ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA ELETTRONICA E TELECOMUNICAZIONI PER LO SVILUPPO SOSTENIBILE

ANALISI DI ALGORITMI PER RETI DI SENSORI RADAR UWB BASATI SU MISURE SPERIMENTALI

Tesi di Laurea in:

Laboratorio di Telecomunicazioni LM

Relatore:

Prof. Ing. **Enrico Paolini**

Correlatore:

Prof. Ing. **Andrea Giorgetti**

Dott. Ing. **Matteo Mazzotti**

Presentata da:

Enrico Forti

Sessione II

Anno Accademico 2013/2014

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

In un sistema radar è fondamentale rilevare, riconoscere e cercare di seguire il percorso di un eventuale intruso presente in un'area di osservazione al fine ultimo della sicurezza, sia che si consideri l'ambito militare, che anche quello civile. A questo proposito sono stati fatti passi avanti notevoli nella creazione e sviluppo di sistemi di localizzazione passiva che possano rilevare un *target* (il quale ha come unica proprietà quella di riflettere un segnale inviato dal trasmettitore), in modo che esso sia nettamente distinto rispetto al caso di assenza dell'intruso stesso dall'area di sorveglianza. In particolare l'ultilizzo di *Radar Multistatico* (ossia un trasmettitore e più ricevitori) permette una maggior precisione nel controllo dell'area d'osservazione. Tra le migliori tecnologie a supporto di questa analisi vi è l'UWB (*Ultra Wide-Band*), che permette di sfruttare una banda molto grande con il riscontro di una precisione che può arrivare anche al centimetro per scenari *in-door*.
L'UWB utilizza segnali ad impulso molto brevi, a banda larga e che quindi permettono una risoluzione elevata, tanto da consentire, in alcune applicazioni, di superare i muri, rimuovendo facilmente gli elementi presenti nell'ambiente, ossia il *clutter*. Quindi è fondamentale conoscere algoritmi che permettano la *detection* ed il *tracking* del percorso compiuto dal *target* nell'area. In particolare in questa tesi vengono elaborati nuovi algoritmi di *Clustering* del segnale ricevuto dalla riflessione sull'intruso, utilizzati al fine di migliorare la visualizzazione dello stesso in *post-processing*. Infine questi algoritmi sono stati anche implementati su misure sperimentali attuate tramite nodi PulsOn 410 Time Domain, al fine ultimo della rilevazione della presenza di un *target* nell'area di osservazione dei nodi.
L'eleborato è quindi suddiviso nei seguenti capitoli:

- Capitolo 1 : *Tecnologia Ultra Wideband*;
- Capitolo 2 : *Descrizione del sistema Radar*;

- Capitolo 3 : *Algoritmi CFAR e di Clustering*;
- Capitolo 4 : *Codice*;
- Capitolo 5 : *Simulazioni*;
- Capitolo 6 : *Time Domain PulsOn 410*;
- Capitolo 7 : *Misure Sperimentali*;
- Capitolo 8 : *Elaborazione dei dati*;
- Capitolo 9 : *Conclusioni*.

Nei primi due capitoli vengono descritti in modo dettagliato la tecnologia UWB ed il sistema radar; nel terzo capitolo si parla degli algoritmi utilizzati di *detection* e *tracking*, mentre nel quarto viene riportato il codice utilizzato per eseguire il *clustering*; il quinto capitolo riporta le simulazioni eseguite in base alle analisi precedenti; il sesto fornisce una descrizione delle antenne e nodi utilizzati nelle misure sperimentali del settimo capitolo; l'ottavo riporta una serie di elaborazioni successive e anche diverse fra loro, al fine di giungere alle conclusioni riportate nel nono ed ultimo capitolo.

Capitolo 1

Tecnologia *Ultra Wideband*

L'*Ultra Wideband (UWB)* è una tecnologia utilizzata nell'ambito delle radio comunicazioni che, nella maggior parte delle applicazioni, sfrutta le potenzialità dell' *Impulse Radio (IR)*, ossia impulsi radio a breve durata temporale. Il loro utilizzo e la bassa durata (fra i *100 ps* e *1 ns*) implica bande molto elevate; infatti *Wideband* non significa, come avviene convenzionalmente, l'utilizzo di un'ampia banda di modulazione ed un elevato *Data-rate* (come avviene nel *W-CDMA*, tecnica di accesso multiplo al canale nel caso di reti cellulari *3G*, per la quale si hanno canali definiti appunto *Wideband* di 5MHz), ma appunto grandi bande rispetto a quelle della modulazione e una riduzione della densità spettrale di potenza (vedi Figura 1.1)[5] [8].

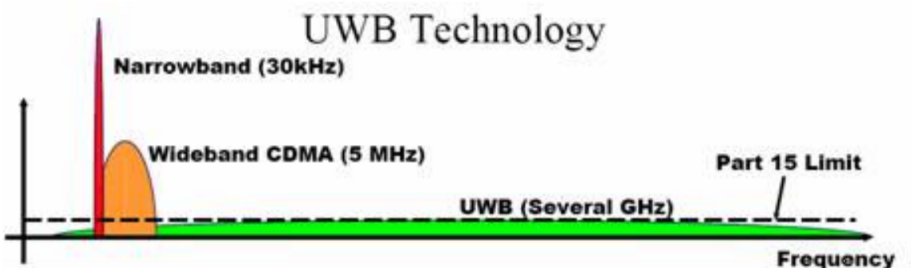


Figura 1.1: Confronto fra le varie densità spettrali irradiate da diversi sistemi di comunicazione: *Narrowband* ossia a banda stretta (in rosso), *Wideband* ossia a banda larga (in arancione) e UWB (in verde)[3].

Quindi i segnali sono classificati come UWB in base alla loro larghezza di banda e soprattutto si ha una distinzione fra definizione utilizzata negli USA e quella in Europa, nel senso che:

- In USA sono definiti UWB segnali con banda maggiore di 500 MHz [6].

- In Europa invece per bande maggiori dei 50 MHz [7].

In particolare l'FCC definisce un segnale UWB anche in base alla banda frazionaria che deve essere maggiore di 0.25; la banda frazionaria è definita quindi come:

$$B_f = 2 \cdot \frac{f_H - f_L}{f_H + f_L} \tag{1.1}$$

dove f_H e f_L sono rispettivamente il più alto e il più basso valore in frequenza a $-3dB$ del segnale UWB.

1.1 Specifiche *UWB*

La banda allocata ai sistemi UWB è di 7.5 GHz la quale, se viene utilizzata in modo ottimale, permette di avere una massima potenza a lato trasmettitore pari a 0.5 mW. La bassa potenza di trasmissione implica l'utilizzo di impulsi UWB multipli combinati fra loro con il fine ultimo di trasportare un bit di informazione.
I sistemi di comunicazione radio ad impulsi detti *Impulse Radio* (IR), utilizzano impulsi di brevissima durata temporale: ciò permette di avere una banda spettrale *Ultra Wide* [19]. In termini di applicazioni radio, questo metodo è visto come una tecnica di modulazione ad impulso PPM: *Pulse Position Modulation*.
I segnali IR modulati nel tempo (TM ossia *Time modulated*), sono considerati come trasmissioni a banda base senza portante; questa assenza della portante differenzia questi segnali da quelli *Direct Sequence Spread Spectrum Multi-Carrier* (DS-SS-MC) i quali possono essere utilizzati come alternativa ai segnali IR per l'UWB.
L'UWB, avendo una banda così elevata, potrebbe interferire con la maggior parte dei sistemi di Telecomunicazione a causa della presenza di componenti spettrali distribuite su una banda molto ampia. Per evitare questo problema la *Federal Comunication Commition (FCC)* ha regolamentato le maschere di emissione degli apparati UWB (come si può notare in Tabella 1.1 e Figura 1.2).

Si nota, dalla Figura 1.2, che le frequenze più colpite dall'interferenza dell'UWB, sono quelle del GPS (*Global Positioning System*), per questo l'FCC ha definito per esse un EIRP molto basso (una maschera di *-73.5 dBm*).
Invece i sistemi UWB sono per la maggior parte allocati nella banda $[3.1 \div 10.6]$ GHz, poiché è quella con il miglior valore dell'EIRP

Frequenza [MHz]	**EIRP [dBm/Mhz]**
960 - 1610	-75.3
1610 - 1990	-53.3
990 - 3100	-51.3
3100 - 10600	-41.3
Oltre 10600	-51.3

Tabella 1.1: Tabella che mostra i limiti imposti all'EIRP dall'FCC in base alla frequenza di utilizzo nel caso *indoor*.

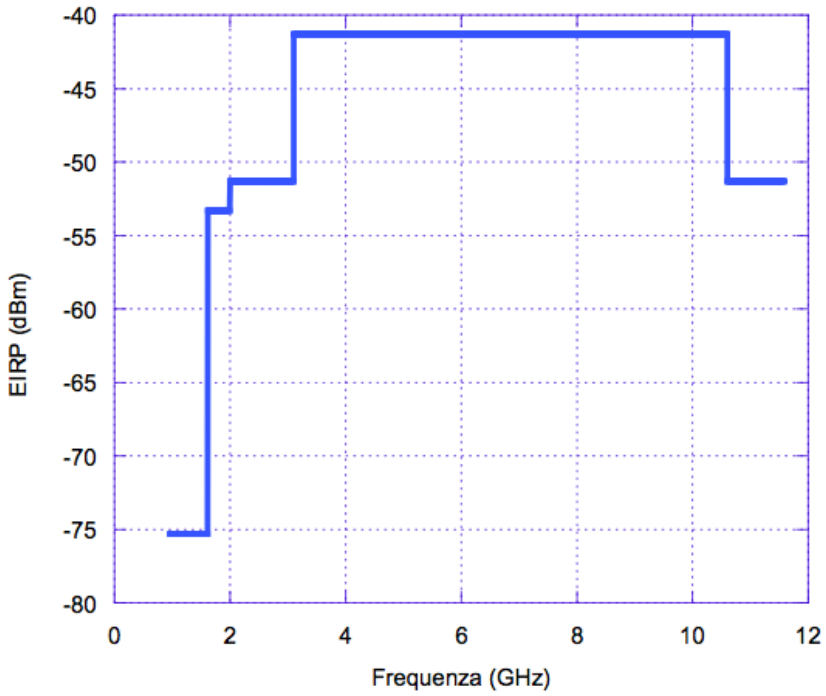


Figura 1.2: Maschera delle limitazioni EIRP imposte dall'FCC nel caso *indoor*[10].

$(-41.3dBm)$.

1.2 Segnale UWB

Il segnale UWB è storicamente basato sul concetto di *Impulse Radio*, che si riferisce alla generazione di un treno di impulsi di breve durata, per i quali esiste un ampio spettro che deve sottostare a certi requisiti imposti dalla maschera (vedi Figura 1.2). La combinazione di più impulsi comporta il trasporto dell'informazione e quindi dei bit.
Un'alternativa rilevante all'IR è quella di utilizzare segnali multibanda che permettono di raggiungere le specifiche (definite dall'FCC) di una banda di $500MHz$, o altri tipi di segnali utilizzati nei sistemi UWB che verranno descritti di seguito.

