e il diminuire della varianza avvicina il modello al caso reale. Tuttavia porre a zero tale varianza dà risultati peggiori, dal momento che per aggiornare la stima e convergere verso un valore migliore é neccessaria una componente aleatoria nella dinamica del sistema.

Il valore ottimale trovato per σ_R^2 é legato invece al rumore sull'uscita. Essendo tale rumore molto elevato e soprattutto non gaussiano é ovvio che per modellizarlo in maniera ottima tramite una variabile aleatoria gaussiana si rende necessario un elevato valore della varianza di quest'ultima.

5 Approccio Map Based

Sotto la denominazione "Map Based" sussistono tutte le metodologie di localizzazione che permettono di stimare la posizione di un nodo nella rete senza passare per la stima della sua distanza dai nodi che ricevono il suo segnale trasmesso; in termini generali si può affermare che tali tecniche utilizzano informazioni derivanti dai valori di RSSI e LQI ricevuti dalla rete, unitamente alle posizioni di alcuni nodi riferite ad un prefissato sistema di riferimento assoluto, al fine di determinare una stima più o meno approssimata della posizione dei nodi che trasmettono.

Esistono in letteratura svariati articoli che trattano di localizzazione Map Based, in larga parte vengono usati metodi non parametrici di learning a kernel [1]. Altri metodi consistono nell'interpolare un'ipersuperficie $f : \mathbb{R}^2 \rightarrow RSSI^t$ utilizzando tecniche di ottimizzazione e poi determinano la posizione dei nodi ignoti minimizzando un certo indice quadratico che tiene conto della "distanzà' tra il set di valori RSSI di un generico punto dell'ipersuperficie ed il punto da localizzare; a questo proposito si veda[2]. Alcuni metodi non usano una ipersuperficie abbinata ad un indice quadratico, bensì indici più complessi a cui non ricorre nessuna struttura di supporto operando gli stessi sui momenti campionari di ordine basso dei valori di RSSI ricevuti. L'articolo ivi citato ha anche il pregio di essere già predisposto all'utilizzo pratico, decentralizzato e robusto[3].

Tra tutti i metodi sopra elencati si é scelto di investigare maggiormente quelli a kernel, essendo state precedentemente recensite in letteratura prestazioni migliori per quest'ultimo gruppo di metodi, si veda ad esempio:[4]

5.1 Tecniche Kernel

5.1.1 Formulazione del problema

Questa sezione vuole dare una breve introduzione sulle tecniche kernel utilizzate in seguito, si é preso spunto da [10]. Estendendosi l'argomento ben oltre il caso di regressione implementato ed essendo le proprietà statistiche di tali metodi ancora oggetto di ricerca la trattazione non ha alcuna pretesa di esaustività, si rimanda a [5] e [6].

Nel problema di regressione affrontato si considerano n nodi ancora posti in locazioni spaziali bidimensionali note e m nodi target le cui coordinate sono ignote, inoltre i nodi sono in grado di comunicare tra loro con un range di connettività variabile sia topologicamente (coordinate spaziali) che nel tempo. Mediando i valori di RSSI ed LQI ricevuti da ogni nodo per un certo periodo di tempo si ottiene per il nodo i-esimo $(x, y)_i = Y_i$ posizione del nodo e $\{(RSSI, LQI)_j \; 1 \leq j \leq m + n\}_i = X_i$ valori medi dei due indici relativi

alle trasmissioni degli altri nodi della rete verso il nodo i, $1 \leq i \leq m+n$.[10]

Identifichiamo quindi secondo le convenzioni in uso in questo campo $X = \left\{X_i \subset \mathbb{R}^{2\cdot(n+m)}\right\}$ come "feature space" ed $Y = \mathbb{R}^2$ come "traget space"; le tecniche di learning utilizzano un insieme di feature con target corrispondenti noti nella fase denominata di "training" per determinare una regola che ad ogni feature associ un elemento dello spazio target, si procede indipendentemente per ogni dimensione dello spazio target: in questo caso si avranno quindi due regressioni separate ed indipendenti determinanti due modelli che restituiranno le coordinate spaziali di un nodo una volta forniti i valori di RSSI ed LQI relativi agli altri nodi della rete ricevuti dal nodo stesso. A questa fase segue quella denominata di "testing" che utilizza il modello costruito sul training set per determinare il target dei feature desiderati e quindi localizzare i nodi in questione.

5.1.2 Elementi di Statistica

L'obiettivo della regressione basata su metodi di learning é determinare una funzione che descriva il comportamento dei dati il meglio possibile. Tale funzione si pensa quindi come dipendente da una certa distribuzione di probabilità ignota in una classe di distribuzioni date (si assume di avere a che fare con processi stazionari i.i.d.), minimizzando un certo funzionale di costo (in generale quadratico) che tiene conto delle performance della funzione.

$$\hat{f}(\theta, \cdot) = argmin_\theta V(\theta) = argmin_\theta \int \|Y - f(\theta, X)\| \, dP(X, Y) \qquad (24)$$

Si noti che qui con θ non si intende una parametrizzazione (alla Fisher) della funzione in questione bensì una generica dipendenza della funzione da una serie di attributi soggetti a minimizzazione.

Per computare l'integrale é però richiesta la conoscenza esatta della distribuzione in esame, mentre ciò che si implementa in pratica é una minimizzazione su di una somma finita del comunemente denominato Rischio Empirico ovvero:

$$\frac{1}{N} \sum_{i=1}^{N} \|Y_i - f(\theta, X_i)\|$$

Risultati ricavati negli articoli sopracitati assicurano che, se le coppie (X_i, Y_i) sono realizzazioni di v.a. realizzate da un processo stazionario i.i.d, per una particolare estensione alle funzioni della legge dei grandi numeri si ha che la somma del Rischio Empirico converge in probabilità all' integrale 24. Stabilito questo non é detto che la minimizzazione su di un indice converga a

[10] Semplifichiamo la trattazione posticipando l'analisi del fatto che due nodi potrebbero non comunicare tra loro e che nessun nodo può comunicare con se stesso .

quella sull'altro, in realtà la questione é complicata e coinvolge una stima non banale della complessità strutturale del problema in esame(si rimanda alla bibliografia per ulteriori dettagli). Basti sapere che condizione necessaria e sufficente alla convergenza dei minimi é la uniforme convergenza degli indici in probabilità, che noi di seguito assumeremo.

5.1.3 Elementi di algebra dei kernel

Dato un generico insieme I si ha che H é un Reproducing Kernel Hilbert Space se:

- H é sottospazio vettoriale dello spazio vettoriale delle funzioni da I ad un generico campo $\mathbb{F}$(tipicamente $\mathbb{C}$ o $\mathbb{R}$)
- H é equipaggiato con un prodotto interno $\langle\cdot\rangle$ che lo rende spazio di Hilbert
- $\forall x \in I$ il funzionale di valutazione lineare $E_y : H \longmapsto I$ definito da $E_y(f) = f(y)$ é limitato

Sia ora H RKHS su I, poiché ogni funzionale lineare limitato é determinato in modo univoco dal prodotto interno con un vettore di in H negli spazi di Hilbert si ha che

$$\forall x \in I \; \exists! k_x \in H \text{ tale che } \forall f \in H \; f(x) = \langle f, k_x \rangle .$$

Si definisce quindi k_x Reproducing Kernel del punto x.

Non é difficile mostrare che se $\mathbb{F} = \mathbb{R}$ si ha $k_x(y) = k_y(x) = K(x,y) = \langle k_x, k_y \rangle$

Risultati ottenuti nei riferimenti sopracitati stabiliscono che in un problema formulato come in 24 la soluzione, benché potenzialmente infinito-dimensionale, giace in realtà in un insieme finito-dimensionale ed é data dall'espressione:

$$\hat{f}(x) = \sum_{i=1}^{N} a_i \cdot K(x, x_i) + b$$

Tale risultato stabilisce il fondamento teorico per l'utilizzo pratico di funzioni kernel con le proprietà di consistenza sopra discusse, quelle di più frequente utilizzo sono:

- Linear kernel $K(x,y) = x \cdot y$
- Gaussian Radial basis functions Kernel $K(x,y) = e^{-\frac{|x-y|^2}{2\cdot\sigma^2}}$
- Polynomial Kernel $K(x,y) = (x \cdot y + 1)^d$

5.1.4 Support Vector Machines

I metodi definiti "Support Vector" vengono naturalmente utilizzati per implementare l'approccio a kernel sia per quanto riguarda il caso della classificazione sia per quanto riguarda gli scopi di regressione. Lo studio di tali metodi si presenta inscindibile da argomenti avanzati sulla teoria dei kernel e notevolmente complesso, per questo motivo i metodi SVM sono stati utilizzati a scatola chiusa, senza pretendere di investigarne le proprietà generali che ci limitiamo ad elencare brevemente, si veda [8] e [7]:

- Le SVM permettono di eseguire il learning di una funzione non lineare mappando in uno spazio "feature" (indotto dalle funzoni kernel) di dimensione più alta tramite un meccanismo lineare.
- Le performance del sistema sono determinate da parametri di dimensione fissa, tale dimensione cioé non dipende dallo spazio "feature" di partenza
- Le SVM non risentono di problemi legati a minimi locali o di soluzioni troppo "sparse" (con molte cifre a zero), inoltre premettono una fase di regolazione agendo sul numero di vettori o sulla forma dei margini della superficie desiderata.

5.1.5 Osservazioni

- Il funzionale di costo presentato ha scopi puramente didattici, spesso é più utile aggiungere allo stesso un termine che tiene conto di quanto varia la funzione di regressione privilegiando funzioni più piatte; il problema assume quindi condizioni vincolate ma la trattazione non perde generalità se opportunamente modificata. Si noti che γ é un parametro di tuning, in seguito si discuterà della sua opportuna taratura.

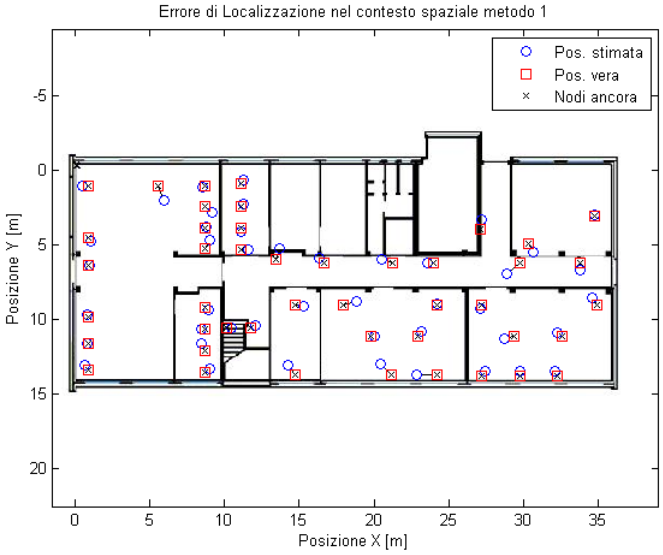
$$\tilde{f} = argmin_\theta \frac{1}{N} \sum_{i=1}^{N} \|Y_i - f(\theta, X_i)\| + \gamma \|f(\theta, X)\|$$

- Recentemente é stata distribuita in versione "Open Source" una nuova serie di techinche SVM computazionalmente molto performanti che fanno uso di metodi ai minimi quadrati, il seguente sito mette a disposizione codice matlab commentato e funzionante, [9] di seguito utilizzato per il learning di interesse.
- Come preannunciato é verosimile che alcuni nodi non siano in grado di comunicare tra loro, inoltre ogni nodo nello schema prospettato necessita anche di informazioni derivate da se stesso che ovviamente non può ottenere. Per far fronte a questa problematica si é ricorso

a dei valori di default per RSSI ed LQI: se i due nodi non riescono a comunicare si presuppone un basso LQI ed un basso RSSI; mentre intuitivamente un nodo "percepisce" se stesso con RSSI = 0 ed LQI alto, durante le prove sperimentali si sono tarati questi valori di default.

5.2 Localizzazione

Come preludio all'autolocalizzazione, che é lo scopo di questa relazione, si é scelto di testare le tecniche scelte nel più semplice caso della localizzazione; supponendo cioé nota la posizione di ciascun nodo, eseguendo quindi il training (con valori di default come spiegato) separatamente per le due dimensioni dello spazio target. Si é quindi proceduto al test della funzione trovata con i feature estratti da una serie temporale di dati diversa e successiva alla precedente. Si é scelto, in accordo con gli articoli [10], di utilizzare il kernel gaussiano. Si é poi proseguito ottimizzando i parametri del kernel ed il fattore di sconto sulla bontà del fitting dei dati γ e σ^2tramite l'algoritmo del simplesso già implementato nel toolbox usato. Il codice usato é :$localizzazione_svm_1.m$.



Il calcolo dell'errore medio di stima[11] su tutti i 45 nodi porge $0,4639$ metri, l'errore di stima più alto é di 1.3637 metri relativo al nodo #7; sono stati usati valori di RSSI ed LQI di default di $[0\,,\,0]$ per le misure che ciascun nodo dovrebbe trarre da se stesso, e di $[-88\,,\,70]$ per quanto riguarda i sensori che non riescono a scambiare informazioni. I valori dei parametri di kernel e funzione costo scelti dopo varie prove sono :

$$\gamma = [12984587.8701\,,\,1168187.2364] \qquad \sigma^2 = [270.5658\,,\,47562.9113]$$

[11] Errore di localizzazione $= \sum_{i=1}^{m} \frac{\sqrt{(x_i - \hat{x}_i) + (y_i - \hat{y}_i)^2}}{m}$

Si nota come la localizzazione presenti già errori non nulli sebbene di modesta entità, qualsiasi tecnica di autolocalizzazione non potra avere performance migliori del caso appena esaminato.

5.2.1 Approccio alternativo alla localizzazione

Si esamina di seguito un approccio alternativo proposto ai fini della localizzazione. Nel caso di localizzazione appena trattato si utilizzava come spazio feature l'insieme

$$X = \{X_i\}\,,\ X_i = \{(RSSI, LQI)_j\, 1 \leq j \leq m+n\}_i$$

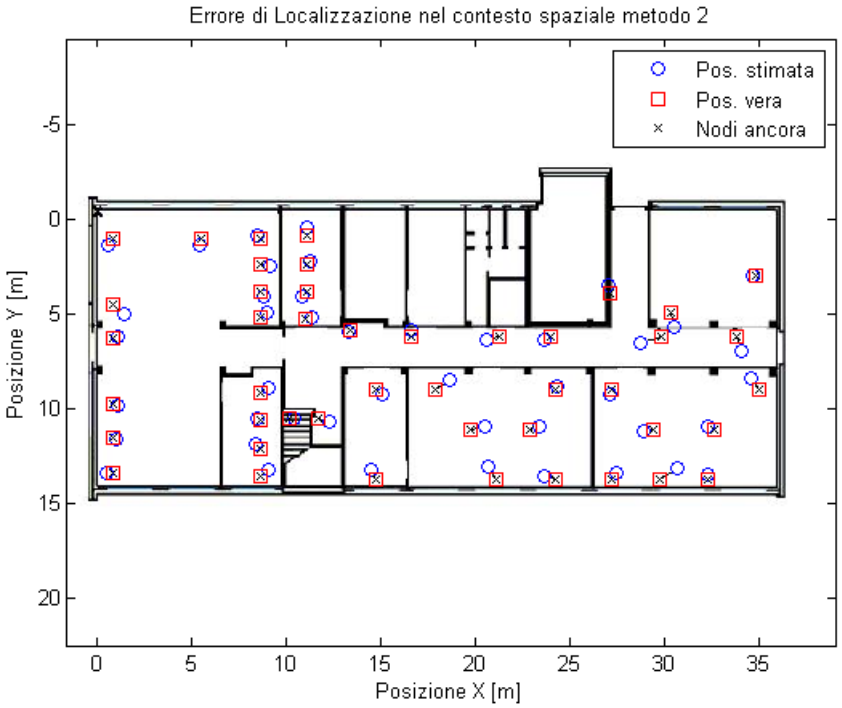
Ovvero l'insieme dei valori di RSSI ed LQI relative alle trasmissioni degli altri nodi componenti la rete ricevute dal nodo i-esimo in esame. Una scelta diversa potrebbe essere quella di considerare come spazio feature

$$X' = \left\{X'_i\right\}\,,\ X'_i = \left\{\{(RSSI, LQI)_i\}_j\ 1 \leq j \leq m+n\right\}$$

In pratica si utilizzano le misure di RSSI ed LQI inviate dal nodo i-esimo verso gli altri nodi della rete e non più quindi quelle ricevute.

L'utilizzo del seguente metodo appare giustificato sul piano teorico: infatti ci si aspetta che gli indici in esame varino in misura maggiore in relazione al comportamento del dispositivo trasmittente piuttosto che in relazione a quello del dispositivo ricevente, quindi si dovrebbero ottenere risultati migliori.

Di contro l'implementazione nei dispositivi di quest'ultimo metodo potrebbe risultare problematica, infatti l'informazione necessaria alla creazione di un feature é contenuta nei registri di tutti i nodi della rete al contrario del caso convenzionale in cui il mote i-esimo é già fornito di tutti i dati necessari alla creazione del suo feature; si presuppongono quindi maggiori problematiche ad una modifica del codice per ottenere un algoritmo distribuito. Il codice usato é $localizzazione_svm_2.m$



L'errore medio di stima sui 45 nodi é di $0,4714$ metri, quello massimo é di 1.0936 metri ed é relativo al nodo #45; sono stati usati valori di RSSI ed LQI di default di $[0\,,\,0\,,\,-88\,,\,70]$. I valori del parametri di kernel e funzione costo scelti dopo varie prove sono :

$$\gamma = [914361.7178\,,\,3386348.9165] \qquad \sigma^2 = [265.8537\,,\,47287.3073]$$

Si nota che il metodo proposto ha performance paragonabili al metodo precedentemente esaminato, compaiono errori medi leggermente più alti a fronte di errori massimi più esigui.

5.3 Autolocalizzazione

L'approccio all'autolocalizzazione di una rete di sensori di posizione fissa, stando alla formulazione del problema esposta, é immediato :

- Si sceglie una delle due rappresentazioni dello spazio feature documentate
- Si raccolgono i dati di feature relativi ai nodi di cui si conosce la posizione (nodi ancora)
- Si costruisce un modello SVM per la localizzazione a partire dai feature collezionati

- Si raccolgono i dati di feature relativi ai nodi di posizione incognita
- Si stima la posizione di questi ultimi con il modello costruito precedentemente

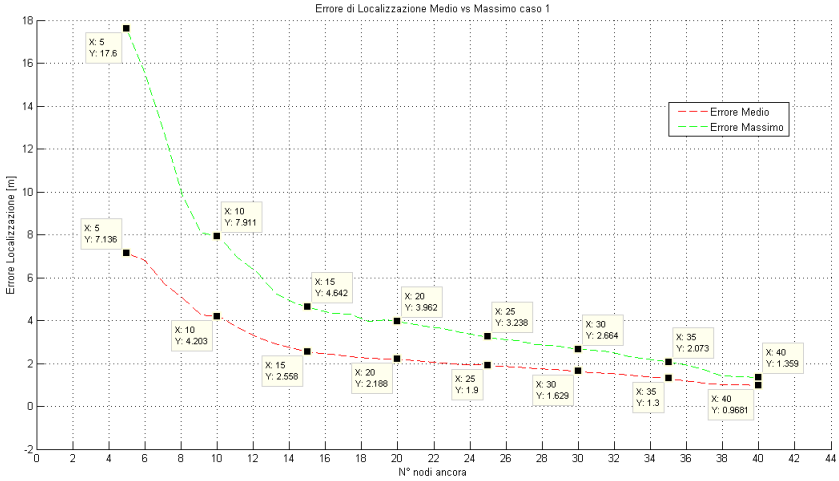
Si é proceduto a testare i metodi proposti come segue: si sono utilizzati i dati sperimentali raccolti ed elaborati come precedentemente documentato; tramite Matlab sono stati implementati i due metodi richiesti scegliendo di interpolare i dati servendosi del kernel gaussiano ed affidando, di volta in volta, a metodi di ottimizzazione a griglia il compito di determinare i parametri γ e σ^2 (in accordo con [10]); si é utilizzato un indice quadratico per la minimizzazione.

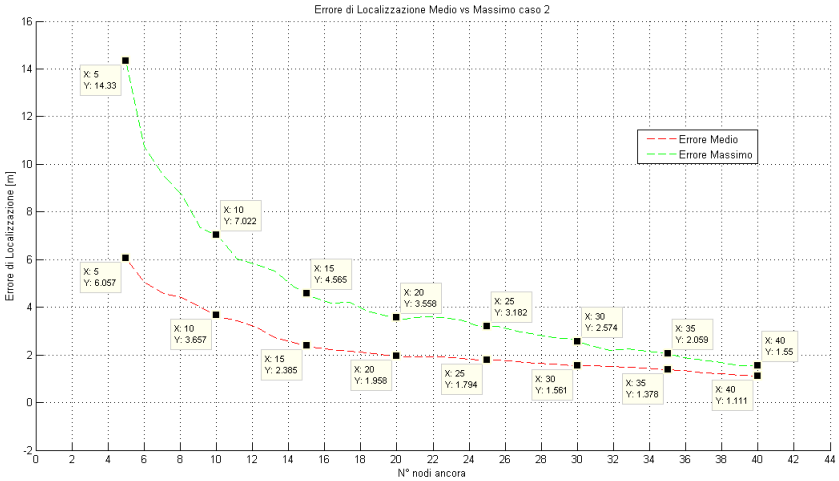
Si é proceduto a selezionare i nodi ancora da utilizzare per l'autolocalizzazione dei nodi rimanenti secondo criteri empirici. Pensando alle applicazioni infatti pare logico che una volta posizionati i nodi si determineranno le posizioni solo di alcuni nodi rispetto ad un sistema di riferimento assoluto, andando poi a determinare la posizione di altri nodi una volta che la localizzazione sia risultata non sufficentemente precisa su di essi, sulla base di osservazioni empiriche e qualitative (confrontando "ad occhio" la reale posizione dei nodi con la stima da grafico).

Si può iterare questo procedimento fino a raggiungere un errore basso a piacere purché sopra la soglia dataci dalla localizzazione, tuttavia non é chiaro a priori il comportamento dell'errore di localizzazione al variare del numero (ovviamente intero) dei nodi ancora né il variare di tale errore in funzione dei particolari nodi ancora selezionati. Qualitativamente utilizzare "griglie di nodi ancorà' più fitte porterà ad errori di localizzazione minori, inoltre nodi ancora in posizioni sfavorevoli alla trasmissione, oppure troppo vicini tra loro, potrebbero non migliorare significativamente la stima.

Per implementare questo criterio di selezione si é scelto di generare casualmente una permutazione degli ID dei nodi della rete, successivamente si autolocalizza la rete sfruttando i soli primi 5 nodi della sequenza; in seguito si aumenta l'insieme di nodi ancora di un nodo ad iterazione, scegliendo quelli con errore di localizzazione maggiore, e si autolocalizza nuovamente. Appare evidente che le osservazioni qualitative che si accennavano non possono garantire che in una applicazione effettivamente si scelga ad ogni iterazione il nodo con errore di localizzazione massimo, tuttavia ripetute osservazioni dei grafici rapportati alla pianta della disposizione dei nodi hanno portato ad un moderato ottimismo circa la capacità di un operatore di scegliere i nodi ancora adatti.

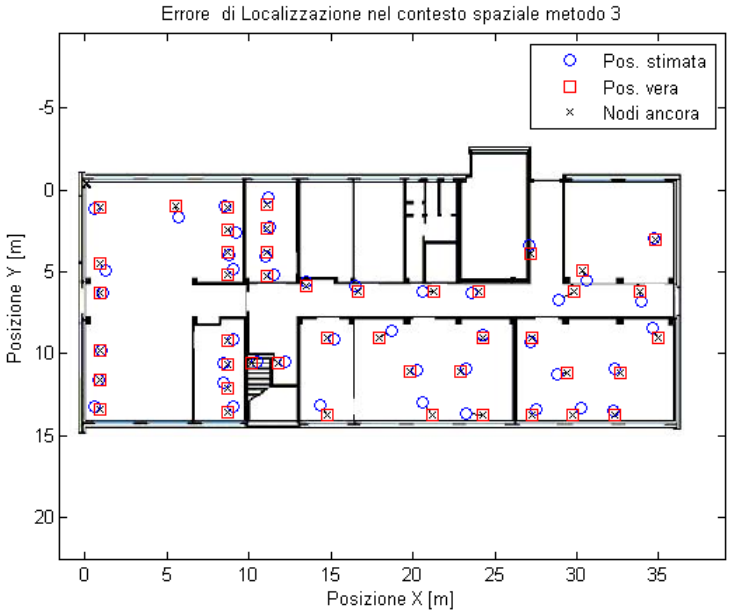
Per testare le prestazioni dell'autolocalizzazione si é ripetuta l'intera procedura sopra descritta per 10 volte mediando sull'errore di localizzazione per un numero di nodi ancora fissato, al variare dei nodi ancora.





Si nota che, come precedentemente anticipato, il metodo alternativo proposto peggiora lievemente le prestazioni sull'errore medio di localizzazione in favore ad un aumento nella precisione per quanto concerne l'errore massimo; si evince dal grafico inoltre come il metodo 2 mantenga prestazioni migliori con un esiguo numero di nodi ancora, mentre il metodo 1 ha performance migliori con un numero di nodi ancora elevato.