1.2.1 *Impulse Radio* UWB

La comunicazione TM-UWB (*Time Modulated Ultra Wide-Band*) si basa sulla trasmissione di un segnale che, nei casi più comuni, può essere: un impulso (di solito Gaussiano) o un Monociclo. L'utilizzo di questo tipo di trasmissione non richiede un'aggiuntiva portante modulante, ma sfrutta un approccio analogo ai segnali in banda base. Un simbolo trasmesso (quindi un bit di informazione) viene sparso su un numero di monocicli con il fine ultimo di aumentare il guadagno di processo e ridurre l'interferenza e il rumore. Il guadagno di processo può essere definito come [19]:

$$PG_1\,[dB] = 10\log_{10}[N]\,. \tag{1.2}$$

In questo caso N è il numero di monocicli/impulsi utilizzati per trasportare un bit di informazione. Gli impulsi non occupano necessariamente tutto il periodo, anzi spesso ne occupano una porzione molto limitata riducendo così il *duty cycle*. I ricevitori devono quindi ricevere informazioni dal canale solo per una frazione di periodo, per cui l'impatto dell'interferenza è notevolmente ridotto; questo permette di definire un secondo guadagno di processo:

$$PG_2\,[dB] = 10\log_{10}\left(\frac{T_f}{T_p}\right) \tag{1.3}$$

dove T_p è l'ampiezza dell'impulso (in termini di durata temporale), mentre T_f è il *time hopping* che si ha tra un impulso e il successivo. Ovviamente il guadagno di processo totale (PG) sarà definito dalla

somma dei due guadagni visti nelle Equazioni (1.2) e (1.3):

$$PG\,[dB] = PG_1 + PG_2. \tag{1.4}$$

Impulso Gaussiano

L'impulso Gaussiano e le sue derivate hanno una risposta in frequenza abbastanza piatta, ma soprattutto la componente in continua maggiormente ridotta.

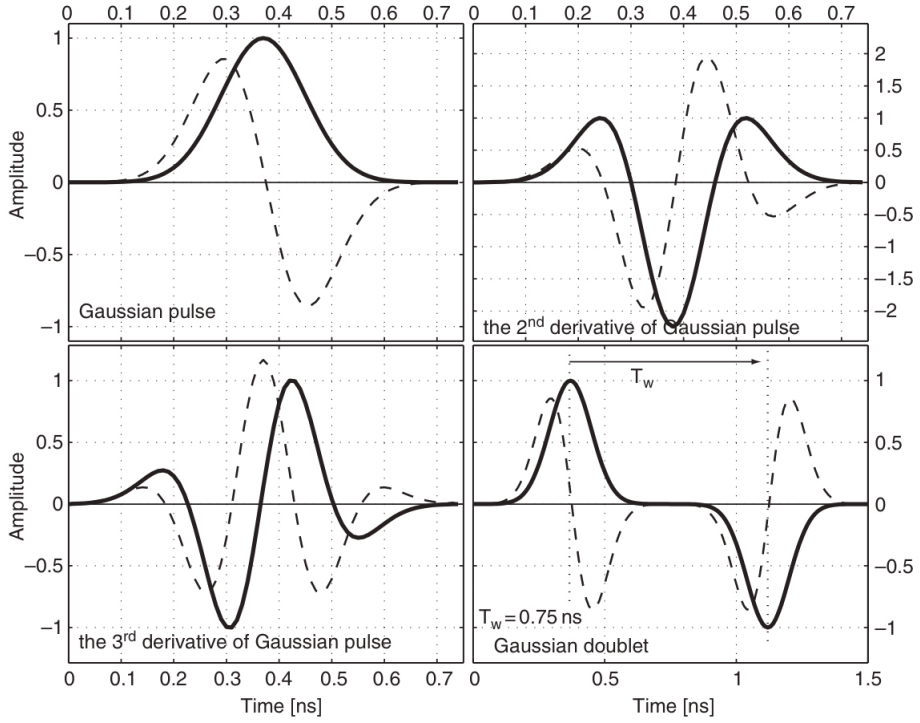


Figura 1.3: Impulso Gaussiano e sue derivate nel dominio del tempo [19].

L'equazione dell'impulso Gaussiano è invece [11]:

$$s(t) = Ke^{-(\frac{t}{\tau})^2} \tag{1.5}$$

dove τ rappresenta la larghezza dell'impulso, mentre K il valore di ampiezza del segnale.

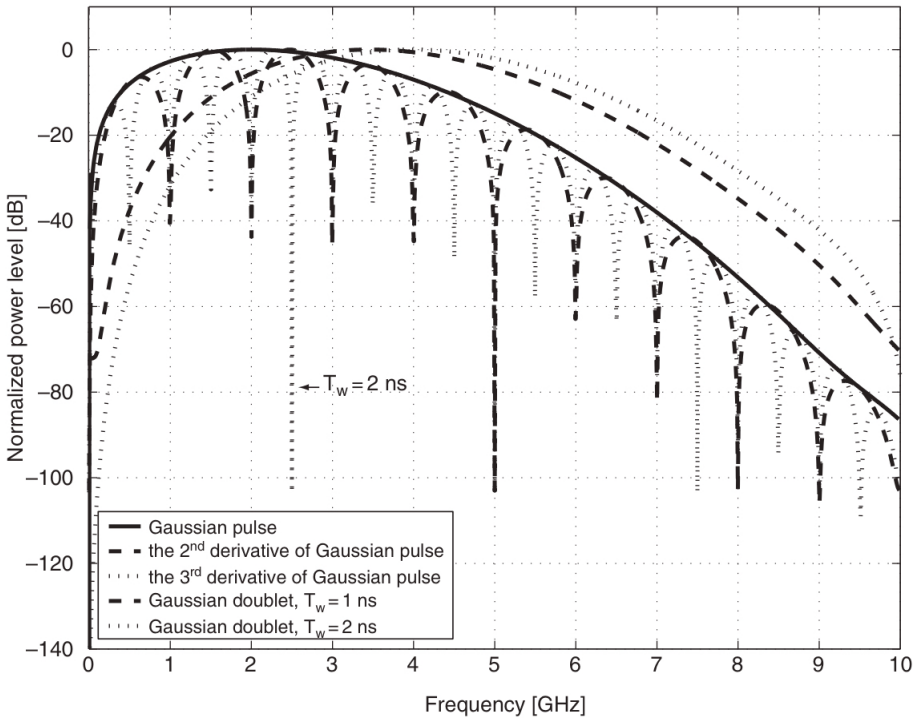


Figura 1.4: Impulso Gaussiano e sue derivate nel dominio delle frequenze [19].

Monociclo Gaussiano

Il monociclo Gaussiano, grazie alla sua breve durata, garantisce una banda abbastanza piatta e una relativa diminuzione della componente continua. Si considera quindi il caso in cui l'impulso abbia una durata di 0.1 ns, per cui la formula diventa (1.6):

$$s(t) = 0.1[1 - \cos(2\pi f_s t)]cos(2\pi f_c t) \tag{1.6}$$

dove f_c è la frequenza di centrobanda, mentre f_s è il valore che influisce sulla banda complessiva (vedi Figure 1.5 e 1.6).

Si nota che il monociclo è molto simile alla derivata prima dell'impulso Gaussiano (vedi Figura 1.3) [12], in particolare si ha una distribuzione della banda che a -10dB ha una ampiezza di banda attorno alla frequenza f_c di centro banda (vedi Figura 1.6) [14]. Per questi motivi il monociclo Gaussiano è quello più utilizzato nella radiolocalizzazione tramite sistemi UWB.

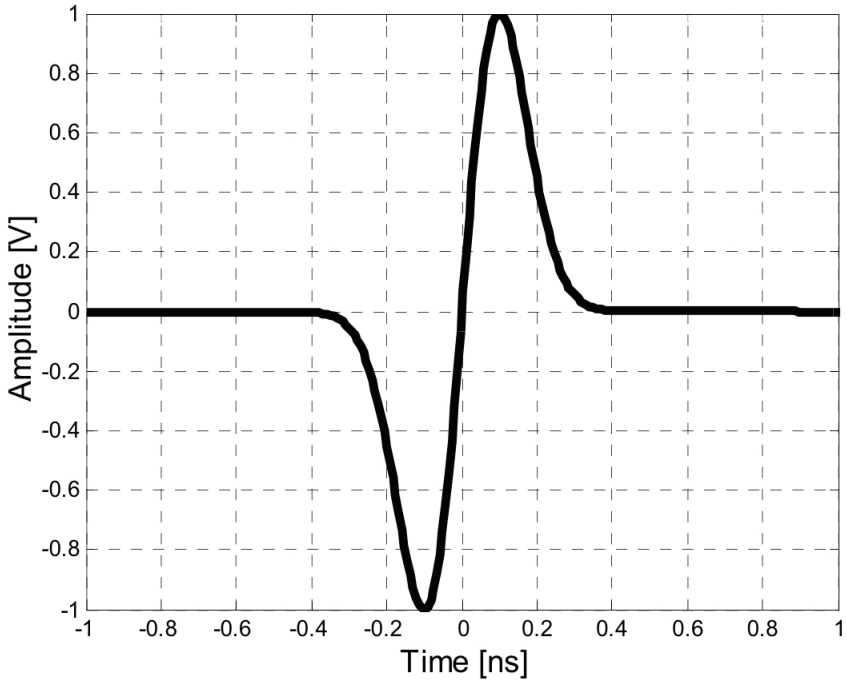


Figura 1.5: Monociclo Gaussiano nel dominio del tempo di durata 0.1 ns[13].

1.2.2 Schemi Multi-Portante

Spread Spectrum Multi-portante

Un approccio alternativo all'IR è quello che consiste nell'estendere le tecniche utilizzate nel *Direct Sequence Spread Spectrum* ($DS-SS$), o nel $CDMA$ (*Code Division Multiple Access*). Il $CDMA$ a larga banda con multiple portanti ($Multi-Carrier$ o MC) può permettere di creare un segnale UWB così come è stato definito direttamente dall'FCC (ossia con una banda maggiore dei $500MHz$).
Esistono tre principali tecniche per creare una trasmissione *Spread Spectrum multi-carrier* (SS-MC):

- *Multi carrier CDMA*: tramite un codice pseudo-random (PR), il codice iniziale subisce lo spreading nelle frequenze delle sottoportanti.
- *Multi carrier Direct Sequence (DS) CDMA*: lo *spreading* avviene nel dominio del tempo grazie ad un blocco serie/parallelo ed un unico codice di *spreading*.
- *Multitone (MT) CDMA*: anche in questo caso lo *spreading* avviene nel dominio del tempo, anche se, a causa della piccola distanza

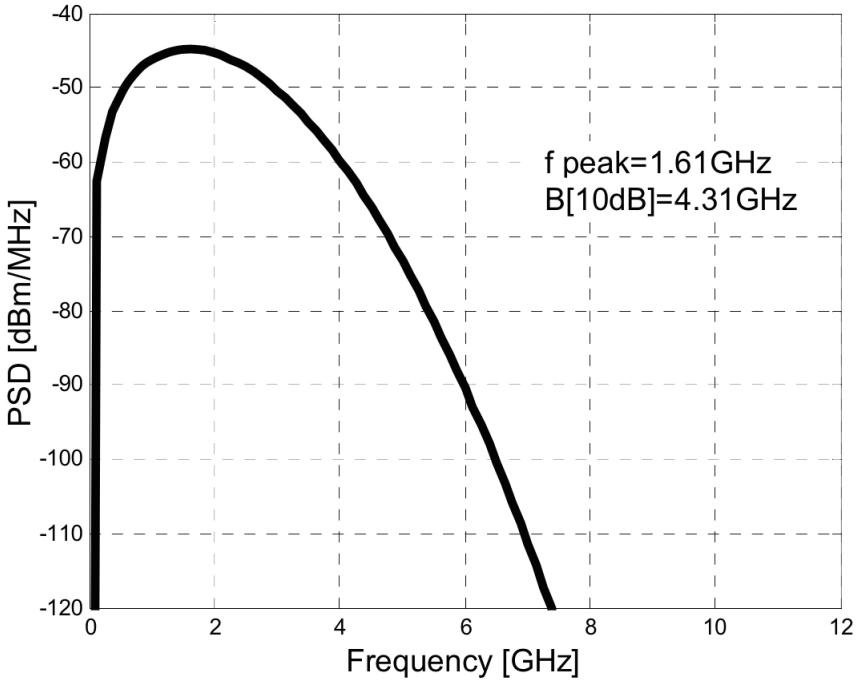


Figura 1.6: Rappresentazione della densità spettrale di potenza normalizzata del monociclo di Figura 1.5[13].

tra le sottoportanti, la banda è più piccola rispetto ai precedenti due casi.