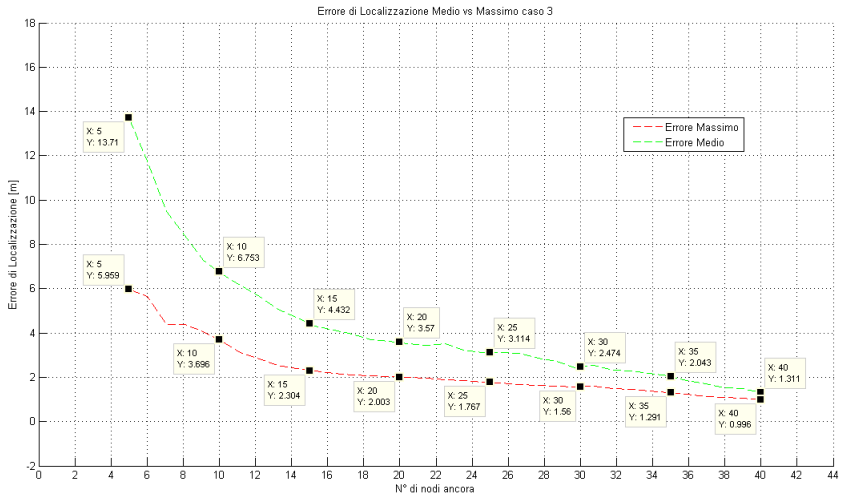
Si é tentato in seguito di ottenere prestazioni migliori unendo i due metodi sopra discussi: semplicemente il vettore feature risultante sarà la giustapposizione dei vettori feature dei due metodi sopra discussi. Sono quindi riportati i grafici relativi alla localizzazione e agli errori medi ottenuti. Il codice utilizzato è :$localizzazione_svm_3.m$



L'errore medio di stima sui 45 nodi é di $0,4473$ metri, quello massimo é di 1.0807 metri ed é relativo al nodo #45; sono stati usati valori di RSSI ed LQI di default di $[0\,,\,0\,,\,-88\,,\,70]$. I valori del parametri di kernel e funzione costo scelti dopo varie prove sono :

$$\gamma = [14718.3701\,,\,6468810.3278] \qquad \sigma^2 = [531.4288\,,\,269463.7818]$$

Si nota che l'errore medio é sui livelli del metodo 1 mentre l'errore massimo ricalca quello del metodo 2, l'algoritmo mantiene le prestazioni migliori di entrambi i metodi.



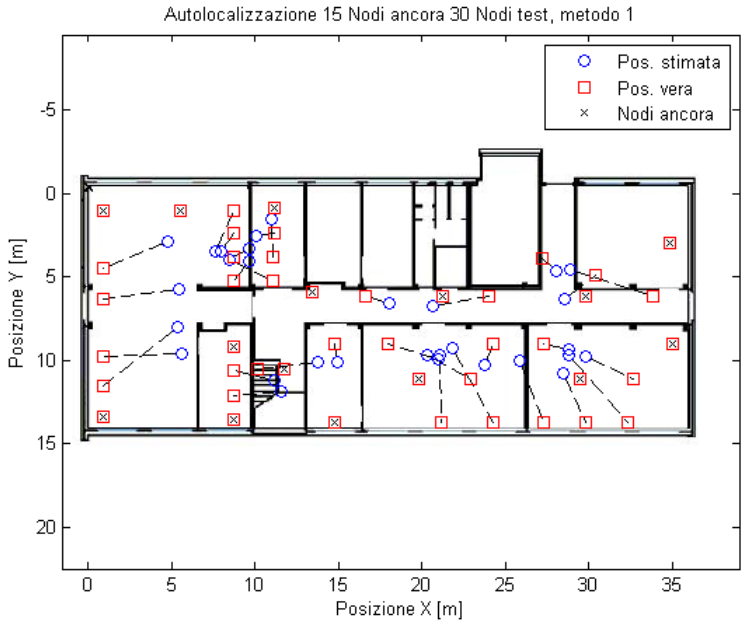
Anche da questa serie di simulazioni si evince quanto affermato prima, in particolare i valori di errore medio sono apprezzabilente più bassi di quelli ottenuti con entrambi i metodi quando i nodi ancora sono pochi (meno di 35) mentre la prestazione degrada quando i nodi ancora sono molti, in ogni caso essa non é mai peggiore del caso 2; l'errore massimo invece appare più contenuto lungo tutto il grafico.

Si può pensare di spiegare i risultati ottenuti sperimentalmente sulla base di considerazioni qualitative sullo spazio feature introdotto nei diversi casi: si richiede agli indici di RSSI ed LQI di afferire informazioni sulla posizione del nodo che trasmette a degli altri nodi (o che riceve le trasmissioni di questi ultimi). Analisi di connettività hanno tuttavia evidenziato che gli indici in esame dipendono fortemente anche dalla posizione in cui la trasmissione viene ricevuta, a causa di effetti quali le attenuazioni ed i distrurbi che i segnali wireless subiscono in ambiente indoor e la direzionalità dell'antenna dei dispositivi. Alla luce di tutto ciò il metodo proposto ottiene migliori performance dei due metodi di cui é sintesi.

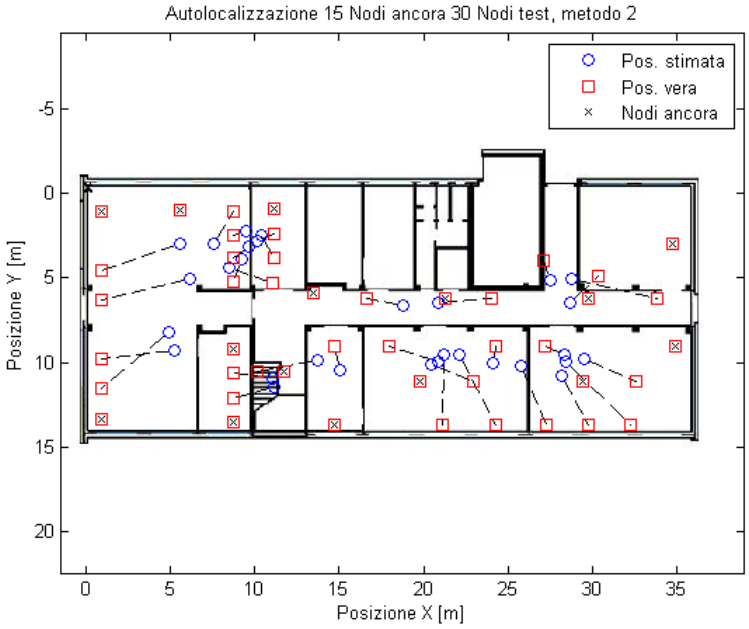
In seguito si sono scelti casualmente 15,20,25 nodi ancora; si sono testati i 3 metodi in esame sullo stesso set di nodi ancora, si sono riportati i risultati ottenuti:

15 Nodi Ancora

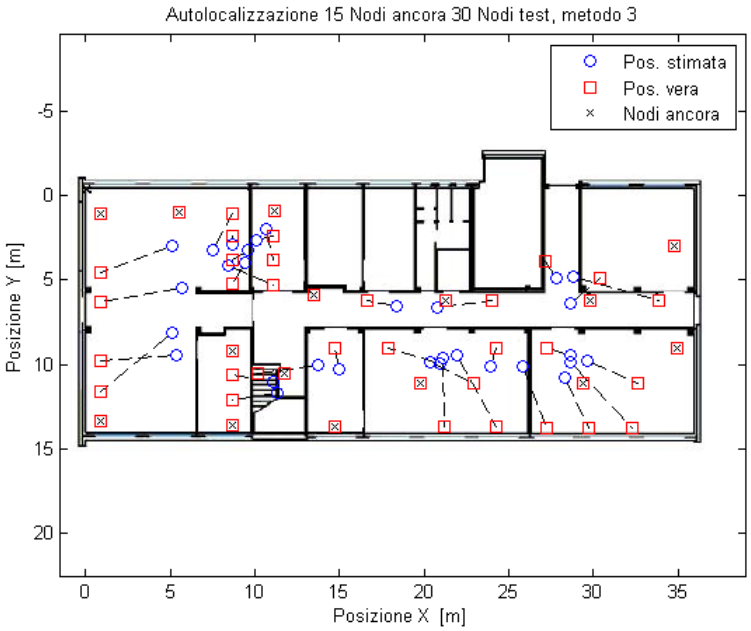
Nodi Utilizzati: $[26, 21, 27, 30, 18, 45, 1, 3, 16, 29, 42, 15, 5, 33, 41]$



Errore Medio: $2.9787m$ Errore Massimo: $5.7252m$ relativo al nodo #22



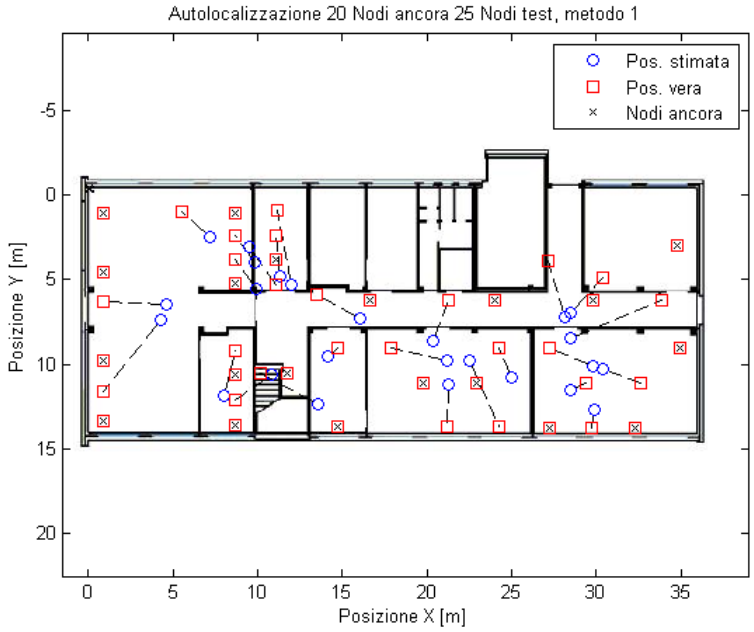
Errore Medio: 2.9197m Errore Massimo: 5.4066m relativo al nodo #24



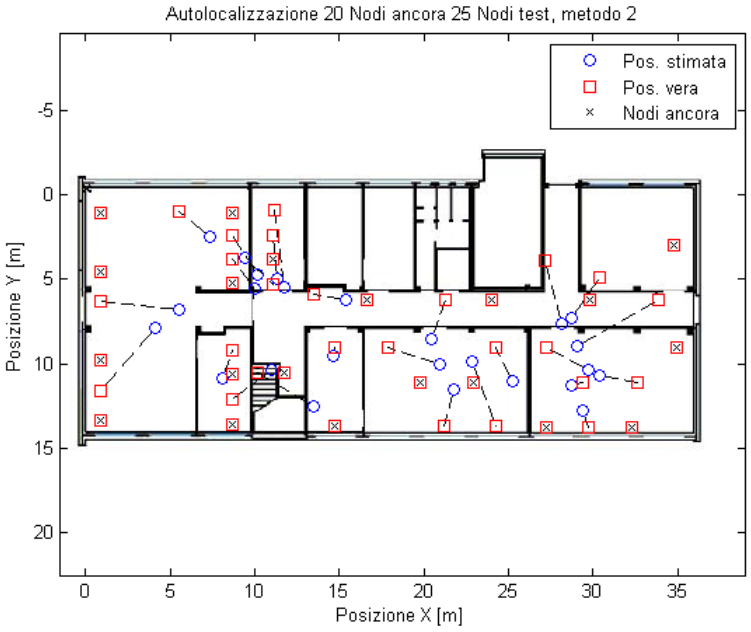
Errore Medio: 2.8943m Errore Massimo: 5.3493m relativo al nodo #22

20 Nodi Ancora

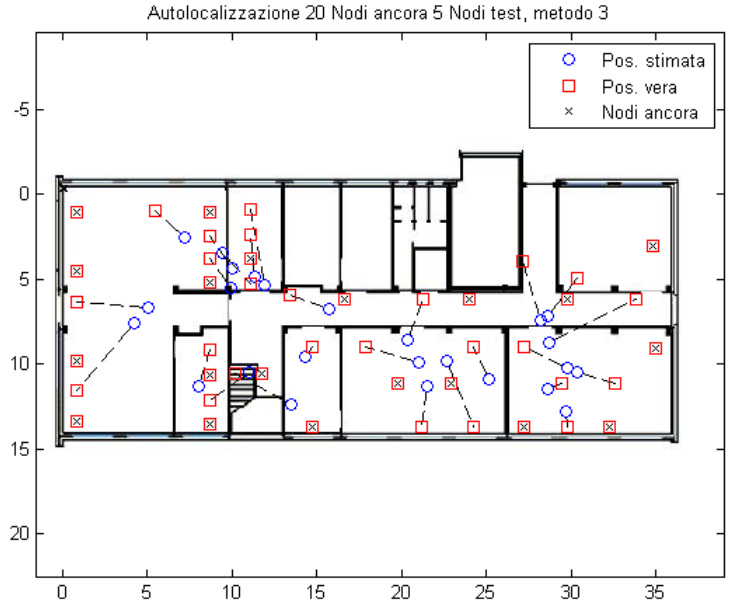
Nodi Utilizzati: [26, 25, 34, 37, 23, 31, 21, 33, 39, 29, 5, 17, 1, 2, 19, 8, 10, 15, 42, 45]