La tecnologia multi-portante è utilizzata in applicazioni ad alto *Data-rate*. I vantaggi di questi sistemi dipendono dal fatto che per ogni sotto-portante, si ha un *Data-rate* minore di quello che si avrebbe nel caso di singola portante, per cui la sincronizzazione delle sequenze di *spreading* a ricevitore, è migliore poichè l'interferenza intersimbolo (ISI) è ridotta [19]. Gli svantaggi invece riguardano la complessità dei ricevitori che, a causa dello *spreading* e dell'utilizzo di più portanti a frequenze diverse, necessitano di mixer o di blocchi che possano eseguire la FFT (*Fast Fourier Transform*).

UWB Multibanda

La sovrapposizione di segnali UWB con una banda approssimativa di $500MHz$, porta alla definizione dell'UWB multibanda che è comunque definito come un segnale UWB secondo l'FCC. L'utilizzo di questa tecnica permette di considerare un gruppo di segnali UWB sfasati in frequenza e sovrapposti, al fine di aumentare la flessibilità, di rispettare la coesistenza di più sistemi e di mitigare l'attenuazione dovuta alla

propagazione[19].

1.2.3 *Data Modulation*

Ai fini dell'invio, trasporto e ricezione dell'informazione, la modulazione rimane un fattore fondamentale ed imprescindibile che deve essere utilizzato anche nei segnali UWB. Esistono molti tipi di modulazioni applicabili, sia ortogonali che antipodali, più o meno efficienti a seconda dell'applicazione e dalle forme d'onda utilizzate.

Pulse Amplitude Modulation (PAM)

La modulazione in ampiezza utilizzata per l'UWB, è la classica modulazione bipolare nella quale i due bit 0 e 1 possono essere rappresentati mediane due impulsi gaussiani antipodali. Il segnale trasmesso sarà quindi [19]:

$$x(t) = d_j \cdot w_{tr}(t) \tag{1.7}$$

dove $w_{tr}(t)$ è il segnale UWB, invece j è il bit trasmesso (0 o 1) mentre:

$$d_j = \begin{cases} 1, & \text{se } j = 1 \\ -1, & \text{se } j = 0 \end{cases} \tag{1.8}$$

On-Off Keying

Questa modulazione è simile alla PAM, ciò che cambia nel segnale trasmesso è la codifica stessa; infatti in questo caso se si necessita del bit 0, non viene trasmesso nulla (a differenza della modulazione PAM nella quale veniva trasmesso un impulso di segno opposto rispetto all'altro). Per cui si ha[19]:

$$d_j = \begin{cases} 1, & \text{se } j = 1 \\ 0, & \text{se } j = 0 \end{cases} \tag{1.9}$$

Pulse Position Modulation (PPM)

In questo caso l'informazione è trasmessa modificando la posizione degli impulsi trasmessi sull'asse del tempo. Il bit che viene trasmesso quindi influenza la trasmissione del segnale UWB poichè, mentre il bit 0 viene

trasmesso con un impulso al "tempo zero", il bit 1 è ad una posizione distante δ (in termini temporali) dal "tempo zero".
La miglior prestazione teorica su un canale AWGN può essere raggiunta con impulsi non sovrapposti, ortogonali e con indice di modulazione $\delta > T_p$ (con T_p la durata dell'impulso). Nonostante ciò l'ottimo BER (*Bit Error Rate*) e il massimo *Data Rate* sono ottenibili se $\delta < T_p$. Di conseguenza è fondamentale trovare l'indice di modulazione ottimo per la forma d'onda e per l'applicazione utilizzata, al fine di avere il miglior comportamento del segnale utilizzato. Per questa analisi viene in aiuto l'autocorrelazione del segnale nella modulazione PPM, infatti il comportamento della cross-correlazione è fondamentale per il calcolo del δ ottimale[19]:

- L'autocorrelazione delle forme d'onda gaussiane, ha sia valori positivi che negativi, per questo il migliore BER si ha per $\delta < 1$.
- La minima autocorrelazione si verifica per valori di δ che corrispondono ai migliori valori del BER.

Per cui, una volta scelta la forma d'onda da utilizzare, il calcolo della cross-correlazione risulta fondamentale per il δ da utilizzare poichè esso può essere fissato a priori; infatti si possono notare i valori ottimali nella tabella 1.2:

Waveform	**Optimal δ**
Derivata seconda	$0.292683 \cdot T_p$
Derivata terza	$0.243902 \cdot T_p$
Derivata quarta	$0.219512 \cdot T_p$
Derivata quinta	$0.195122 \cdot T_p$

Tabella 1.2: Valori ottimi di δ da utilizzare in base alla forma d'onda scelta per la modulazione PPM in canale AWGN [19]

1.2.4 *Spectrum 'Spreading'*

Trasmettere periodicamente uno stesso impulso, implica avere in frequenza linee spettrali a multipli della frequenza di ripetizione dell'impulso stesso, per cui occorre fare in modo che, grazie alla modulazione ed altre tecniche, lo spettro venga modificato in modo tale che i picchi di energia vengano diffusi nelle frequenze. Per applicare questa tecnica

detta di *Spectrum 'Spreading'*, occorre procedere per *step*:

- Si inzia generando un treno di impulsi scegliendo la forma d'onda desiderata.
- Si applica una tecnica di randomizzazione al fine di "rompere" lo spettro del treno di impulsi.
- Viene applicata la modulazione al fine del trasporto dell'informazione.

I principali metodi di randomizzazione sono:

- *Time hopping (TH)*;
- *Direct Sequence (DS)*.

Time Hopping **UWB**

Questa tecnica utilizza una modulazione PPM con la tecnica TH, al fine di creare un segnale UWB che diffonda l'energia a radio frequenza dovuta ai picchi di energia del treno di impulsi. Per trovare l'ottimo *Time Shift* si calcola e si valutano le proprietà della cross-correlazione (vedi tabella 1.2), oltre al fatto che il *Time Hopping* viene fatto in modo pseudo casuale (*Pseudo Random* o PR) rendendo casuale la posizione degli impulsi nel tempo. L'utilizzo di un codice PR (quindi una tecnica PR-TH) permette di ridurre drasticamente le collisioni fra utente in sistema a multiplo accesso. L'unico inconveniente è che il ricevitore, per ricostruire la sequenza binaria trasmessa, deve conoscere il codice PR utilizzato dal trasmettitore. Si può quindi ricevare la frequenza dei simboli trasmessi (R_s o *Symbol Rate*) la quale dipenderà dal tempo di simbolo (T_s o *Symbol Time*) e di conseguenza dal numero di monocicli utilizzati (N_s) e dal *Time Hopping* (T_f); si avrà quindi[19]:

$$R_s = \frac{1}{T_s} = \frac{1}{N_s \cdot T_f}. \tag{1.10}$$

Accesso Multiplo con TH-UWB

Come già detto in precedenza, l'utilizzo di diversi codici pseudo random per ogni utente in trasmissione permette di avere il multiplo accesso al canale. In un determinato *frame* si avranno quindi un massimo numero di utenti che possono trasmettere senza creare interferenza fra loro, il quale è determinato dalla lunghezza del codice PR (vedi Formula 1.11):

$$N_u = 2^n - 1 \tag{1.11}$$

dove n è il numero di bit utilizzato nel generatore della sequenza PN. L'indice di modulazione (δ) invece è scelto come il valore minimo utilizzabile per evitare la sovrapposizione degli impulsi. In base al suo valore, si avrà che la lunghezza di ogni *time slot* (T_c) deve essere almeno di durata maggiore a quella vista in Formula 1.12:

$$T_c > 2 \cdot T_p + \delta \tag{1.12}$$

dove T_p è il tempo di ritardo utilizzato nella modulazione PPM.

Direct Sequence UWB

Nelle tecniche DS-SS (*Direct Sequence Spread Spectrum*) un codice *Pseudo Random* viene utilizzato per diffondere il bit di informazione come una sequenza di termini ridondanti (*chip*). Nel caso DS-UWB le forme d'onda dell'impulso prendono il posto dei *chip* ridondanti. Questa tecnica è stata studiata per segnali PAM (*Pulse Amplitude Modolation*), OOK (*On-Off Keying*) e PSM (*Phase Shift Modulation*). La modulazione PPM invece utilizza intrinsecamente una tecnica di *Time Hopping*, poiché il valore del bit (0 o 1) è dato dalla posizione dell'impulso utilizzzato nello slot di trasmissione. Quindi l'utilizzo di tecniche DS come un approccio di *spreading* porta ad una configurazione ibrida DS/TH.

1.2.5 Conclusioni

I segnali UWB possono essere distinti in impulsi radio (IR) e sistemi multibanda: i segnali IR sono potenzialmente a basso costo e in banda base, mentre i segnali multi banda possono utilizzare in modo più efficiente la banda a disposizione di questa tecnologia. In ogni caso le tecniche di modulazione da utilizzare devono includere delle tecniche

di randomizzazione dello spettro al fine di limitare le interferenze dovute alla trasmissione del treno di impulsi. Nonostante ciò, la scelta ottimale fra le tecniche TH o DS va fatta sempre in base agli schemi di modulazione, alla complessità dei sistemi ed alle condizioni iniziali del caso analizzato.

1.3 Radiolocalizzazione e UWB

La Radiolocalizzazione è una tecnica che richiede un'implementazione *Hardware* e *Software* al fine di avere un rilevamento *Wireless* di un intruso (*target)* presente all'interno di un area d'osservazione.
I metodi di localizzazione richiedono almeno una rappresentazione bidimensionale (latitudine e longitudine) e di punti di riferimento (antenne, stazioni radio base o satelliti) i quali hanno lo scopo di rilevare uno o più intrusi. La determinazione della posizione *target* può avvenire secondo vari metodi tramite misure:

- Temporali, in base al tempo di andata e ritorno del segnale dal trasmettitore al ricevitore (*Time of Arrival,* ***ToA***).
- Delle direzioni quindi in base gli angoli di arrivo (*Angle of Arrival* ***AoA***).
- Di potenza, mediante i modelli di propagazione (*Signal Strength Analysis*.
- Dei cammini multipli in base al calcolo del *Delay Spread*.

La tecnologia UWB è particolarmente portata alla Radiolocalizzazione per via dei numerosi vantaggi descritti in questo capitolo, relativi all'utilizzo di impulsi molto brevi e ad una banda molto grande. Tutto ciò permette un'elevata precisione di questi apparati nel *positioning* di eventuali *target* presenti in un'area.

1.4 Vantaggi e svantaggi dell'UWB

Oltre ai numerosi vantaggi di questa tecnologia, sono presenti alcuni svantaggi che per alcune applicazioni non rendono ottimale l'UWB.

Vantaggi:

- Elevata velocità di trasmissione (per alcuni apparati si può arrivare fino all'ordine dei Gbps) per applicazioni di comunicazione.

- Potenziale immunità ai cammini multipli; infatti, essendo utilizzato spesso in ambienti ristretti, il *Duty cycle* risulta talmente basso da permettere di riconoscere distintamente i vari cammini.
- Bassa complessità e basso costo.
- Risoluzione temporale molto spinta a causa della grande ampiezza di banda.
- Bassi consumi di potenza da parte di tutti i dispositivi (sia mobili che fissi).
- Elevata capacità dovuta alla banda molto grande; infatti, dal Teorema di Shannon si ha [15]:

$$C = B \log_2 \left(1 + \frac{S}{BN_0}\right). \tag{1.13}$$

  Si nota che conviene aumentare la banda piuttosto che la potenza, poiché quest'ultima dipende dal logaritmo in base due.

Svantaggi:

- Sono sistemi a corto raggio per via della stessa limitazione in potenza (si possono raggiungere anche distanze di centinaia di metri ma si riduce notevolmente la *BitRate*).
- Distorsione del segnale a causa dell'ampiezza di banda elevata.
- Sincronizzazione problematica dovuta alla breve durata degli impulsi.
- Distorsione introdotta dagli apparati stessi poiché è difficile creare filtri, amplificatori, antenne o altri strumenti che abbiano un comportamento costante su bande così elevate.
- Possibili interferenze dannose con altri sistemi che magari, per un certo periodo di tempo, non rispettano le specifiche della maschera di emissione (vista in Figura 1.2).

Capitolo 2

Descrizione del sistema Radar

Lo scenario standard di un sistema di localizzazione passiva UWB ha come obiettivo quello di rilevare, localizzare ed inseguire uno o più *target* in movimento in un'area.
Nel caso preso in esame di un Radar multistaico si ha: un trasmettitore (TX) ed almeno tre nodi ricevitori (RX); è quindi necessario descrivere innanzitutto il sistema come tipologia di *Radar* e successivamente le fasi di funzionamento[8] [9].

2.1 Radar

Il Radar può essere classificato in modi diversi, tra i quali la distinzione più rilevante si ha per [11]:

- Radar Monostatico : se vi è una sola antenna sia per la trasmissione che per la ricezione. Spesso però non è molto performante avere la stessa antenna sia per trasmettere che ricevere sia in termini di precisione, che di resistenza ai guasti.

- Radar Bistatico : nel caso in cui la distanza tra trasmettitore e ricevitore è comparabile con quella tra trasmettitore e *target*. In questo caso si ha maggior precisione rispetto al caso monostatico, anche se è necessaria una sincronizzazione fra TX e RX.

- Radar Multistatico : sistemi composti da più ricevitori e trasmettitori (di solito un TX con svariati RX oppure un RX con più TX). Questo sistema riduce sensibilmente il problema dei cammini multipli, del *fading* e dell'eliminazione del *clutter*, aumenta la

sensibilità, anche se comunque necessita di una buona sincronizzazione di tutti gli apparati.

2.2 Radar UWB

Come già detto in precedenza, la brevità degli impulsi generati e l'elevata banda, permettono all'*Ultra Wideband* di avere una precisione molto alta per la localizzazione. Lo scenario utilizzato anche nelle simulazioni di questo elaborato, considera un TX e tre RX (quindi un Radar Multistatico), in particolare i ricevitori sono collegati ad un *Fusion Center* il quale ha il compito di elaborare i dati ricevuti (vedi Figura 2.1).

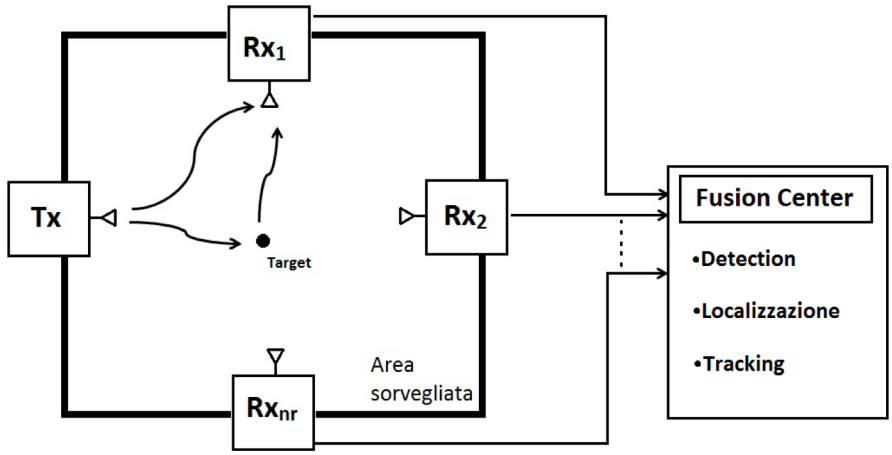


Figura 2.1: Scenario che rappresenta il radar multistatico UWB utilizzato nelle simulazioni e nelle misure sperimentali [3].

Tra le forme d'onda più utilizzate dal trasmettitore per il segnale UWB, vi sono i monocicli Gaussiani (che, come visto nel capitolo precedente sono i più performanti in termini di banda, vedi figura 1.6), mentre i ricevitori utilizzano la tecnica *ToA* per stima del tempo di arrivo del cammino diretto (vedi Capitolo 1.3); infine tutto viene elaborato dal *Fusion Center* il quale deve gestire:

- Detection;

- Localizzazione;
- Tracking.

2.3 Funzionamento del Radar UWB

Successivamente all'invio del segnale da parte del TX, vi sono una serie di passaggi che il ricevitore e il *Fusion Center* dovranno compiere[16] [8]:

- Stimare il tempo di arrivo mediante la tecnica *ToA*;
- Rimuovere il *Clutter* (ossia quei segnali ricevuti dal Radar che costituiscono l'ambiente e gli oggetti fissi nell'ambiente di rilevazione);
- Generare una mappa locale (*soft-valued map*);
- *Imaging*: creazione della mappa globale;
- *Detection*;
- *Tracking*;

Le prime tre operazioni vengo eseguite dal singolo ricevitore, mentre le ultime tre direttamente dal *Fusion Center*.
Il ricevitore quindi riceve impulso associato al cammino diretto dal TX, l'eco del *Clutter* e l'eventuale eco di un *target* che sarà ritardato rispetto al cammino diretto, poiché esso è frutto di una riflessione e dello scattering sull'intruso nell'area di rilevazione. A questo punto si considera il sistema come un insieme di Radar bistatici composti dalle coppie TX-RX (vedi Figura 2.2).

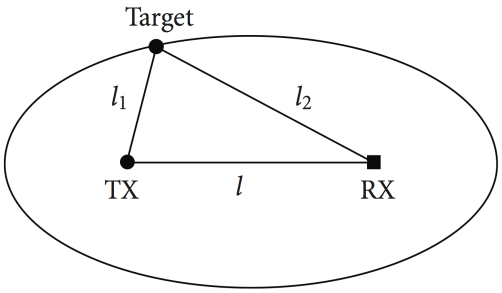


Figura 2.2: Ellisse del ToA che rappresenta una posizione possibile del target nell'area coperta dalla coppia TX-RX

.

La stima del ToA permetterà quindi il calcolo delle distanze l_1, l_2 e l, fondamentali per la localizzazione del *target* (Figura 2.2); infatti, poiché TX e RX sono i fuochi di quest'ellisse, da questi pochi dati è possibile trovare l'informazione sulla distanza:

$$l_1 + l_2 = ToA \cdot c. \tag{2.1}$$

In realtà l'ellisse è curva equi-ToA nella quale può essere presente il *target* in base all'eco ricevuto. L'ovale di cassini è invece determinato dai livelli di potenza riscontrati dal ricevitore nel caso di *scattering* del segnale trasmesso, sempre dovuto alla presenza di un intruso.

2.3.1 Analisi della Potenza

Per radar bistatici il RX può quindi calcolare due potenze fondamentali per la localizzazione, ricavabili dalle formule di *Friis*:

- Potenza ricevuta dovuta al cammino diretto:

$$P_{r-NB}^{direct} = \frac{P_t G_t G_r \lambda^2}{l^2 (4\pi)^2} \tag{2.2}$$

  dove P_t e G_t sono rispettivamente la potenza e il guadagno trasmessi, G_r è il guadagno in ricezione, λ è la lunghezza d'onda e l è la distanza TX-RX.

- Potenza ricevuta dovuta alla riflessione sul *target*, la quale dipende dalla *Radar Cross Section* (σ):

$$\sigma = 4\pi l_2^2 \frac{P_s}{P_i} \tag{2.3}$$

dove σ considera come il target re-irradia (P_s è la densità di potenza ricevuta dal ricevitore arrivata dal target) in base alla densità di potenza incidente su esso (P_i). Di conseguenza si può descrivere la potenza ricevuta da RX per via del target:

$$P_{r-NB}^{target} = \frac{P_t G_t G_r \lambda^2 \sigma}{(l_1 \cdot l_2)^2 (4\pi)^3}. \tag{2.4}$$

Questa trattazione però può essere applicata per sistemi a banda stretta (*narrowband*) [8], per cui, per avere una rappresentazione consona all'UWB ed in particolare alla sua elevata larghezza di banda, bisogna generalizzare le relazioni (2.2) e (2.4). É quindi necessario estendere l'analisi integrando le potenze ottenute con l'equazione di *Friis* nelle frequenze, fra gli estremi della banda dell'UWB e fare alcune semplificazioni [8]. Si ottengono quindi le potenze:

$$P_{r-UWB}^{direct} = \frac{S_t G_t G_r c^2}{l^2 (4\pi)^2} \cdot \left(\frac{1}{f_l} - \frac{1}{f_l + B} \right) \tag{2.5}$$

$$P_{r-UWB}^{target} = \frac{S_t G_t G_r c^2 \sigma c^2}{(l_1 \cdot l_2)^2 (4\pi)^3} \cdot \left(\frac{1}{f_l} - \frac{1}{f_l + B} \right) \tag{2.6}$$

dove appunto f_l e $f_l + B$ sono gli estremi della banda considerata.

2.3.2 Analisi del rapporto Segnale-Rumore (SNR)

Per definire lo scenario delle coppie TX-RX radar bistatici, bisogna considerare l'analisi appena conclusa (Paragrafo 2.3.1) e quella relativa al rapporto segnale-rumore (SNR), le quali permettono di descrivere l'ovale di Cassini massimo e l'ellisse minima (che sono gli estremi entro i quali può essere rilevato un eventuale intruso).
L'SNR è definito come:

$$SNR = \frac{N_s \cdot P_{r-UWB}^{target}}{N_0 \cdot PRF} \tag{2.7}$$

dove N_0 è la densità spettrale di potenza monolatera e N_s è il guadagno di processo; PRF (*Pulse Repetition Frequency*) è invece la frequenza con la quale si ripete l'impulso UWB.
Affinché un *target* venga individuato nell'area di sorveglianza, bisogna che il rapporto segnale-rumore sia maggiore di un certo valore di soglia:

$$SNR \geq SNR_{th}. \tag{2.8}$$

Di conseguenza anche la potenza ricevuta dovuta alla riflessione del *target* deve soddisfare la relazione:

$$P_{r-UWB}^{target} \geq P_{th} \tag{2.9}$$

dove la potenza con valore di soglia è definita come:

$$P_{th} = \frac{SNR_{th} N_0 PRF}{N_s}. \tag{2.10}$$

2.3.3 *Ovale di Cassini* massima e minima ellisse

É ora possibile definire il valore massimo del prodotto fra le distanze l_1 e l_2, affinchè un *target* sia rilevabile:

$$(l_1 \cdot l_2)^* = \sqrt{\frac{S_t G_t G_r \sigma_c^2}{P_{th}(4\pi)^3} \cdot \left(\frac{1}{f_l} - \frac{1}{f_l + B}\right)}. \tag{2.11}$$

Quindi la prima condizione che un *target* deve assolvere al fine di essere rilevato, è quella di stare all'interno del massimo ovale definito dalla relazione (2.11).
La seconda condizione è invece sul minimo valore che $l_1 + l_2$ può assumere. Il ricevitore considera il segnale del cammino diretto e del cammino riflesso dall'intruso, però possono esserci alcuni casi in cui RX non li distingue. Ciò può avvenire a causa della differenza fra la stima dei due tempi, ossia quando essa è troppo piccola. Si definisce quindi γ come il più piccolo valore per il quale il ricevitore riesce ancora a distinguere il cammino diretto da quello riflesso dalla presenza del *target*:

$$l_1 + l_2 \geq l + \gamma \cdot c. \tag{2.12}$$

L'ellisse con parametri $l+\gamma \cdot c$ è quindi la minima ellisse dentro la quale un target non è visualizzabile.
La conclusione è quindi quella visibile nella Figura 2.3, la quale denota il fatto che un *target* può essere localizzato da una coppia bistatica all'interno di una regione, fra la minima ellisse e il massimo ovale di Cassini:

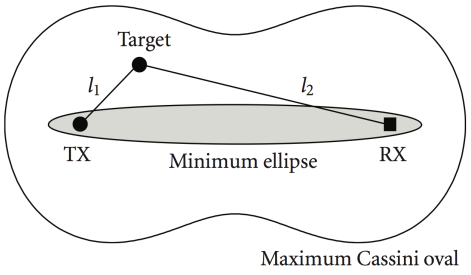


Figura 2.3: Rappresentazione dell'ovale di Cassini e della minima ellisse.