Errore Medio: 2.8776m Errore Massimo: 5.8119m relativo al nodo #43



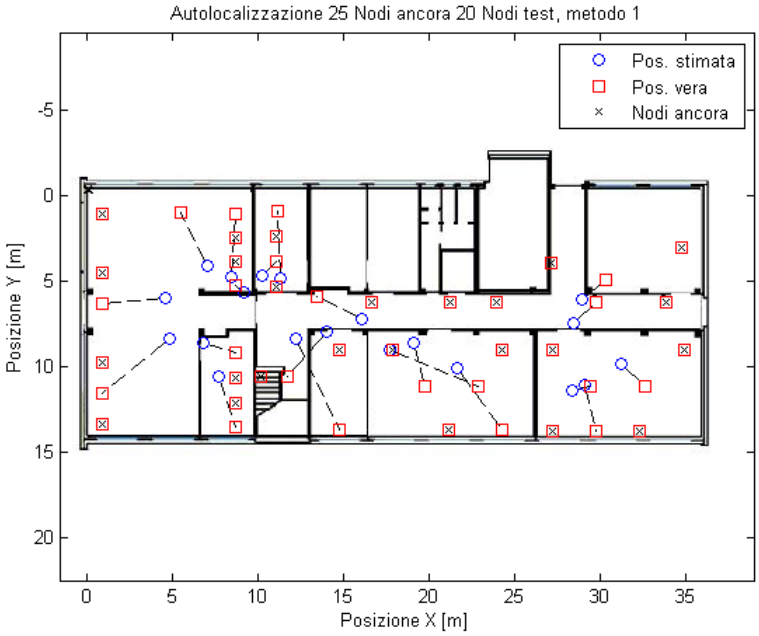
Errore Medio: 2.8382m Errore Massimo: 5.5516m relativo al nodo #43



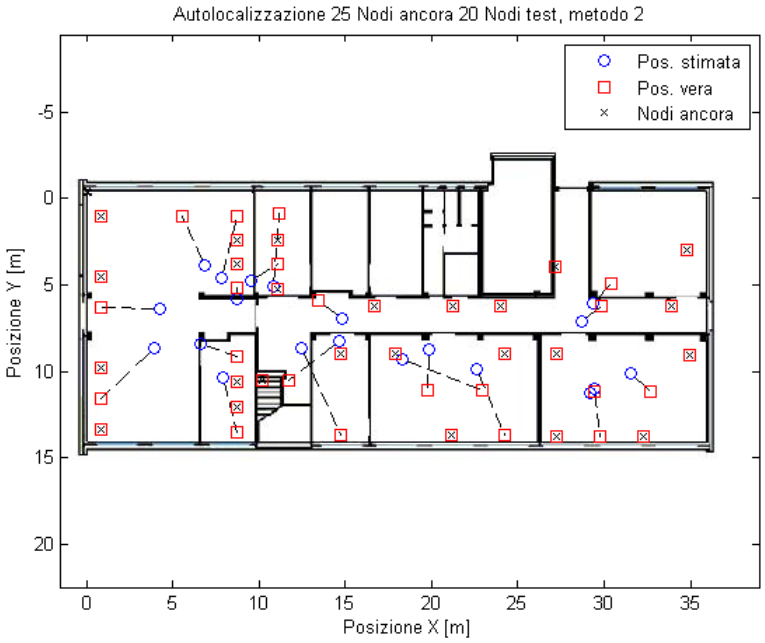
Errore Medio: 2.8457m Errore Massimo: 5.6966m relativo al nodo #43

25 Nodi Ancora

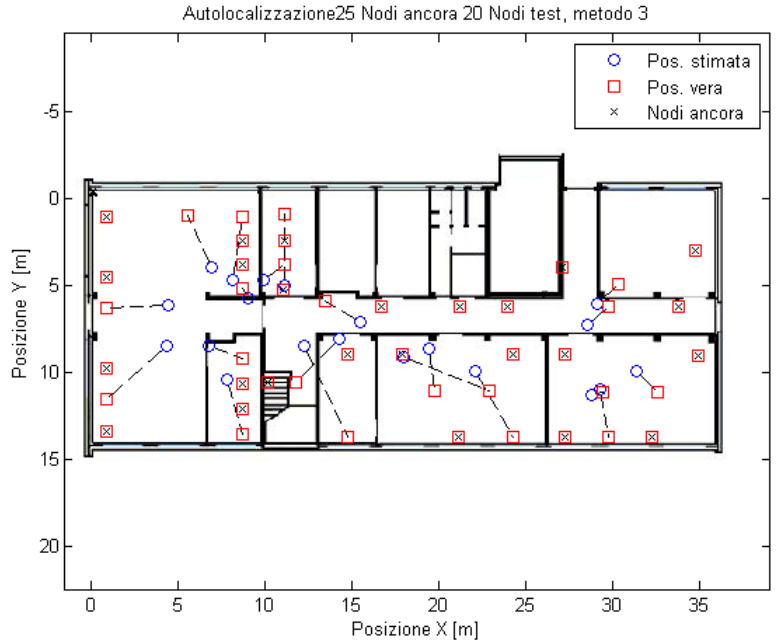
Nodi Utilizzati: [21, 23, 25, 26, 36, 35, 31, 32, 38, 40, 28, 11, 17, 6, 12, 13, 8, 10, 14, 15, 18, 19, 20, 42, 43]



Errore Medio: 3.0598m Errore Massimo: 5.8913m relativo al nodo #5



Errore Medio: 2.8292m Errore Massimo: 5.5524m relativo al nodo #5



Errore Medio: 2.9414m Errore Massimo: 5.7577m relativo al nodo #5

Dalle simulazioni sembra che il metodo 2 sia più performante del metodo 3, in disaccordo con quanto affermato. Ciò é dovuto al fatto che i nodi

ancora scelti non sono necessariamente tra quelli più performanti, difatti queste simulazioni volevano dare una idea di fondo sulle prestazioni dei tre metodi e non confrontare i tre metodi tra loro, cosa che é gia stata fatta precedentemente.

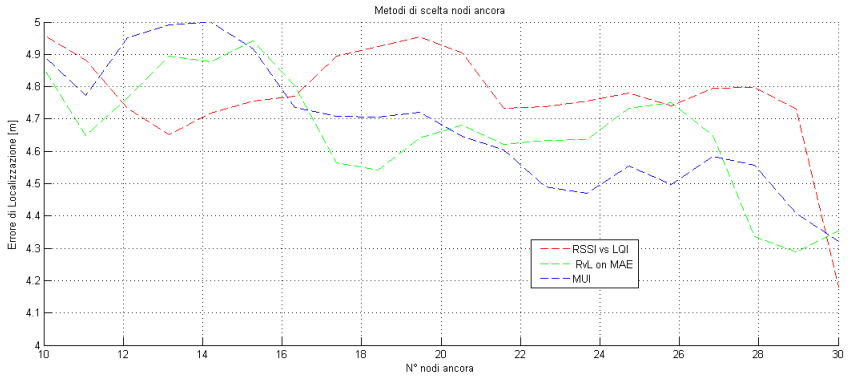
Si propone infine di eliminare il criterio di selezione "qualitativo" in precedenza introdotto per la scelta di altri nodi da localizzare, in favore di un criterio implementabile nei calcolatori.

Si sono valutate varie metodologie di seguito illustrate, purtroppo senza raggiungere le prestazioni ottenibili con il tuning manuale precedentemente descritto. Si sono graficati risultati per l'autolocalizzazione a partire da 10 nodi fino a 30 mediando su 10 casi.

Min RSSI vs LQI In questo metodo si crea per ogni nodo test un indice direttamente proporzionale alla somma di RSSI ed inversamente a quella LQI mediati tra ricevuti e trasmessi rispetto a tutti i nodi ancora della rete, con opportuni valori di default**??**. Successivamente si sceglie come nuovo nodo ancora il nodo che minimizza l'indice. Il codice utilizzato è: $localizzazione_svm_4.m$.

Min RvL on Max Anchor - Error In questo metodo si testa il modello creato anche sul set dei nodi ancora e si seleziona il nodo ancora con errore di localizzazione massimo, successivamente si determina con l'indice descritto sopra il nodo ignoto candiato a diventare nodo ancora nel prossimo step di localizzazione. Il codice utilizzato è: $localizzazione_svm_5.m$.

Max Uncensored Information In questo metodo si calcola, per ciascun nodo test, il numero di nodi test che riescono a trasmettere con esso, e successivamente si sceglie come nuovo nodo ancora il nodo test che massimizza questo indice. Il codice utilizzato è: $localizzazione_svm_6.m$.

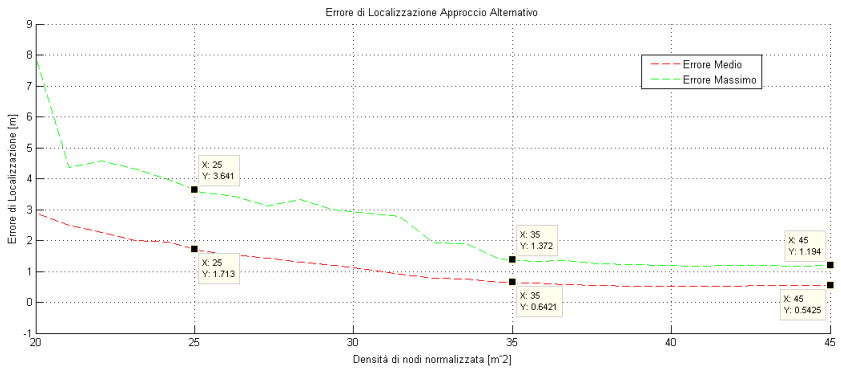


5.4 Approccio alternativo all'Autolocalizzazione

Abbiamo supposto, secondo la formulazione del problema adottata, di avere informazione posizionale solo riguardo ai punti dello spazio in cui é posizionato un nodo della rete. Questo appare restrittivo se si suppone di disporre di un nodo "mobile" (ad esempio un dispositivo palmare) in grado di trasmettere ovunque posizionato nello spazio di interesse. Si può quindi immaginare di ripetere una serie di misure identiche a quelle fino ad ora trattate per una griglia di punti nello spazio di interesse e di utilizzare poi tale set di dati al fine della localizzazione dei nodi della rete.

Tale approccio ha la caratteristica di essere perfezionabile iterativamente su basi qualititative come precedentemente argomentato nel caso di autolocalizzazione trattato e di ottenere un modello di stima della posizione non dipendente dalla posizione dei nodi della rete: infatti poiché é possibile campionare i features in griglia in modo arbitrario é teoricamente possibile sebbene oneroso in termini di realizzazione manuale ottenere precisioni elevate anche con un numero ridotto di nodi ancora aumentando la "risoluzione" della griglia in cui si campiona.

Non disponendo di dati sperimentali su cui tarare l'approccio formulato si é scelto di localizzare 20 nodi a caso della rete (scelti una volta per tutte per simulazione, si é utilizzato il metodo misto descritto in preceden) variando il numero di punti in cui si campiona lo spazio feature (i punti coincidono necessariamente con i nodi della rete da cui le misure sono state rilevate[12]). I risultati sono mostrati in funzione della densità di nodi normalizzata per area dell'edificio unitaria $[\frac{n^\circ\, nodi}{Area} \sim m^{-2}]$.



I risultati sono incoraggianti, sebbene sia doveroso notare che potrebbe apparire un controsenso localizzare 45 posizioni con il palmare al fine di determinarne poi 20. Infatti assume fondamentale importanza il contesto della procedura di localizzazione o autolocalizzazione adottata: autolocalizzare una

[12] Si é scelto di non interpolare le misure in quanto l'informazione portata da una ipotetica interpolazione non avrebbe potuto essere maggiore di quella contenuta nel set di dati stesso.

rete con un campionamento fitto per poi usare le informazioni ottenute per rilocalizzare la rete quando necessario può risultare più o meno conveniente a seconda delle particolari applicazioni.

6 Autolocalizzazione distribuita: approccio ARAP^2

Nei molteplici casi in cui una WSN debba soddisfare esigenze di robustezza ai guasti, scalabilità, e riduzione dell'overhead di comunicazione, si rende necessaria un'implementazione distribuita, in cui cioé ogni nodo della rete ragiona *autonomamente* e *localmente*.

La strategia di seguito adottata per la simulazione é stata denominata ARAP^2: essa infatti implementa l'algoritmo *As-Rigid-As-Possible* [12] per l'autolocalizzazione di una rete di sensori, rispettando delle condizioni di simulazione il più vicine possibile alla realtà (*As-Real-As-Possible*), sia nella modellizzazione del canale, sia nella comunicazione tra nodi. In letteratura non é ancora presente un esperimento simile, che faccia uso di dati reali e che trasponga l'algoritmo ad un grafo orientato; nella pubblicazione originaria, le connessioni tra i nodi erano simmetriche, e il rumore di misura simulato era uniformemente distribuito nell'intervallo $\pm 10\%$ della distanza effettiva.

La sezione 6.1 descrive genericamente il funzionamento dell'algoritmo, approfondito nella sezione 6.2. Tutte le problematiche, e le relative soluzioni, connesse all'applicazione dell'algoritmo ad un dataset non simulato sono trattate in 6.3, mentre i risultati ottenuti sono mostrati in 6.4. Infine, nella sezione 7.3 sono presenti ulteriori spunti e soluzioni che per mancanza di tempo non é stato possibile implementare.

6.1 Panoramica generale

In generale, il problema di autolocalizzazione di una rete di sensori espressa dal grafo orientato[13] $\vec{G} = (V = \{1, \ldots, n\}, \vec{E})$, e di cui si conoscano le distanze (rumorose) d_{ij}, é caratterizzabile formalmente come la minimizzazione di una *funzione stress*:

$$Stress(p_1, \ldots, p_n) = \sum_{(i,j) \in \vec{E}} \left(\|p_i - p_j\| - d_{ij} \right)^2 . \tag{25}$$

Questa funzione, quando non siano note tutte le distanze tra i nodi, non é convessa, e la ricerca globale di una soluzione ottima può diventare estremamente inefficiente all'aumentare del numero dei nodi.