2.4 Stima del ToA

L'analisi vista in precedenza (Figura 2.2) rappresenta un'approssimazione di ciò che avviene nella realtà, che però permette di semplificare e spiegare l'utilizzo della tecnica di stima del ToA. Infatti l'intero sistema non conosce a priori, con errore nullo, il ToA dell'eventuale *target*, quindi deve essere considerata una stima, più o meno precisa, del tempo che il segnale impiega ad eseguire il percorso TX-intruso-RX. Una perfetta stima di questo periodo, implica una stima perfetta delle somme $l_1 + l_2$, per cui un'incertezza su questo valore significa non avere un'ellisse (Figura 2.2), ma un'area ad anello entro la quale si può trovare il *target* (vedi Figura 2.4).

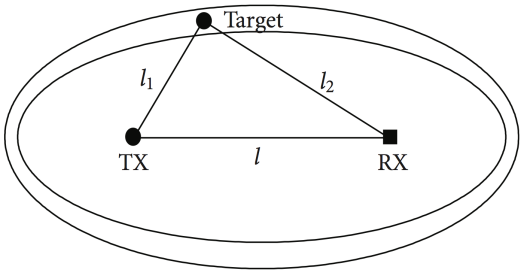


Figura 2.4: Rappresentazione dell'anello in cui si trova il *target* a causa della stima del ToA

.

La larghezza dell'anello e quindi l'incertezza, dipendono dal rapporto segnale-rumore che si ha a lato ricevitore: più l'SNR è alto, più stretto è l'anello e quindi minore è la possibilità di commettere un errore sulla stima della posizione del *target*.

2.5 Radar Multistatico

Per un sistema multistatico vengono definiti alcuni limiti in potenza e in frequenza di ripetizione dell'impulso (PRF). Di seguito verranno descritti [8]:

$$(l_1 \cdot l_2)_{max} = \max_{i=1,\dots,N-1} \{(l_1 \cdot l_2)_{max,i}\}. \tag{2.13}$$

$$P_{min} = \max_{i=1,\dots,N-1} \{S_{t_{min},i}\}. \tag{2.14}$$

Per cui le equazioni (2.13) e (2.14) fissano dei limiti al sistema in termini di potenza, oltre ai quali non si può più avere la *detection* di un eventuale intruso. Se $P_t \geq P_{min}$ (dove $S_{t_{min}}$ è la minima densità spettrale di potenza trasmessa), allora l'area di sorveglianza è inclusa nel massimo *ovale di Cassini*; ossia ogni coppia di radar bistatici TX-RX può rilevare l'intruso nell'area (a meno che all'interno della minima ellisse 2.3).
Si considera quindi $(l_1 + l_2)^*$ come il massimo valore di $l_1 + l_2$ per il quale vale $(l_1 \cdot l_2) \leq (l_1 \cdot l_2)^*$, da cui si può ricavare il massimo tempo

di propagazione per un impulso riflesso (2.15):

$$\tau = \frac{(l_1 + l_2)^*}{c}. \tag{2.15}$$

Si possono quindi impostare le condizioni sul PRF (già visto nell'equazione del rapporto segnale-rumore 2.7) 2.16, 2.17:

$$(l_1 + l_2)_{max} = \max_{i=1,\ldots,N-1} \{(l_1 + l_2)_{max,i}\}. \tag{2.16}$$

$$PRF_{max} = \frac{c}{(l_1 + l_2)_{max}}. \tag{2.17}$$

Capitolo 3

Algoritmi CFAR e di *Clustering*

Un problema fondamentale affrontato in questo elaborato, è quello relativo al *Clustering*; infatti in reti radar composte da un trasmettitore e multipli ricevitori, possono esservi problemi relativi a misure multiple generate dallo stesso *target*.
Ogni ricevitore stima il tempo di arrivo dei *target*, rimuove il *Clutter*, ottiene un vettore di campioni *Soft* e confronta il tutto con una soglia detta *Constant False Alarm Rate* (o CFAR) ottenendo un vettore binario. Il problema riscontrato sta nel fatto che questa matrice composta da '0' (ossia *target* non rilevato) e '1' (ossia *target* rilevato) è spesso composta da un *burst* di '1', quindi una serie di valori che spesso sono riferiti ad un solo ed unico intruso. Ciò è causato dalla non perfetta rimozione del *Clutter*, da errori nella stima del TOA e dal fatto che il *target* non è puntiforme.
L'obiettivo quindi delle simulazioni e del codice scritto per quest'elaborato, è quello di fare il *Clustering* dei *Burst* di '1' poiché questi rappresentano lo stesso target.

3.1 *CFAR*

Prima di introdurre gli algoritmi analizzati in questo elaborato, bisogna definire cos'è e come viene utilizzata una soglia fondamentale per le simulazioni: il CFAR.
Questo capitolo considera radar multistatici UWB nei quali la *detection* e la localizzazione sono state realizzate utilizzando rispettivamente: la soglia del CFAR e il metodo della trilaterazione. In prima istanza, il rivelatore stima il tempo di arrivo (ToA) dei eventuali *target* presenti nell'area, successivamente si trova l'interesezione delle ellissi ricavate

dai ToA, calcolato dai ricevitori, al fine di ottenere le misure in ingresso al blocco di *tracking*. L'approccio basato sulla tecnica di *detection* mediante l'utilizzo di *detector CFAR*, stima la posizione dell'eventuale *target*, paragonando il valore del campione rilevato con la soglia CFAR. A seguito di una Clusterizzazione bidimensionale, i *cluster* di cardinalità più alta, saranno quelli in cui si stima la posizione dei target.

3.1.1 *CA-CFAR Detector* e Metodo Diretto di Localizzazione

Il *Cell Averaging* CFAR è la versione più semplice e comune fra le tradizionali tecniche di *detection* (vedi Figura 3.1) [17]. Si considera $r_{\mathrm{k,n}}$ (vedi Figura 3.1) come l'n-esimo campione dei segnali ricevuti da uno degli N_{R} ricevitori, ottenuto mediando gli N_{S} segnali ricevuti, relativi agli impulsi trasmessi nel *frame* k-esimo. Per ogni ricevitore si ha quindi un vettore di N campioni che è passato dal *detector* CFAR. Nel *detector* CA-CFAR il livello di soglia viene calcolato attraverso la stima del livello di rumore di fondo nei campioni vicini a quelli considerati (in questo caso vicini a $r_{\mathrm{k,n}}$). Per questo vengono impostati: il numero di campioni di guardia (N_g) da non considerare attorno al campione analizzato e invece il numero di campioni da confrontare con quello considerato (N_{ref}). Per cui il CA-CFAR funziona considerando un blocco di campioni vicini e calcolando il livello di potenza media di rumore, scartando i campioni immediatamente adiecenti come livello di guardia. Importante è quindi sapere cosa c'è in uscita dal *detector*: se infatti la soglia viene superata (ossia si stima la presenza di un eventuale *target*), il campione *soft* viene quanizzato a '1', altrimenti, se la soglia non viene raggiunta (stima della non presenza di un eventuale intruso), si pone a '0' il valore del vettore di uscita.

Il metodo diretto di localizzazione è l'approccio più utilizzato per la misura della posizione di un target nel caso di radar UWB basati sulla stima del ToA. Questa tecnica tradizionale di localizzazione è chiamata trilaterazione poiché utilizza triplette di ricevitori i quali stimano il ToA degli eventuali target presenti nell'area. Dopo aver fatto il calcolo diretto per tutte le possibili triplette di ricevitori e considerandoo tutte le possibili combinazioni di ToA, si avranno un numero di misure relative alla localizzazione dei target per ogni singola combinazione. Se un numero di combinazioni di ricevitori ha le soluzioni molto vicine fra loro, queste vengono mantenute, le altre invece vengono scartate, al fine di ridurre il numero di falsi target considerati. La soluzione finale sarà la media delle soluzioni vicine fra loro.

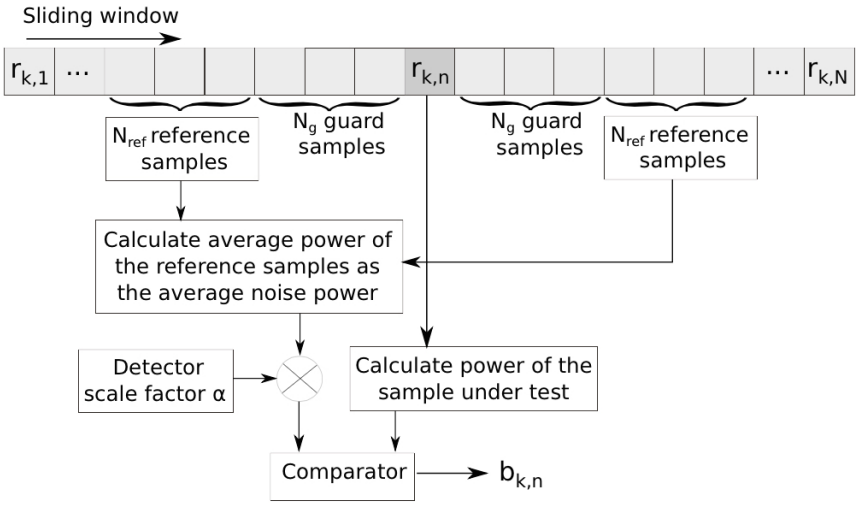


Figura 3.1: Rappresentazione del funzionamento del CA-CFAR *detector*[17].

3.1.2 CA-CFAR e Median Filtering

Dall'analisi precedente si capisce come sia fondamentale settare la soglia del CFAR in modo consono, poichè questa rende più o meno preciso il sistema; infatti, riducendo questo valore, cala il numero delle *missed detection* (ossia la probabilità di non rilevare un eventuale intruso), ma aumentano i falsi allarmi (ossia la probabilità di rilevare un target quando in realtà esso non è presente). Nel caso in cui la soglia venga impostata ad un valore troppo basso, i falsi allarmi, disegnati su un'immagine bidimensionale che rappresenta il tempo di propagazione nelle scansioni per ogni ricevitore, appaiono come un rumore *"salt and pepper"*[17]. Ossia si ha una rappresentazione mediante immagini e quindi con la suddivisione dell'area d'osservazione in pixel. Per risolvere questo problema si utilizzano dei *Median filter* i quali elaborano l'immagine così che ogni pixel venga rimpiazzato dal valore medio dei pixel 3×3 vicini. Questo tipo di processamento dell'immagine permette di ridurre la soglia del CFAR senza preoccuparsi dei falsi allarmi, poichè il *Median filter* è in grado di ridurli [17]. In questo elaborato, si è scelto di non utilizzare il *Median filter*, al fine di settare direttamente una soglia che riduca il numero di falsi allarmi nel sistema sia simulato, che derivante dalle misure sperimentali.

3.1.3 CFAR nell'elaborato

In questa tesi la soglia del CFAR è il valore che, sia nelle simulazioni, che nelle misure sperimentali, viene variato analizzando i dati forniti in uscita, al fine di trovare una soglia adeguata al tipo di elaborazione considerata. In particolare viene modificato parte dell'approccio utilizzato nell'analisi antecedente a questo elaborato (vista nei paragrafi 3.1.1 e 3.1.2, oppure nei lavori [17], [4] e [3]). Infatti di seguito verranno riportate le descrizioni degli algoritmi studiati e successivamente di quello implementato in questo elaborato, al fine di eliminare il *Median filter* ed avere una *clusterizzazione* dei punti che superano la soglia del CFAR e sono molto vicini fra di loro. Di conseguenza il CFAR, nelle simulazioni di questa tesi, viene utilizzato come soglia per diminuire il numero di falsi allarmi, assieme agli algoritmi di clusterizzazione mono e bidimensionale (vedi capitoli successivi). Si può quindi vedere dalla Figura 3.2 uno schema della simulazione effettuata, nel quale si andrà appunto a modificare la parte relativa al *Clustering* dei punti del *target* nell'area di sorveglianza.

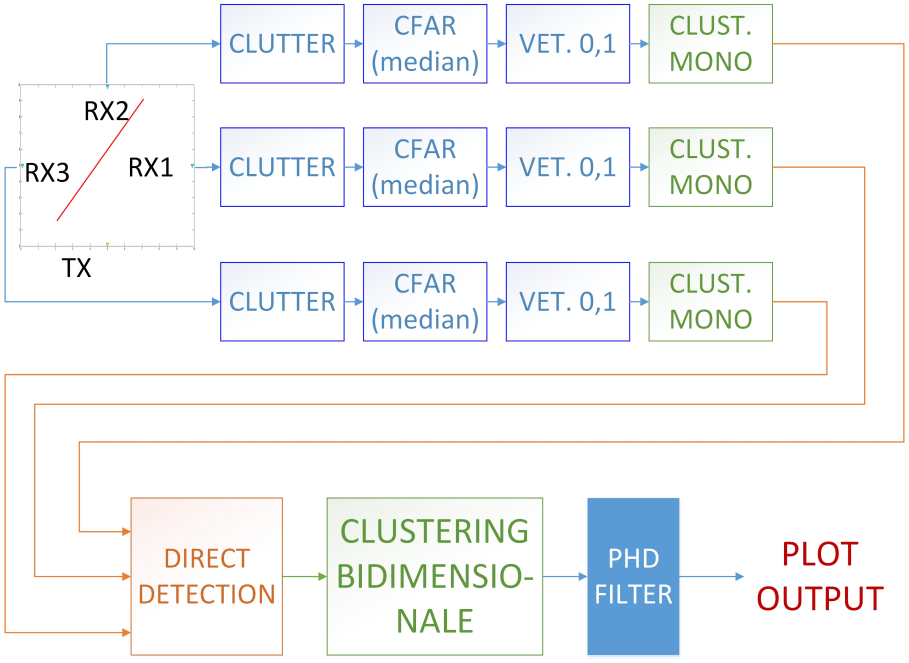


Figura 3.2: Schema a blocchi che rappresenta il funzionamento del simulatore.

3.2 Algoritmi di Clustering

3.2.1 Algoritmo K-Means

K-Means è un algoritmo utilizzato per suddividere l'insieme dei dati in K gruppi sulla base delle loro caratteristiche (nel caso in analisi di distanza minima) [1]. È un algoritmo iterativo che crea K insiemi e assegna i punti di ingresso ad ognuno di essi, calcola il punto medio (centroide) per ogni insieme, genera le nuove K partizioni assegnando i punti più vicini al centroide ed infine itera tutto al fine di ottenere una soluzione ottimale.
Il problema fondamentale di questo algoritmo sta nel fatto che il valore di K deve essere impostato a priori e non può essere trovato dai valori dati in ingresso.

3.2.2 Algoritmo Hierarchical Clustering

Come si intende dal nome stesso, il *Clustering* gerarchico (HC) permette di costruire dei raggruppamenti utilizzando uno schema ad albero (o *dendrogramma*). In questo caso esistono algoritmi di *Clustering* sia di tipo *top-buttom*, che *bottom-up*: i primi considerano un *cluster* globale per poi frazionarlo; i secondi invece, partono dai singoli punti considerandoli tutti come *cluster*, per poi successivamente raggrupparli fra loro. In questa tesi è stato utilizzato un approccio *bottom-up*, implementando un algoritmo in *Matlab*, mediante tre funzioni distinte che eseguono altrettante fasi [2]:

1. Si trovano le proprietà in comune fra i punti, ossia si calcola la distanza minima fra le coppie che permette loro di essere nello stesso *Cluster*.

2. Vengono raggruppati gli oggetti in modo da poterli rappresentare mediante un grafico gerarchico ad albero.

3. Viene scelto dove tagliare l'albero al fine di creare i *Cluster* desiderati.

Il problema di questo algoritmo è appunto dove tagliare il dendrogramma poiché può accadere che venga scelto un numero non consono al *Clustering* voluto e quindi vengano generati più o meno *Cluster* rispetto al caso ottimale.

3.2.3 Algoritmo Implementato

In relazione all'analisi dei problemi riscontrati precedentemente, in questa tesi è stato implementato un algoritmo che prende spunto sia dal K-means che dall' HC, scegliendo un approccio agglomerativo (ossia in partenza ogni punto è un *Cluster*, poi, in modo iterativo, avviene il raggruppamento). Questo permette di avere una *'Clusterizzazione'* senza conoscere a priori il numero di raggruppamenti presenti, semplicemente impostando una soglia sulla distanza minima fra i campioni di '1' vicini fra loro.

Si considera un vettore di campioni dove, sempre in base alla soglia impostata del CFAR:

- '1' rappresenta la presenza di un target nello spazio della simulazione
- '0' invece è la rappresentazione di una scelta negativa alla presenza del target

Il funzionamento può essere suddiviso in varie fasi e spiegato semplicemente mediante un vettore di partenza preso come esempio:

$$\begin{bmatrix}0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\end{bmatrix} \tag{3.1}$$

- **Fase 1 :** *Matrice delle distanze*
  Analizzando il vettore, si notano 5 elementi ('1'), dei quali è importante la distanza (intesa come numero di campioni fra un '1' e l'altro) fra loro poiché sarà il fattore fondamentale impostato per effettuare o meno il *Clustering*. Per cui, calcolata la distanza fra tutti gli '1', si genera una matrice $N \times N$ (dove N è il numero di '1', quindi nel nostro caso 5) delle distanze fra essi:

$$\begin{bmatrix}0 & 1 & 4 & 6 & 8\\ 1 & 0 & 3 & 5 & 7\\ 4 & 3 & 0 & 2 & 4\\ 6 & 5 & 2 & 0 & 2\\ 8 & 7 & 4 & 2 & 0\end{bmatrix} \tag{3.2}$$

  Gli indici delle colonne e delle righe della matrice delle distanze indicano quindi gli '1' considerati nel calcolo della distanza fra loro; ad esempio: l'elemento alla riga 2 e colonna 3 ($[2, 3]$), che ha

valore 3, indica che il secondo '1' è distante 3 posizioni dal terzo. Quest'analisi può essere semplificata con la rappresentazione tramite dendrogramma (vedi Figura 3.3), la quale permette una descrizione più intuitiva del *Clustering*; infatti, dall'esempio considerato, si parte da un vettore di cinque elementi (che sono le posizione degli '1' nel vettore di partenza), il quale verrà via via ridotto a seconda dei raggruppamenti effettuati:

$$\begin{bmatrix} 4 & 5 & 8 & 10 & 12 \end{bmatrix} \tag{3.3}$$

- **Fase 2 :** *Ricerca del minimo e valutazione della soglia*
  Si passa ora alla ricerca della minima distanza che c'è fra una coppia di '1', ossia il numero più piccolo diverso da zero presente nella matrice. Nel caso preso in considerazione, il minimo valore si trova in posizione $[1, 2]$ (e speculare), ciò significa che il primo e il secondo '1' del vettore di partenza, sono quelli a distanza minima.

  Se il valore di minimo sarà sotto una certa soglia impostata precedentemente (nel caso preso in analisi questa soglia è appunto 5 o 6 a seconda delle simulazioni effettuate), allora si procede con le fasi successive, altrimenti l'algoritmo termina.
  La soglia è un parametro fondamentale poiché esprime il ritardo oltre al quale è possibile considerare un '1' come generato da un nuovo target e non come un'eco o comunque un errore di ripetizione derivante dal medesimo *target*.

- **Fase 3 :** *Marging del dendrogramma e modifica del vettore di partenza*
  Trovato il minimo sotto soglia, si passa al *Marging* del dendrogramma che, nell'esempio, porta ad accoppiare il primo '1' con il secondo (vedi vettore 3.4).

$$\begin{bmatrix} (4,5) & 8 & 10 & 12 \end{bmatrix} \tag{3.4}$$

  Questo significa che si sta raggruppando la prima colonna/riga e la seconda colonna/riga della matrice delle distanze (vedi matrice 3.2) in un unico valore; di conseguenza si andrà a modificare il vettore di partenza sostituendo ai due '1', un unico valore che

sarà posto a distanza media fra i due (se la distanza media non è intera, come per questa prima iterazione, allora è stato scelto l'intero più grande):

$$\begin{bmatrix}0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\end{bmatrix} \tag{3.5}$$

- **Fase 4 :** *Iterazione*
  A questo punto, con il nuovo vettore di partenza, si ripetono le varie fasi a partire dalla **Fase 1** fino a che: o si esce dalla **Fase 2** poiché il valore minimo è sopra soglia, o si è giunti alla *'Clusterizzazione'* di tutti gli elementi per cui il dendrogramma è composto da un unico elemento/*'Cluster'* (come avviene nel caso preso in esame). In seguito vengono rappresentate tutte le iterazioni dell'esempio considerato al fine di spiegare al meglio il funzionamento dell'algoritmo proposto in questo elaborato di tesi.