[13] In letteratura si opera quasi sempre su grafi non orientati: come spiegato in 6.3, nel caso reale questa ipotesi spesso non può essere garantita.

L'algoritmo utilizzato sfrutta la strategia PATCHWORK [13], che consiste nella riduzione dell'intero problema di autolocalizzazione a molteplici sottoproblemi: a partire da una matrice di interdistanze locali, eventualmente completata da stime nel caso sia sparsa, ogni nodo esprime in un proprio sistema di coordinate la posizione dei suoi vicini, generando così un *patch* centrato attorno a sé. Questi patch vengono "cuciti" assieme, mappandoli in un sistema di riferimento globale attraverso trasformazioni affini (AAAP); il modello così ottenuto viene utilizzato come punto di partenza per l'applicazione iterativa di un metodo dei minimi quadrati alternato in due fasi, che ripete la localizzazione utilizzando solo trasformazioni rigide[14] (ARAP) per preservare le relazioni locali tra nodi vicini.

Poiché il processo cosiddetto di *stitching* é limitato solo ad un nodo e i suoi diretti vicini, non si verificano fenomeni di propagazione dell'errore come avviene per altri algoritmi incrementali, e anzi viene incrementata la resistenza al rumore di misura. Alle fasi AAAP e ARAP é associata la risoluzione di due sistemi di equazioni (rispettivamente, lineare e non lineare), che tuttavia risultano sparsi: ciò permette di distribuire il calcolo tra i nodi della rete, fino alla loro convergenza.

6.2 Algoritmo: As-Rigid-As-Possible

6.2.1 Condizioni di localizzabilità

Perché un'algoritmo di autolocalizzazione possa avere successo, é necessario che il grafo $\vec{G}$ su cui si opera possieda certe proprietà [14]: la più ovvia é la *3-connettività* di ogni nodo relativa agli archi entranti da vicini non allineati, che garantisce la rigidità del grafo e impedisce ambiguità di flip. In alcuni casi, ipotizzando di operare su un disk graph[15], é possibile risolvere in un'ottica globale tali ambiguità, anche se il punto in questione é solo 2-connesso.

Un'altra proprietà essenziale é la *rigidità ridondante*: ovvero, il grafo deve mantenere la semplice rigidità (sono ammessi i flip) anche dopo la rimozione di un qualsiasi arco. Nello spazio bidimensionale, queste due proprietà si riassumono nella *rigidità globale generica* [15] (da noi estesa intuitivamente ai grafi orientati, ma non formalizzata), la quale, assieme alla presenza di almeno tre nodi ancora con archi uscenti, garantisce la localizzabilità della rete.

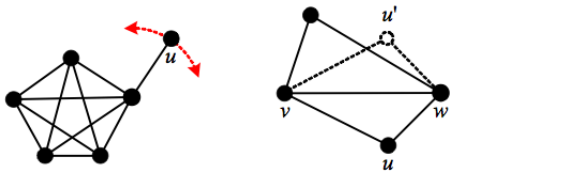
Per verificare tali condizioni, anche qui é spesso necessaria un'analisi globale

[14] Cioé traslazioni, rotazioni, e ribaltamenti.

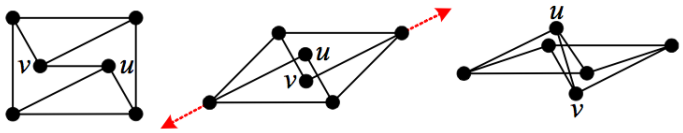
[15] In un disk graph generico (orientato), se la distanza tra due nodi d_{ij} é inferiore al maggiore tra i raggi associati ad ogni nodo (cioé se $d_{ij} \leq \max\{r_i, r_j\}$), allora questi sono (fortemente) connessi.

e non distribuita: localmente, é possibile individuare componenti trilaterализzabili o WHEEL-lateralizzabili [16], ma non sempre gli algoritmi associati riescono a classificare la totalità della rete, e i grafi correttamente individuabili costituiscono un sottoinsieme di tutti quelli localizzabili.

Di seguito non verranno approfondite ulteriormente le questioni di localizzabilità della rete e localizzabilità dei nodi, per le quali si rimanda principalmente al lavoro di Yang e Liu [16][17]. La rete $\vec{G}$ in esame si suppone quindi genericamente globalmente rigida; nella sezione 7.3 si teorizza una possibile soluzione per alcuni casi di connettività scarsa o non simmetrica, che fa uso di un nodo mobile collocato appropriatamente.



(a) Non rigido (b) Non 3-connesso



(c) Non ridondantemente rigido

Figura 14: Realizzazioni non uniche

6.2.2 Star

In una rete di sensori espressa sotto forma di grafo orientato, ogni nodo i é rappresentabile da un grafo a stella di dimensione S_i, cioé da un albero avente S_i foglie e la radice i come unico nodo interno. L'insieme delle foglie viene definito *neighbourhood*, e vi appartengono tutti i nodi di cui i può misurare la distanza d_{ji}: in altre parole, poiché tale misura é possibile solo ricevendo un qualsivoglia segnale, la *star* $S(i)$ é formata dall'insieme degli archi di $\vec{E}$ entranti in i e pesati con d_{ji}, e dei nodi associati:

$$S(i) = \{j \,:\, (j,i) \in \vec{E}\} \cup \{i\} \,. \tag{26}$$

6.2.3 Patch

Partendo dalle informazioni contenute nella star d'interesse $S(i)$ e in quelle dei suoi vicini $S(j)$, dove $j = \{1, \ldots, S_i\} \in S(i)$, si cerca quindi di identificare

una localizzazione univoca (a meno di una trasformazione rigida) dei vertici i e j. Ciò porta alla creazione di un patch non orientato e completo $P(i)$, derivato parzialmente da $\vec{G}$, ovvero dell'espressione delle posizioni dei nodi della stella in un sistema di coordinate locali indotte $Q_i = \{q_i, q_{j_1}, \ldots, q_{j_{S_i}}\}$ avente origine in i.
Il procedimento costruttivo della localizzazione Q_i é diviso in tre fasi, adattate dal metodo PATCHWORK [13]:

1. *Completamento del patch.* Poiché le informazioni contenute in $S(i)$ non bastano per garantirne l'unicità in 2 dimensioni (si pensi al caso degenere in cui tutti i punti giacciono sulla stessa retta), é necessario completare la matrice di interdistanze associata al patch aggiungendo gli archi di lunghezza:

$$\text{mean}(d_{jk}, d_{kj}) = \begin{cases} \dfrac{d_{jk} + d_{kj}}{2}, & (j,k) \in S(k), (k,j) \in S(j) \\ d_{jk}, & (k,j) \notin S(j) \\ d_{kj}, & (j,k) \notin S(k) \end{cases} , \{j,k\} \in S(i). \tag{27}$$

   La funzione presentata esegue la media tra le misure di due archi quando questi siano entrambi presenti in $\vec{E}$ e connettano gli stessi vertici, altrimenti restituisce l'unico arco esistente tra i due nodi interessati. A livello grafico, ciò equivale ad aggiungere il terzo lato del triangolo $\widehat{ijk}$, i cui lati appartengano a E non orientato.
   Spesso questo completamento non é sufficiente, e la matrice é ancora sparsa: si stimano pertanto le distanze rimanenti con la media di un upper bound B_{jk}, derivato dalla disuguaglianza triangolare, e di un lower bound b_{jk}, derivato dall'ipotesi di operare su un disk graph[16]:

$$B_{jk} = \min_{h:(j,h)\in E,(h,k)\in E} \{d_{jh} + d_{hk}\} , \tag{28}$$

$$b_{jk} = \max \left\{ \max_h \{d_{jh}\}, \max_h \{d_{hk}\} \right\} . \tag{29}$$

2. *Classical MDS.* Una volta completato il grafo $P(i)$ (che é pertanto una clique di G non orientato), la localizzazione é affidata al Multi-Dimensional Scaling classico [18], calcolato in MATLAB e troncato alla sua variante planare dalle istruzioni:

```
MDSembedding = cmdscale(fill_dist);
pos = MDSembedding(:,1:2);
```
(30)

   dove `fill_dist` é la matrice derivata al punto precedente. Operando localmente, la complessità di questa fase é $O(mS_i^2)$, dove $m \leq S_i$ é il

[16] Questa ipotesi, che migliora sensibilmente la precisione dell'algoritmo, non é generalmente verificata: si pensi alla presenza di un ostacolo tra due nodi vicini che impedisca loro di comunicare. In questi casi, solo l'upper bound viene usato come stima.

numero di dimensioni a cui viene innalzato il problema di localizzazione Q_i: l'indipendenza dalla cardinalità dei vertici del grafo G allevia notevolmente l'onerosità computazionale, ma potrebbe lo stesso rivelarsi un potenziale collo di bottiglia dell'intero procedimento, specie nel caso di nodi molto connessi e con ridotta capacità di calcolo[17].

3. *Stress majorization.* Quest'ultimo passaggio cerca di aumentare il fitting del modello ottenuto, usando solo le distanze di misura nota. Ciò consiste nell'applicazione iterativa della seguente regola d'aggiornamento, per ogni $j \in S(i)$:

$$q_j \leftarrow \frac{1}{S_i} \sum_{k \in S(i)} \left[q_k + d_{jk}(q_j - q_k) \operatorname{inv}(\|q_j - q_k\|) \right] , \tag{31}$$

dove

$$\operatorname{inv}(x) = \begin{cases} 1/x, & x \neq 0 \\ 0, & x = 0 \end{cases} . \tag{32}$$

Si noti che questo processo modifica anche la posizione q_i del nodo centrale, che non sarà più pari a $[0,0]^T$; essendo interessati a mantenere questa proprietà per facilitare calcoli successivi, si é operata una traslazione del tipo:

$$\bar{Q}_i = Q_i - q_i . \tag{33}$$

6.2.4 Stitch

Assumendo di aver correttamente localizzato i vari patch $P(i)$ a meno di una trasformazione rigida, la loro posizione espressa in coordinate assolute é calcolabile con:

$$P_i = \mathbf{R}_i \bar{Q}_i + \mathbf{T}_i , \tag{34}$$

dove $\mathbf{R}_i \in \mathbb{R}^{2\times 2}$ é una matrice di trasformazione rigida e $\mathbf{T}_i$ é un vettore di traslazione, che si può eliminare assumendo di operare solo sulle differenze vettoriali tra coordinate locali e globali.
La localizzazione della rete diventa così un problema di ottimizzazione *simultanea* per ogni $\mathbf{R}_i$ e per P, vettore delle posizioni assolute indotto dai vari P_i, in cui si cerca di minimizzare la cosiddetta *energia ARAP*:

$$(P, \mathbf{R}_1, \ldots, \mathbf{R}_N) = \arg \min_{P_1, \mathbf{R}_1, \ldots, P_N, \mathbf{R}_N} \left\{ \sum_{i=1}^{N} \left\| \bar{P}_i - \mathbf{R}_i \bar{Q}_i \right\|_F^2 : \mathbf{R}_i^T \mathbf{R}_i = I \right\} , \tag{35}$$

[17] Inoltre, non sono attualmente conosciute implementazioni del MDS in linguaggio nesC.

con ovvia interpretazione di $\bar{P}_i$ come localizzazione centrata in p_i, e dove $\|\cdot\|_F$ é la norma matriciale secondo Frobenius:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}\ . \tag{36}$$

La risoluzione di (35) non é immediata: sono dunque necessarie due osservazioni, per i cui dettagli si rimanda a [12].

1. Date le posizioni P_i e Q_i, é possibile calcolare localmente ogni matrice $\mathbf{R}_i$: si tratta di una versione del problema di rotazione ortogonale di Procruste[18], risolvibile in forma chiusa da:

$$\mathbf{R}_i = V_i U_i^T\ , \tag{37}$$

dove U_i e V_i sono le componenti ortogonali della decomposizione a valori singolari (SVD) di $\bar{Q}_i \bar{P}_i^T$:

$$\bar{Q}_i \bar{P}_i^T = U_i \Sigma V_i^T\ . \tag{38}$$

L'algoritmo risolutivo della SVD, pur avendo complessità $O(4m^2n + 8mn^2 + 9n^3)$ per una generica matrice $\mathbb{R}^{m\times n}$, é qui applicato sul prodotto tra $\bar{P}_i, \bar{Q}_i \in \mathbb{R}^{2\times S_i}$, cioé su $\mathbb{R}^{2\times 2}$, e non comporta una significativa onerosità computazionale[19].

2. Date le trasformazioni $\mathbf{R}_i$, azzerando i gradienti del problema globale di ottimizzazione quadratica che produce P, si ottiene un sistema $N \times N$ lineare formato dalle equazioni:

$$\begin{aligned}\sum_{j\in S(i)}(p_i - p_j) &= \frac{1}{2}\sum_{j\in S(i)}[\mathbf{R}_j(q_{j(i)} - q_j) + \mathbf{R}_i(q_i - q_{i(j)})] \\ &= \frac{1}{2}\sum_{j\in S(i)}[\mathbf{R}_j\bar{q}_{j(i)} - \mathbf{R}_i\bar{q}_{i(j)})]\ , \qquad i = 1,\dots,N\ ,\end{aligned} \tag{39}$$

dove é stata sfruttata la traslazione (33), e dove per $\bar{q}_{j(i)}$ si intende la localizzazione del nodo j nelle coordinate locali $\bar{Q}_i$ indotte da i.

Questi due risultati suggeriscono un approccio iterativo che usi il metodo Alternating Least-Squares (ALS), o metodo *locale-globale*, suddiviso in due fasi: nella prima fase *locale*, P si suppone fissato, e ogni nodo ricava la sua $\mathbf{R}_i$ risolvendo la propria equazione (37); nella seconda fase *globale*, si suppongono fissate tutte le $\mathbf{R}_i$, e si ricava P dalla (39).

[18] In inglese é *rotation orthogonal procrustes problem*, o ROPP

[19] Permane comunque il problema di trovarne un'implementazione in nesC.

In realtà, essendo il sistema $N \times N$ un sistema sparso, ed essendo la matrice associata al sistema dotata di valori non nulli solo tra elementi connessi, il calcolo della posizione p_i di ogni nodo é legato solo a quello delle posizioni p_j dei nodi $j \in S(i)$. é quindi possibile utilizzare una versione completamente distribuita dell'algoritmo iterativo di Successive Over-Relaxation [19]:

$$A\mathbf{x} = \mathbf{b}$$
$$\Downarrow$$
$$x_i^{(k+1)} = (1-w)x_i^{(k)} + \frac{w}{a_{ii}}(b_i - \sum_{j>i} a_{ij}x_j^{(k)} - \sum_{j<i} a_{ij}x_j^{(k+1)}) \; . \tag{40}$$

Il fattore di rilassamento é stato scelto come $w = 1$, ottenendo quindi il più semplice algoritmo di Gauss-Seidel. Si é preferito comunque lasciarlo parametrizzabile, nel caso in cui si desideri manipolare la velocità di convergenza, o controllare eventuali effetti divergenti; operando su matrici simmetriche e definite positive[20], la convergenza può essere dimostrata per valori di $0 < w < 2$.
Inoltre, l'energia ARAP non aumenta durante le due fasi, ed é pertanto garantito che essa raggiunga un minimo locale.

6.2.5 Starting map

L'algoritmo ARAP, per via della sua particolare natura, necessita di una localizzazione iniziale nota, che viene perfezionata iterandovi sopra le due fasi precedentemente descritte: questo punto di partenza é fornito dall'algoritmo As-Affine-As-Possible (AAAP), che opera un rilassamento su $\mathbf{R}_i$ ammettendo le famiglie delle trasformazioni affini.
Ciò produce una localizzazione di G unica, a meno di una trasformazione della stessa famiglia di $\mathbf{R}_t$, cioé anch'essa affine: per evitare che il sistema diventi omogeneo e ammetta la soluzione triviale $P = 0$ per tutti i punti, é necessario fissare un numero minimo di $N_a \geq 3$ ancore non allineate, evitando così degenerazioni con perdita di dimensionalità. Parametrizzando poi $\mathbf{R}_i$ come:

$$\mathbf{R}_i = \begin{pmatrix} a_i & c_i \\ b_i & d_i \end{pmatrix}, \tag{41}$$

si giunge ad una formulazione globale, lineare ed omogenea, del processo di localizzazione:

$$p_j - p_i - \mathbf{R}_i(q_j - q_i) = 0 \;\Rightarrow\; p_i + \mathbf{R}_i\bar{q}_j = p_j \;\Rightarrow\; \begin{cases} p_i^x + a_i\bar{q}_j^x + c_i\bar{q}_j^y = p_j^x \\ p_i^y + b_i\bar{q}_j^y + d_i\bar{q}_j^y = p_j^y \end{cases}, \tag{42}$$

[20] La simmetria della matrice al primo membro di (39) é associata alla simmetria della matrice di connettività: per grafi orientati essa non é ovvia, e si é quindi dovuti intervenire come spiegato in 6.3.

dove $i = 1, \ldots, N,\ j \in S(i)$. Il numero delle incognite é 6, legando la posizione p_i a quella di altri 3 nodi non allineati $j, k, l \in S(i)$: si definiscono così le componenti A e $\mathbf{b}$ del sistema lineare locale che sarà quindi risolto sempre tramite la Successive Over-Relaxation[21]:

$$\begin{bmatrix} 1 & 0 & \bar{q}_j^x & 0 & \bar{q}_j^y & 0 \\ 0 & 1 & 0 & \bar{q}_j^x & 0 & \bar{q}_j^y \\ 1 & 0 & \bar{q}_k^x & 0 & \bar{q}_k^y & 0 \\ 0 & 1 & 0 & \bar{q}_k^x & 0 & \bar{q}_k^y \\ 1 & 0 & \bar{q}_k^x & 0 & \bar{q}_l^y & 0 \\ 0 & 1 & 0 & \bar{q}_k^x & 0 & \bar{q}_l^y \end{bmatrix} \begin{bmatrix} p_i^x \\ p_i^y \\ a_i \\ b_i \\ c_i \\ d_i \end{bmatrix} = \begin{bmatrix} p_j^x \\ p_j^y \\ p_k^x \\ p_k^y \\ p_l^x \\ p_l^y \end{bmatrix}, \tag{43}$$

dove $\bar{q}_j, \bar{q}_k, \bar{q}_l \in Q_i$ sono espresse secondo le coordinate locali indotte da i.

Il successo della risoluzione distribuita di (43) dipende dalla 3-connettività di $\vec{G}$ e dal posizionamento delle ancore: ricordiamo che un nodo localizzabile in un grafo orientato deve avere 3 cammini disgiunti *entranti* da altrettanti nodi di posizione nota. Ciò porta ad un'importante conclusione: per evitare comunicazioni multi-hop, devono esistere almeno 3 ancore posizionate vicine tra loro, in modo da individuare dei nodi le cui star "ricevano" un arco da ciascuna di esse; tali nodi possono così calcolare la loro posizione e fungere anch'essi da ancore fittizie per un'iterazione successiva dello stesso procedimento.
Come diretta conseguenza, se dopo un tempo proporzionale a $N - N_a$ (il caso peggiore, con un solo aggiornamento per iterazione) non é stata ancora completata questa propagazione d'informazione, allora quella specifica scelta delle N_a ancore rende il grafo $\vec{G}$ non *1-hop AAAP localizzabile*. Una buona regola euristica é quella di cercare una posizione centrale, con il maggior numero di archi uscenti; ulteriori ancore collocate in zone a scarsa connettività possono migliorare o velocizzare la localizzazione.

6.3 Implementazione: As-Real-As-Possible

6.3.1 Star

Essendo le star $S(i)$ i tasselli fondamentali per il processo di ricomposizione della rete, é stata data particolare importanza all'analisi dei fattori che ne influenzano la costruzione; poiché una stella (orientata, pesata) é interamente definita dal numero e dalla distanza dei collegamenti del centro i con i vicini, ci si riporta ad un problema di localizzazione in cui si é interessati ad avere delle misure precise e di cui si possa avere una buona confidenza. I parametri da valutare sono:

[21] In realtà, A non é simmetrica definita positiva: si risolve invece il problema $A^T A\mathbf{x} = A^T\mathbf{b}$, la cui convergenza é assicurata.

- *Modello di canale.* In un contesto reale, la creazione e l'impiego di un modello di propagazione che ben rappresenti l'area operativa sono vincolati da due condizioni opposte: più precisa é la funzione che traduce valori di RSSI in distanze, più complesso, oneroso e localizzato deve essere il suo calcolo. é palese come, in un ambiente ricco di multipath e ostacoli (che per giunta indeboliscono, ma non impediscono, le trasmissioni) come quello in cui é stato raccolto il dataset, un unico modello di canale non possa interpretare correttamente lo stesso valore di RSSI proveniente da due percorsi differenti, e sia perciò assai impreciso.

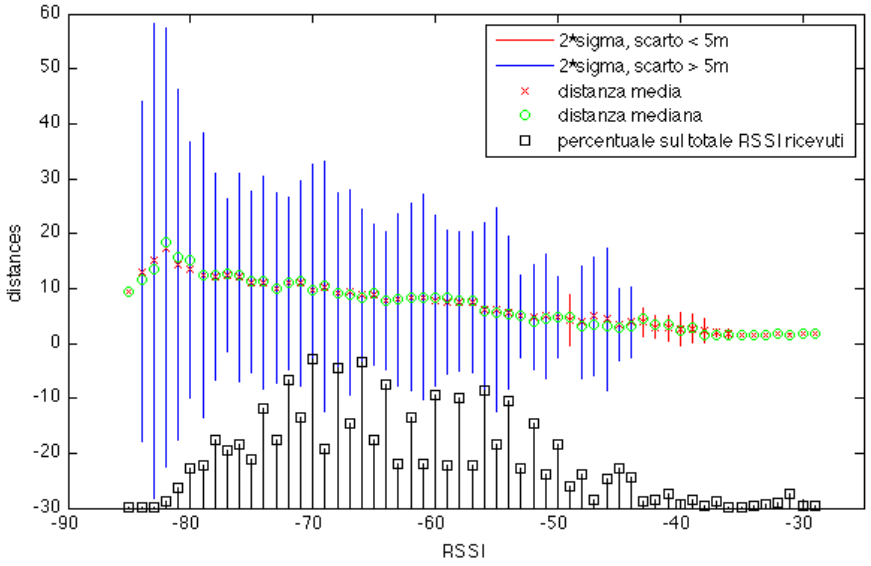


Figura 15: Il modello logaritmico di canale.

Di contro, un'analisi dettagliata dell'ambiente eseguita per ogni stanza o per ogni nodo, e in cui per la creazione dei molteplici modelli di canale si misurino più volte posizioni e distanze, appare eccessivamente onerosa rispetto al semplice posizionamento di ogni singolo nodo ($N_a = N$), rendendo superfluo il problema dell'autolocalizzazione.
La decisione adottata durante lo sviluppo di questo algoritmo é stata quella di fare una valutazione *bad case*: da un ricco dataset iniziale, che verosimilmente possiamo paragonare ad una serie di misure preparatorie effettuate sul campo, é stato estrapolato un modello loglineare[22], visibile in Fig. 16. Per ogni valore di RSSI sono segnati: la media $\bar{d}$ e la mediana $\tilde{d}$ delle distanze associate, l'intervallo di ampiezza $\pm 2\sigma_d$

[22] Questo modello é in realtà fallace, perché realizzato dagli stessi nodi che poi lo utilizzano per re-individuare le distanze. Tuttavia, come si nota ciò influisce solo ai margini dell'intervallo degli RSSI, in cui le misure sono poche e non si confondono tra loro.

(rosso se < 5m), e una visualizzazione della frequenza delle misure. Supponendo poi che i parametri di canale vengano preimpostati nella memoria interna dei mote prima del loro posizionamento, e non essendoci informazione a priori sulla futura collocazione di quest'ultimi all'interno dell'edificio, é stato scelto di adottare un modello unico per l'intero processo di misura. Come appare evidente dalla Fig. 15, si tratta di un modello estremamente inefficace: per evitare ulteriori imprecisioni, nella regressione sono stati accettati solo segnali con LQI > 95, mentre non si é discriminato in base alla potenza del segnale. Nella sezione 7.3 é proposto, ma non realizzato, l'impiego di più modelli per aree diverse.

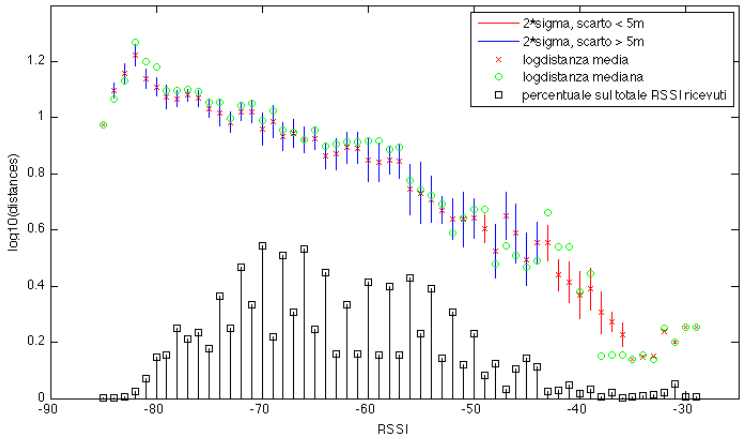


Figura 16: Il modello loglineare di canale.

- *RSSI minimo/buono.* Non tutti i segnali ricevuti possono essere accettati: più bassa é la potenza del segnale ricevuto, più difficile é calcolare la distanza a cui si trova la sorgente. Pertanto vengono scartati tutti i valori di RSSI inferiori ad una certa soglia; il nodo inoltre emette un avviso quando la potenza del segnale ricevuto é superiore alla soglia, ma non sufficiente da permettere una buona ricostruzione.

- *LQI minimo/buono.* Anche se la potenza del segnale é generalmente associata alla sua qualità, viene comunque applicata una selezione simile alla precedente, ma basata sui valori di LQI. Misure molto rumorose di d_{ji} possono inficiare la successiva localizzazione dell'intero patch, non limitando l'errore al solo nodo j.

- *Numero di pacchetti ricevuti.* Analizzando il dataset, ci si accorge della presenza di pacchetti vaganti, cioé di segnali generalmente lontani e ostruiti da pareti che talvolta superano questi impedimenti e comu-

nicano brevemente con nodi distanti: ovviamente queste misure sono troppo erratiche per potervi fare affidamento, a prescindere dalla precisione dell'informazione in esse contenuta. Vengono perciò scartati, dalla creazione di $S(i)$, tutti i nodi j che non inviano un certo numero minimo di pacchetti, o il cui numero di pacchetti ricevuti da i sia inferiore ad una certa percentuale sul totale.