  Iterazione 1
  Alla prima iterazione viene ricalcolata la matrice delle distanze:

$$\begin{bmatrix}0 & 3 & 5 & 7\\ 3 & 0 & 2 & 4\\ 5 & 2 & 0 & 2\\ 7 & 4 & 2 & 0\end{bmatrix} \tag{3.6}$$

  Successivamente si calcola il minimo (il quale è sotto soglia quindi si può procedere con le varie Fasi) e si modifica il dendrogramma:

$$\begin{bmatrix}(4,5)\ 8\ (10,12)\end{bmatrix} \tag{3.7}$$

  Di conseguenza anche il nuovo vettore di partenza sarà:

$$\begin{bmatrix}0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\end{bmatrix} \tag{3.8}$$

  Iterazione 2
  Questa iterazione è identica alla precedente a parte i valori aggiornati delle matrici:

$$\begin{bmatrix}0 & 3 & 6\\ 3 & 0 & 3\\ 6 & 3 & 0\end{bmatrix} \tag{3.9}$$

$$\big[(4,5)\ (8,10,12)\big] \tag{3.10}$$

$$\big[0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\big] \tag{3.11}$$

Iterazione 3
Questa iterazione è identica alle precedenti a parte i valori aggiornati delle matrici:

$$\begin{bmatrix} 0 & 5 \\ 5 & 0 \end{bmatrix} \tag{3.12}$$

$$\big[(4,5,8,10,12)\big] \tag{3.13}$$

$$\big[0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\big] \tag{3.14}$$

Iterazione 4
A questo punto si ha l'ultima iterazione poiché è stata raggiunta la *'Clusterizzazione'*: la matrice delle distanze è formata da un unico elemento (per via del fatto che il vettore di partenza è composto da un solo '1'):

$$\big[0\big] \tag{3.15}$$

Come si può notare in Figura 3.3, sono stati raggruppati tutti gli '1' del vettore in un unico *cluster*. Ovviamente vi è la possibilità che ciò non avvenga; ciò è dovuto al fatto che la distanza tra due '1' è tale da superare la soglia di *clustering*, per cui sono talmente distanti da essere considerati come due *cluster* differenti (cosa che potrebbe accadere in presenza di più *target* nell'area di osservazione).
Tornando all'esempio di prima, il dendrogramma finale è il seguente:

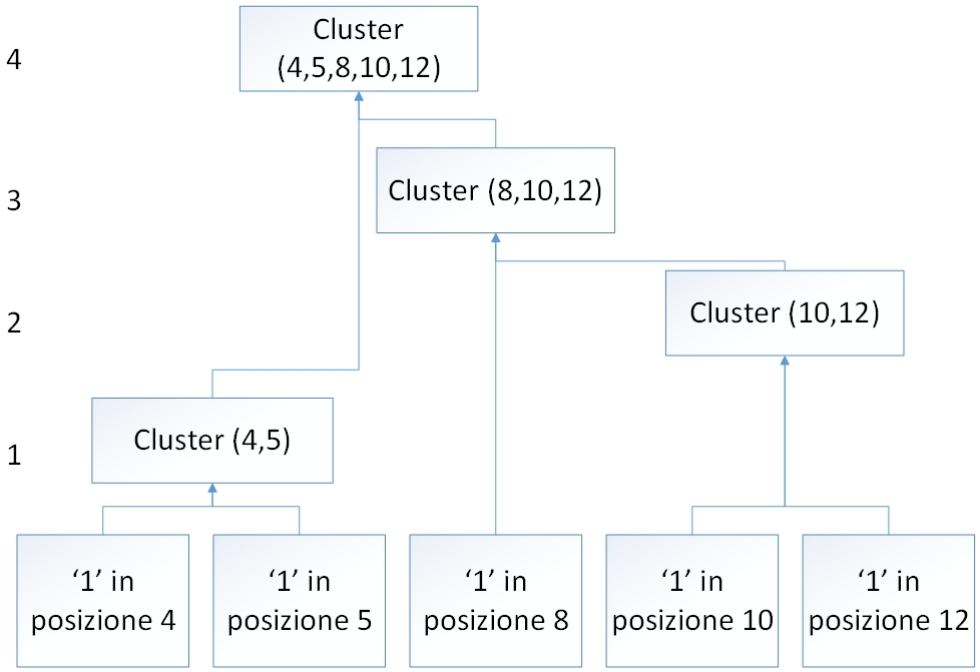


Figura 3.3: Dendrogramma d’esempio

Capitolo 4

Codice

L'algoritmo preso in considerazione per questa tesi (vedi Capitolo 3), supera l'approccio euristico utilizzato negli elaborati precedentemente sviluppati([3] e [4]) e propone un metodo di *Clustering* che cerca di minimizzare i falsi allarmi del sistema.
La *Clusterizzazione* avviene in due momenti distinti nel codice:

- ***Cluster Monodimensionale*** : Per ciascun ricevitore, non appena il vettore dei campioni *soft* viene trasformato '0' e '1' mediante la soglia del CFAR.
- ***Cluster Bidimensionale*** : Alla fine del processo di *Detection* sulle coppie (x, y) che si riferiscono alle misure in ingresso al blocco di *tracking*.

4.1 Cluster Monodimensionale

Il primo *step* nello sviluppo del codice Matlab, è stato quello di suddividere le varie fasi dell'algoritmo in un insieme di funzioni richiamate da un *Main*. Lo scopo di tutto ciò è quello di ridurre il tempo di computazione ed elaborazione del programma stesso.
Per cui il codice, nel caso Monodimensionale, viene descritto dalle varie funzioni create per l'occasione:

4.1.1 Funzione *'VettFind'*

Questa funzione prende in ingresso la matrice dei campioni confrontati con il CFAR e la analizza al fine di trovare la posizione del target nella matrice stessa. Infatti, grazie alla funzione *'find'* di Matlab, è possibile ottenere due vettori (I e J) che sono rispettivamente la riga e la colonna a cui si trova un '1' nella matrice di ingresso.

4.1.2 Funzione *'MatDistv02'*

Questa funzione fornisce in uscita l'insieme dei vettori (M) fondamentali per il *Clustering* degli '1' presenti nella matrice di ingresso. *MatDistv02* prende in ingresso i dati usciti da *VettFind* e seguendo di pari passo l'Algoritmo implementato in questa tesi (vedi Capitolo 3.2.3); infatti l'uscita di questa parte di codice, crea le matrici delle distanze viste nella *Fase 1*.

4.1.3 Funzione *'FindMinBlock'*

A questo punto l'obiettivo è trovare il minimo di ogni sotto blocco della matrice di ingresso (quindi di ogni scansione), di conseguenza si utilizza una matrice di appoggio ('MatApp'), nella quale vengono salvati tutti i valori diversi da zero (trovati con la funzione *'find'* e dei quali è nota la posizione nella matrice M tramite 'Col' e 'Rig'). Una volta completata la matrice di appoggio, si calcola il minimo valore di essa e l'indice a cui si trova il suddetto ('Index'). Tramite 'Index' è possibile risalire alla posizione di riga ('Rig') e colonna ('Col') fondamentali per le successive elaborazioni.

4.1.4 Funzione *'FindMinTotv03'*

Funzione che utilizza FindMinBlock e la itera con lo scopo di trovare tutti i minimi in tutte le dimensioni della matrice in ingresso; infatti M è una matrice multidimensionale, con tante dimensioni quante sono le scansioni alle quali si può associare una Matrice di distanza.

4.1.5 Funzione *'New_r_dv01'*

Questa funzione aggiorna la matrice iniziale di ingresso al fine di realizzare passo per passo la *'Clusterizzazione'* impostando una soglia ('thr') in base alla minima distanza che si vuole avere fra due '1' clusterizzati. In sostanza aggiorna i campioni iniziali con una prima clusterizzazione.

4.1.6 *Main*

Create tutte le funzioni principali per applicare l'Algoritmo alla simulazione in codice, viene scritto il *Main* il quale ha lo scopo di iterare tutte le funzioni realizzate. Di seguito viene riproposto parte di ciò che

è stato scritto nel caso della prima iterazione:

```
a=1;
thr=5;
i=1;
disp('Matrice iniziale');
[I1,J1] = VettFind (r_d,a);
[M1] = MatDistv02 (I1,J1);
[ValMinTot1,IndexTot1,ColTot1,RigTot1]=FindMinTotv03(M1);
[r_d,ValMinTot1]=New_r_dv01(ValMinTot1,I1,ColTot1,RigTot1,
    IndexTot1,r_d,a,J1);
go=find((ValMinTot1<=5)&(ValMinTot1>0));
if isempty(go)==1
    disp('Ho gia'' la clusterizzazione');
    break
else
    while i<=size(ValMinTot1,3)
        [I1,J1] = VettFind (r_d,a);
        [M1] = MatDistv02 (I1,J1);
        [ValMinTot1,IndexTot1,ColTot1,RigTot1]=FindMinTotv03(M1);
        [r_d,ValMinTot1]=New_r_dv0(ValMinTot1,I1,ColTot1,RigTot1,
            IndexTot1,r_d,a,J1);
        go=find((ValMinTot1<=5)&(ValMinTot1>0));
        if isempty(go)==1
            disp('Ho la Clusterizzazione :');
            break
        end
        i=i+1;
    end
    i=1;
end
a=a+1;
clear go;
```

Ovviamente per cambiare tutta la matrice di ingresso e soprattutto per non gravare sulla elaborazione di Matlab, questo codice viene ripetuto per ogni dimensione della matrice d'ingresso.

4.2 Cluster Bidimensionale

Il secondo *step* nella scrittura del codice per la *'Clusterizzazione'* è stato appunto quello di raggruppare punti nello spazio. Quindi un raccoglimento dei punti (x, y) i quali, in termini di distanza euclidea, sono vicini fra loro in base ad una soglia impostata in precedenza. L'approccio utilizzato nella stesura del codice è analogo a quello del caso monodimensionale, ossia sono state create delle funzioni di base che poi vengono richiamate da un *Main* per raggiungere lo scopo dell'Al-

goritmo utilizzato.

4.2.1 Funzione *'pdist2'*

Questa funzione permette di calcolare la distanza euclidea fra due punti nello spazio in modo veloce ed ottimizzato, poichè *pdist2* è già preimpostata in Matlab. In uscita si hanno quindi due matrici:

- Una relativa alle distanze di un punto verso tutti gli altri punti dello spazio (matrice D), per colonna e in ordine crescente (tramite l'opzione *'Smallest'*).
- L'altra relativa all'informazione fra i punti considerati; in sostanza spiega quali punti sono stati presi in considerazione nel calcolo della distanza, per quella determinata coordinata della matrice.

4.2.2 Funzioni *'FindMinBlock_2D', 'FindMinTotv03_2D'*

Sono funzioni analoghe a quelle utilizzate nel caso del *Cluster Monodimensionale* però sono state adattate al caso bidimensionale. Lo scopo è analogo ossia di trovare il minimo valore diverso da zero della matrice in ingresso.

4.2.3 Funzione *'New_matt_v01'*

Questa funzione considera le elaborazioni delle precedenti e restituisce la stessa matrice di ingresso (chiamata *alfa* nel codice, nella quale sono presenti ascissa e ordinata in colonna dei punti nello spazio) con la *'Clusterizzazione'* dei due punti più vicini in termini di distanze euclidea. Di conseguenza in uscita *alfa* avrà una riga in meno poiché due punti sono stati uniti in unico punto che sarà il punto medio fra i due.