- *ID di provenienza del segnale.* é possibile che, per errori di trasmissione, vengano ricevuti dei segnali in cui l'ID del nodo sorgente sia corrotto: ad esempio, nel dataset in nostro possesso compare più volte l'ID 44, non associato a nessun mote. Il problema del riconoscimento di eventuali "fantasmi" é stato risolto globalmente, comparando gli ID ricevuti con una lista di elementi noti; se si vuole garantire una rete scalabile e modulare, é necessario spostare il riconoscimento a livello locale, senza presupporre che il nodo conosca tutti e soli gli ID validi. Si propone l'uso di un bit di parità, o di metodi più avanzati di checksum.

Ogni nodo forma gradualmente la propria stella, comunicando in broadcast il proprio ID e ricevendo e classificando i segnali dei suoi vicini: il periodo di *data collection* può essere di durata variabile, al termine del quale vengono mediati i vari RSSI associati ai nodi della neighbourhood, e da questi valori medi vengono stimate le distanze.

6.3.2 Patch

La formazione del patch $P(i)$ prevede l'utilizzo delle informazioni contenute nelle vicine star $S(j)$: piuttosto che stabilire un sistema di interrogazione e risposta tra i e i singoli j, si preferisce far emettere un broadcast da ogni nodo della rete, contenente tutte le informazioni necessarie ai suoi vicini, e cioé:

- il proprio ID;
- la sua intera star $S(i)$, composta da una lista dei propri vicini e dalla distanza stimata da ognuno di essi;
- se é un'ancora o meno;
- nelle fasi successive, la propria localizzazione Q_i, la propria matrice $\mathbf{R}_i$ e la propria posizione assoluta p_i.

Una certa attenzione va dedicata alla stima delle distanze mancanti: il limite superiore B_{jk}, calcolato mediante la (28) dal nodo i con i tale che $j, k \in S(i)$, sfrutta la conoscenza di E non orientato, mentre le informazioni in possesso di i sono $S(i), S(j), S(k) \subset \vec{E}$. é possibile quindi che esista un nodo h non percepito da j o k, che tuttavia riceva i loro segnali: per questo e per

successivi motivi si é deciso di implementare una strategia di comunicazione 2-hop, spiegata nella sezione successiva, che elimini l'unidirezionalità nei collegamenti.
Il calcolo 29 del limite inferiore b_{jk}, invece, é sconsigliato dalla teoria per via della presenza di ostacoli; volendo ugualmente utilizzarne un valore approssimativo, lo si pone pari al raggio massimo entro cui un nodo della rete percepisce tutti i suoi vicini. Ovviamente questo parametro é dedotto da un'analisi a posteriori, che sfrutta la conoscenza delle vere distanze tra i nodi: ciò non toglie che, con una minima conoscenza dei mote, dell'ambiente operativo, e della generica strategia di posizionamento dei nodi (ad. es.: lontani almeno 5m l'uno dall'altro, per coprire una data area con meno dispositivi possibile), si possa stimare un valore approssimativo del raggio del disk graph, comune a tutti i nodi.

6.3.3 Stitch

Quando la risoluzione simultanea del problema globale (39) viene distribuita computazionalmente su tutta la rete, necessita di due particolari condizioni che si é tentato di soddisfare.

Innanzitutto, un metodo iterativo come il SOR può non garantire la simultaneità, essendo calcolato autonomamente dai vari nodi: poiché le loro posizioni assolute sono aggiornate a partire da quelle della rispettiva neighbourhood, non si può permettere che un nodo anticipi gli altri e compia più di un'iterazione rispetto ad un suo vicino.
La nostra soluzione sincronizza i vari nodi inserendo nel loop di convergenza l'intera rete, e risolvendo un nodo alla volta: se anche uno solo di essi deve ancora convergere, l'iterazione viene comunque applicata a tutti. In un contesto reale, si potrebbe aggiungere un contatore di cicli per ogni nodo, comunicato in broadcast: se sono presenti vicini ancora impegnati nell'n-esimo calcolo, un nodo dovrebbe evitare di procedere con l'$(n+1)$-esimo. é doveroso però dire che, potendo controllare la convergenza con il fattore di rilassamento w, un perfetto sincronismo lungo la rete non é strettamente necessario, e non dovrebbe apportare miglioramenti tali da giustificarne il costo implementativo.

Il secondo problema associato all'equazione (39) é che essa presuppone dei collegamenti bidirezionali tra i nodi, dovendo ottenere, per ogni vicino $j \in S(i)$ (quindi per ogni arco entrante da un nodo j), la propria posizione $q_{i(j)}$ espressa nella sua localizzazione Q_j (quindi é necessario che $i \in S(j)$, ovvero che vi sia anche un arco uscente da i a j).
Una possibile soluzione alla presenza di un arco (i, j) privo di elemento simmetrico, codificata dalla funzione `transitivefix.m`, sfrutta un nodo k con cui sia i che j hanno un collegamento bidirezionale: k funge da ponte tra i

due nodi lungo la direzione in cui non riescono a comunicare, ovvero funge da ripetitore delle informazioni $S(j)$ trasmesse da j. Casi più complicati, in cui questa semplice risoluzione non sia sufficiente, sono di solito frutto di configurazioni particolari con grafi a molto bassa connettività, e non sono stati considerati.

6.3.4 Starting map

L'ultima problematica affrontata riguarda il numero e il piazzamento dei nodi ancora: come si può notare in Fig. 17, utilizzare un numero minimo $N_a = 3$ di ancore non é consigliato, poiché un piccolo errore nella localizzazione o nell'orientamento dei patch iniziali si ripercuote e si amplifica su tutti quelli ad essi collegati, pur mantenendo la struttura rigida della rete. Sono state tentate varie disposizioni, ma le più efficaci sembrano essere quelle suggerite da [20], ed illustrate in Fig. 18 : nel paragrafo 6.4 sono presenti i grafici per $N_a = 8$ e $N_a = 20$.

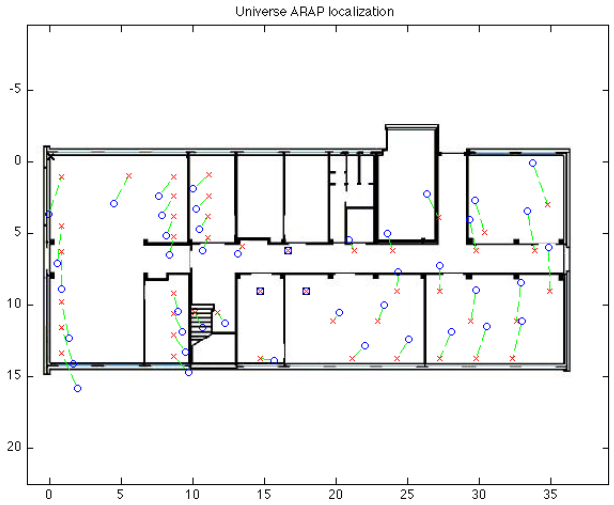


Figura 17: Autolocalizzazione con 3 ancore e misura esatta delle distanze tra i nodi.

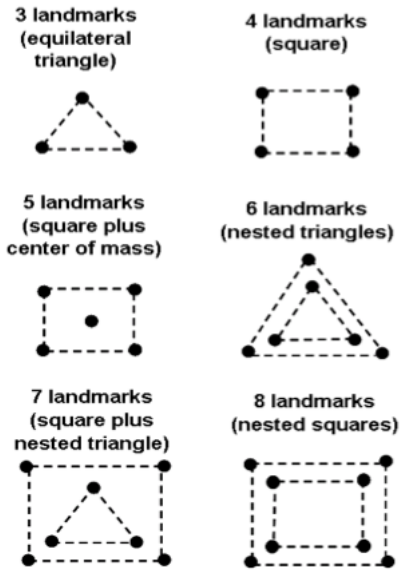


Figura 18: Scelta ottima delle ancore

6.4 Prestazioni ottenute

L'algoritmo é di rapida esecuzione, e la correttezza é dimostrata su una simulazione con rumore di misura normalizzato $\sim \mathcal{N}(0\,,\,0.15)$, come mostrato in Fig. 19. Sfortunatamente, come si può notare nelle Fig. 20, 21 e 23, non é possibile prevedere quanto il numero o il posizionamento delle ancore influenzino gli errori di stima dei vari nodi; inoltre, la poca variabilità dell'errore medio conferma i risultati già previsti a proposito della localizzazione tramite RSSI: cioé, per quanto buono possa essere un algoritmo range-based, la sua accuratezza sarà comunque subordinata alla rumorosità dei segnali scambiati [21]. Per questo motivo, l'imposizione delle soglie di RSSI, LQI e numero di misure, elimina potenziali cause di disturbo e migliora la stima delle posizioni: si confrontino le Fig. 21 e 22.

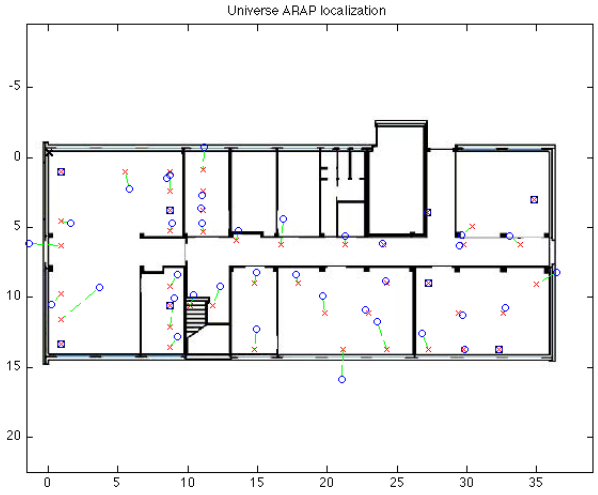


Figura 19: Autolocalizzazione con 8 ancore e misure corrotte solo da rumore gaussiano con varianza normalizzata $\sigma = 0.15$.
Errore medio = 1.1018; Errore massimo = 2.5753.

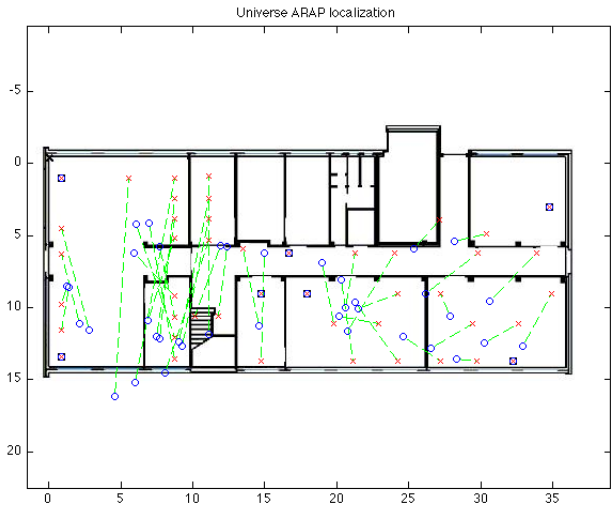


Figura 20: Autolocalizzazione con 7 ancore, dati reali, soglia RSSI di -80dB, soglia LQI di 95, scarto dei nodi inferiori all'1% del totale.
Errore medio = 2.8655; Errore massimo = 5.7349.

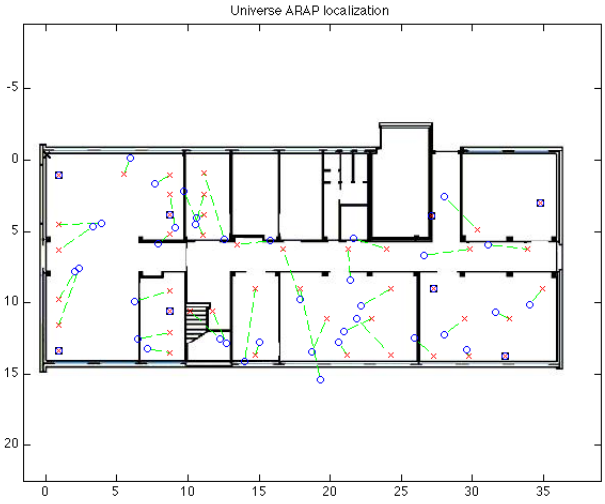


Figura 21: Autolocalizzazione con 8 ancore, dati reali, soglia RSSI di -80dB, soglia LQI di 95, scarto dei nodi inferiori all'1% del totale.
Errore medio = 2.4058; Errore massimo = 5.9531.

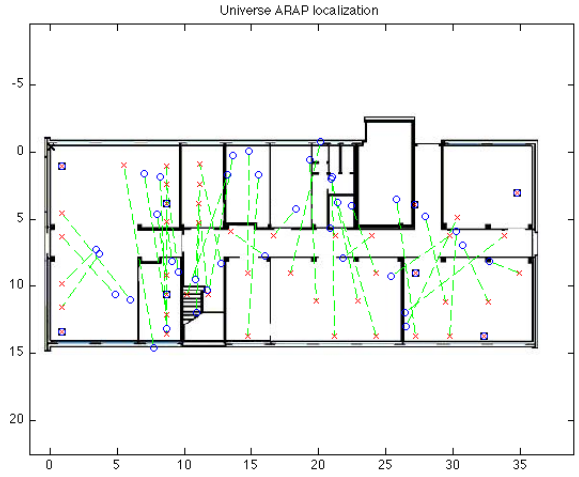


Figura 22: Autolocalizzazione con 8 ancore, dati reali, soglia RSSI di -200dB, soglia LQI di 45, nessuno scarto dei nodi.
Errore medio = 3.4675; Errore massimo = 9.0462.

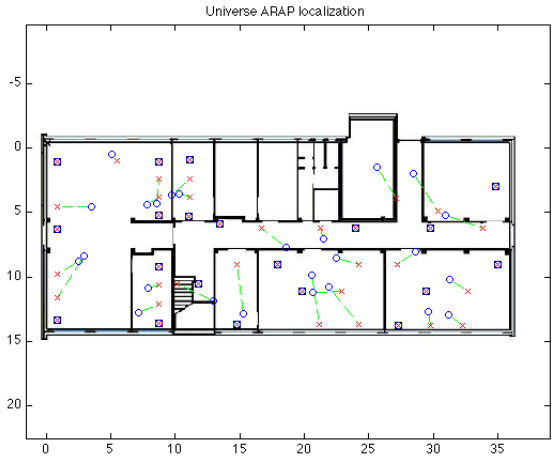


Figura 23: Autolocalizzazione con 20 ancore, dati reali, soglia RSSI di -80dB, soglia LQI di 95, scarto dei nodi inferiori all'0.1% del totale.
Errore medio = 2.1585; Errore massimo = 3.9865.

7 Conclusioni e sviluppi futuri

7.1 *Range-Based*

Si sono studiati ed implementati i metodi *range-based* dei minimi quadrati, della combinazione convessa e del calcolo del baricentro, ottenendo mediamente un errore che si aggira attorno ai 2-3 metri che, come ci si aspettava, diminuisce all'aumentare dei nodi ancora. L'applicazione del filtro di *Kalman* esteso in forma d'informazione riesce poi, a partire da una di queste stime, a migliorare la stima e ridurre anche di un metro l'errore medio. Si é notato che le iterazioni necessarie alla convergenza sono in un numero relativamente basso e questo aspetto riduce la complessitá computazionale e il tempo di esecuzione dell'algoritmo. L'algoritmo di *Kalman* é stato in seguito modificato al fine di tener conto dell'informazione sui nodi ancora, riportando la stima della posizione di questi nodi al valore corretto; in questo modo l'errore risulta ulteriormente ridotto, anche se non di molto. La stima nel caso precedente, infatti, non si discosta molto dal valore reale, dato che la posizione iniziale é corretta e lo stato del sistema non cambia in maniera rapida.

Un interessante metodo che per ragioni di tempo non é stato implementato prevede l'utilizzo di un filtro particellare, particolarmente adatto a situazioni in cui la dinamica é non lineare e il rumore di misura é non gaussiano. Un altro aspetto che sarebbe interessante sperimentare é la *localizzazione* ed il *tracking* basandosi sulle stime delle posizioni dei nodi fissi

ricavate dal processo di autolocalizzazione al fine di vedere come gli errori derivanti dai due processi di stima si combinino.

7.2 *Map-Based*

Le tecniche *map-based* SVM sono state presentate nella loro applicazione alla regressione in termini generali senza pretese di completezza o eccessiva rigorosità. Sono stati reperiti strumenti, implementanti tali tecniche, adatti alla soluzione del problema in esame. Si é proceduto alla progressiva disamina delle controversie incontrate, proponendo soluzioni derivanti da articoli citati o da ragionamenti argomentati; in particolare si é proposto uno spazio feature di dimensione maggiore di quelli trovati in letteratura che le prove sperimentali hanno dimostrato avere modesti effetti di riduzione dell'errore medio e massimo di localizzazione, inoltre metodi algoritmici implementabili al calcolatore sono stati proposti per una procedura di autolocalizzazione completamente automatizzata. Infine é stato brevemente discusso un approccio alternativo di autolocalizzazione indipendente dal posizionamento della rete già presente in alcuni degli articoli citati. Entrambi i campi inerenti i criteri di selezione del nodo (per il caso di autolocalizzazione classica) e quelli di campionamento dello spazio feature (per l'approccio alternativo proposto) sono ambiti di ricerca attivi e futuri sviluppi potrebbero arricchire la letteratura a riguardo non completamente esaustiva.

7.3 ARAP

Date complessità e novità dell'argomento, la dimensione della letteratura e le difficoltà implementative, non é stato possibile approfondire la ricerca come desiderato; la progettazione di una soluzione distribuita al problema dell'autolocalizzazione presenta ancora molte possibilità di miglioramento. Ne elenchiamo alcune:

- La localizzazione del patch (30), e la conseguente stress majorization (31), utilizzano tutte le distanze d_{ij} allo stesso modo: tuttavia alcune di esse sono più precise e quindi più "importanti" delle altre. Ad esempio, le distanze tra le ancore vengono calcolate, non misurate, e l'unico errore che le influenza é quello di posizionamento; le distanze con RSSI alto provengono di solito da nodi nella stessa stanza, dato che eventuali ostacoli degradano la potenza del segnale facendone sembrare la sorgente più lontana; le distanze con LQI alto sono spesso misurate attraverso canali sgombri. Si può quindi cercare di adoperare una forma pesata del MDS (Weighted MDS), aumentando l'influenza di questi archi o anche forzandoli a giacere sullo stesso piano, per non accrescere inutilmente la dimensionalità della soluzione.

- Nel caso di reti di grandi dimensioni, che agiscono su ambienti diversi, é consigliato l'uso di diversi modelli di canale: una scelta conveniente é quella di memorizzarne i parametri solo nei nodi ancora, che attraverso un broadcast li diffondano al resto della rete. Il vantaggio sarebbe duplice: mediando i parametri ricevuti dalle varie ancore, si suppone che il nodo generico ricostruisca un modello più accurato dell'area circostante; inoltre, ad autolocalizzazione completata, un'ancora può continuare a monitorare la potenza dei segnali verso i nodi vicini, in particolare altre ancore. Variazioni ambientali significative (come il fumo di un incendio) modificano il canale, rendendo necessario un modello adattativo. Dai nuovi valori di RSSI, e dalle distanze note, le ancore possono ricalcolare il fattore di pathloss del modello log-lineare e propagarne il valore aggiornato.

- In alcuni casi, nodi collocati in zone poco accessibili di un edificio possono avere scarsa o insufficiente connettività, o possono ricevere solo segnali deboli o rumorosi. In questo caso, l'impiego di un nodo mobile di posizione nota (presumibilmente un palmare, in teoria lo stesso con cui si può configurare in loco le ancore) può aggiungere dei nodi ancora "temporanei" che irrigidiscano il grafo, risolvano ambiguità, migliorino stime imprecise. Ovviamente, con lo stesso dispositivo si potrebbe banalmente impostare in maniera diretta la posizione dei nodi problematici, trasformandoli in ancore, ma questa soluzione é poco efficiente quando si intervenga su aree invece che su singoli punti; inoltre, supponendo oneroso il calcolo della posizione del nodo mobile, si cerca di sfruttarne al meglio l'informazione da parte di tutti i nodi in grado di recepirne la trasmissione.

- Intimamente collegata al punto precedente è la possibilità da parte di un nodo di stimare autonomamente la bontà della propria localizzazione, e la confidenza in questa stima: questo é un problema assai complesso e ancora irrisolto, a cui si é cercato di dare una prima soluzione monitorando alcuni casi critici di LQI, RSSI e connettività. é stato incluso anche un esame degli autovalori prodotti da `cmdscale()` per verificare la fitting del troncamento bidimensionale, ma ignorando i dettagli implementativi della funzione non si sa se, e quanto, essa sia replicabile in linguaggio nesC.

- Essendo l'autolocalizzazione di una rete dotata di errori significativi, derivanti dagli errori di localizzazione, ci si potrebbe chiedere quanto una mappatura imperfetta della rete influisca sul tracking di un nodo mobile (palmare): un primo abbozzo costruttivo, anche considerando le specifiche hardware dei mote utilizzati, potrebbe essere di impiegare un filtro di Kalman alimentato continuamente da nuove misure di distanza, e la cui risoluzione computazionale sia delegata interamente

al palmare. I vari nodi della rete emettono già in broadcast le loro posizioni (ed eventualmente i parametri di canale, vedi sopra); il nodo mobile, su cui é possibile salvare la planimetria dell'area, esegue una stima delle distanze secondo Kalman e da queste calcola la sua posizione con una semplice combinazione convessa delle posizioni dei nodi circostanti.

- In aggiunta, disponendo di una mappa é possibile forzare le posizioni dei nodi in modo che restino all'interno delle mura perimetrali, utilizzando muri di potenziale o semplici intervalli di minimo e massimo; i vari procedimenti di autolocalizzazione, infatti, talvolta identificano un nodo come esterno all'edificio. Questo caso, banalmente risolvibile, prevede però una conoscenza pregressa dell'ambiente di applicazione, preimpostata nella rete o meglio ancora memorizzata e applicata dal nodo mobile.

- Infine, il lavoro svolto finora si é interessato solo al dataset a nostra disposizione, e alcuni risultati sono frutto di fine-tuning: verificare (in maniera indipendente dalla casistica) la robustezza degli algoritmi e delle metodologie presentate potrebbe essere l'argomento di un successivo lavoro.

7.4 La tabella riassuntiva

I risultati ottenuti sono schematicamente riassunti in tabella (3). L'errore medio é stato calcolato in con un numero di nodi ancora pari a 20 uniformemente distribuiti nell'ambiente. É stato poi messo a confronto il tempo di esecuzione e si sono divisi gli algoritmi in algoritmi *lenti* e algoritmi *veloci* (si classificano come veloci quegli algoritmi che danno una stima in tempo reale e come lenti gli algorimi percui é necessario un tempo d'attesa superiore ad un secondo). Infine la terza colonna é un'indice della difficoltá incontrata nell'implementazione del metodo in esame.

Metodo	Errore medio	Tempo di esecuzione	Difficoltá
Minimi quadrati	3.2251	veloce	medio
Baricentro	2.7420	veloce	facile
Combinazione convessa	4.0592	veloce	facile
Filtro di *Kalman*	2.7126	lento	difficile
SVM1	2.1880	veloce	difficile
SVM2	1.9580	veloce	difficile
SVM3	2.0030	veloce (SVM2·2)	difficile
ARAP	2.2272	veloce	molto difficile

Tabella 3: Tabella riassuntiva

Riferimenti bibliografici

[1] A Kernel-Based Learning Approach to Ad Hoc Sensor Network Localization XUANLONG NGUYEN, MICHAEL I. JORDAN, and BRUNO SINOPOLI 30

[2] ARIADNE: A Dynamic Indoor Signal Map Construction and Localization System 30

[3] MoteTrack: a robust, decentralized approach to RF-based location tracking 30

[4] Survey of Wireless Indoor Positioning Techniques and Systems Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu 30

[5] An Introduction to the Theory of Reproducing Kernel Hilbert Spaces Vern I. Paulsen 30

[6] The Nature of Statistical Learning Theory by V. N. Vapnik Berlin: Springer-Verlag, 1995 30

[7] Support Vector Machines: Training and Applications Edgar E. Osuna, Robert Freund and Federico Girosi 33

[8] A Tutorial on Support Vector Regression. NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK 33

[9] http://www.esat.kuleuven.be/sista/lssvmlab/ 33

[10] Localization in Wireless Sensor Networks via Support Vector Regression. WANG Yong, XU Xiaobu,Tao Xiaoling 30, 34, 37

[11] Analisi sperimentale di connettività per una rete di sensori wireless, Luca Schenato, Francesco Roveron

[12] An As-Rigid-As-Possible Approach to Sensor Network Localization Zhang, Liu, 2010 47, 52

[13] PATCHWORK: Efficient Localization for Sensor Networks by Distributed Global Optimization Koren, Gotsman, Ben-Chen, 2005 48, 50

[14] On the Node Localizability of Wireless Ad-hoc and Sensor Networks Yang, Liu, 2010 48

[15] Characterizing generic global rigidity Dylan Thurston `http://www.math.columbia.edu/~dpt/speaking/Rigidity/talk.pdf` 48

[16] Beyond Trilateration: On the Localizability of Wireless Ad-hoc Networks Yang, Liu, 2009

[17] Location, Localization and Localizability Yang, Liu, 2011

[18] Modern Multidimensional Scaling Borg, Groenen, 1997

[19] http://en.wikipedia.org/wiki/Successive_over-relaxation

[20] A Practical Approach to Landmark Deployment for Indoor Localization Chen, Francisco, Trappe, Martin, 2006

[21] Cable Belisari et al., 1993

[22] http://sentilla.com/files/pdf/eol/tmote-sky-datasheet.pdf 49

[23] http://nescc.sourceforge.net/ 49

[24] http://www.tinyos.net/

[25] Metodi Statistici per l'Identificazione
Giorgio Picci

[26] Research on RSSI-Based Localization in WSN
Jin Rencheng, Wang Hongbin, Peng Bo Ge Ning

[27] An interlaced Extended Information Filter for Self-Localization in Sensors Networks
Andrea Gasparri, Federica Pasucci 50