4.2.4 *Main*

Il *Main* considera tutte le funzioni precedentemente create e le compone al fine di avere il *Clustering Bidimensionale*. Un semplice esempio utilizzato nelle simulazioni per comprendere il lavoro di questo algoritmo viene rappresentato nelle seguenti figure:

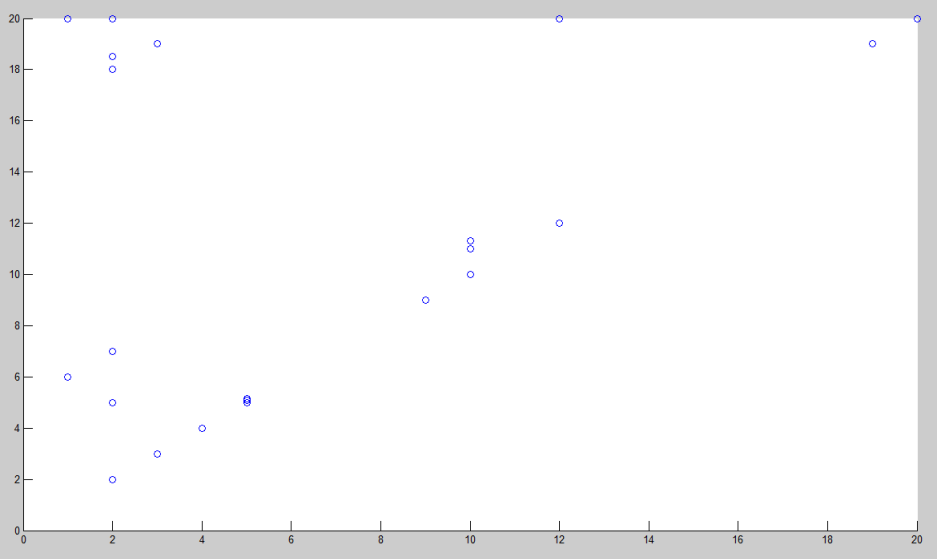


Figura 4.1: Simulazione di alcuni punti da *'Clusterizzare'* nello spazio.

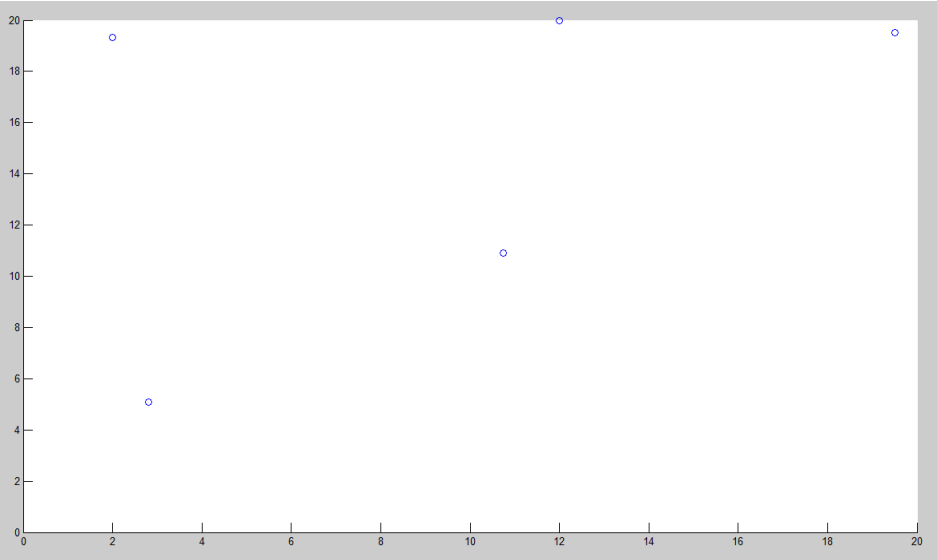


Figura 4.2: Dopo l'elaborazione dell'algoritmo di *Clustering*, si nota che i punti nello spazio (che prima erano circa una ventina) vengono sostituiti da soli 5 punti nello spazio grazie ad una specifica soglia preimpostata.

Capitolo 5

Simulazioni al Calcolatore

In questo capitolo verranno riportate alcune delle simulazioni eseguite in questo elaborato, realizzate al fine di avere un riscontro teorico dell'implementazione degli algoritmi studiati (vedi Capitolo 4).
Lo scenario utilizzato è quello già analizzato in Figura 5.1 proposto e già visto in precedenti articoli[9] [4] [17]:
Quindi si considera uno scenario di larghezza $[100 \times 100] metri$ nel quale

TX / RX	**Posizione (x,y)**
Trasmettitore	(0,50)
Ricevitore 1.1	(33,0)
Ricevitore 1.2	(66,0)
Ricevitore 2.1	(100,33)
Ricevitore 2.2	(100,66)
Ricevitore 3.1	(33,100)
Ricevitore 3.2	(66,100)

Tabella 5.1: Tabella delle posizione dei ricevitori e del trasmettitore nella quale sono riportate ascisse e ordinate di ognuno ([3])

.

sono posizionati sei ricevitori (due per ogni lato dell'area di sorveglianza) e infine un trasmettitore che ha il compito di inviare il segnale UWB a tutti i riceventi. Si nota dalla Figura 5.1 che il trasmettitore è rappresentato da un triangolo giallo, i ricevitori da uno azzurro, mentre in grigio sono presenti le minime ellissi entro le quali non si ha la *detection* (in relazione ad ogni coppia TX-RX di radar bistatico), mentre i pallini neri corrispondono alla riflessione del segnale sul terreno.
In alcune simulazioni, al fine di renderle più realistiche possibili, è utile considerare anche il *Clutter*; per questo spesso vengono aggiunti 100 punti riflettenti che sono rispettivamente collocati nell'area in modo

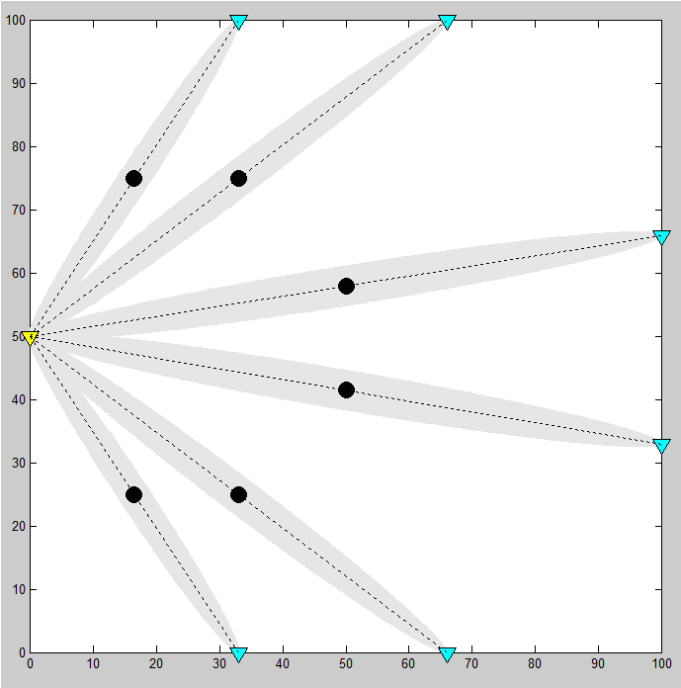


Figura 5.1: Scenario simulazione: si può notare a sinistra il trasmettitore mentre ai lati i tre ricevitori, composti da due antenne ciascuno.

uniforme (ma spesso, per semplificare le simulazioni, il *Clutter* viene rimosso).

5.1 Simulazione base

La simulazione utilizzata in partenza è quella relativa agli articoli [4] e [17], la quale considera il classico scenario in Figura 5.1, un numero scansioni pari a 600, *Clutter* statico a 100 punti, tre target.

Si nota come (Figura 5.2) il radar riesca a fare *detection* e *tracking* in modo abbastanza performante, grazie al fatto che vengono utilizzati due principali approcci alla risoluzione dei problemi delle *Miss detection* e dei *False Alarm* già descritti nel Capitolo 3:

- *CFAR*;
- *Median Filter*.

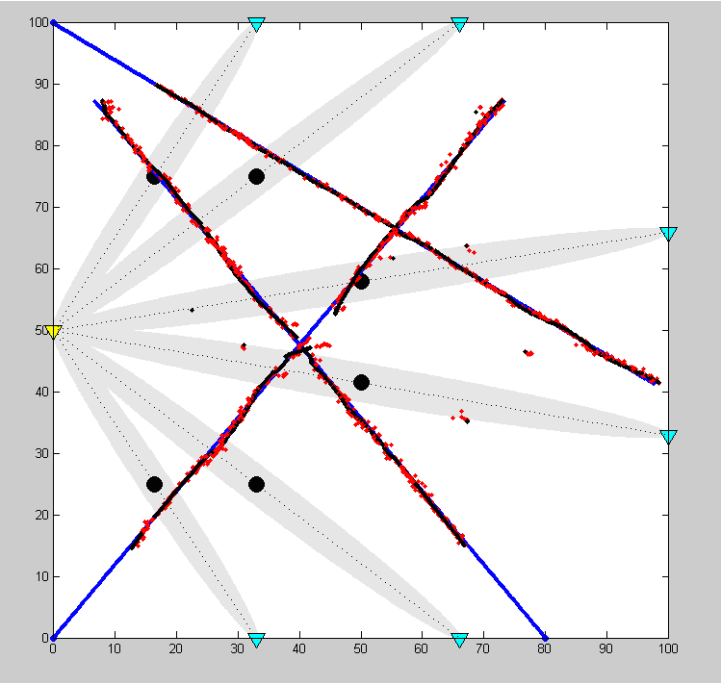


Figura 5.2: Scenario della simulazione base nel caso di 3 target, 3 ricevitori, un trasmettitore

In particolare in rosso viene tracciata la stima del percorso dei *target* senza *median filter* e con CFAR fisso ad una soglia pari a 3, mentre in nero viene lanciato il codice con l'utilizzo del *median filter*.

5.2 Simulazioni con gli Algoritmi implementati

Successivamente al test della precedente versione del simulatore, viene implementato sul codice sorgente l'algoritmo relativo al *Clustering* mono e bidimensionale (esaminati nel Capitolo 4). In particolare si può notare in Figura 3.2 lo schematico del funzionamento del simulatore ed in particolare dove vengono implementati gli algoritmi di *Clustering* che sono stati realizzati per questa tesi. Questi algoritmi sono stati aggiunti al fine di migliorare la clusterizzazione e renderla più efficiente. Le simulazioni eseguite sono riferite al caso con uno o tre target nell'area standard dei $[100 \times 100]m$, con tre ricevitore ed un trasmettitore. Il fattore che viene variato nelle simulazioni è la soglia del CFAR che determina il limite entro il quale il simulatore può distinguere la presenza del target dal semplice *clutter* o dall'assenza stessa dell'eventuale

intruso.
In particolare per avere un riscontro delle simulazioni, vengono mostrate figure nella quali:

- In blu viene disegnata la traccia reale che il target percorre (spesso poco visibile poichè è sovrastata dalle successive due tracce relative alle misurazioni);

- Le tracce in verde rappresentano le misure in ingresso al blocco di *tracking* (ossia successivamente al confronto con il CFAR ed agli algoritmi di *clustering*);

- In nero ci sono le stesse misure però ulteriormente filtrate da un filtro PHD (fitro particellare molto performante nel caso in cui vi sia uno scenario composto da più *target*)[3].

Nel caso di simulazioni con singolo target sono state eseguite simulazioni aumentando il valore del CFAR così come è riportato in Tabella 5.2.

Area [metri x metri]	**Numero di scansioni**	**Punti riflettenti**	**Soglia del CFAR**
100 x 100	600	100	7
100 x 100	600	100	8
100 x 100	600	100	9
100 x 100	600	100	10
100 x 100	600	100	11
100 x 100	600	100	12

Tabella 5.2: Tabella relativa a tutte le simulazioni eseguite in questo elaborato nel caso della presenza di un solo *target* nell'area sorvegliata

.

Anche nel caso di simulazioni con tre target nell'area di sorveglianza, sono state eseguite un certo numero di simulazioni variando il valore di soglia CFAR (tutte le simulazioni eseguite sono riportate in 5.3).

Area [metri x metri]	Numero di scansioni	Punti riflettenti	Soglia del CFAR
100 x 100	600	100	11
100 x 100	600	100	12
100 x 100	600	100	15
100 x 100	600	100	17
100 x 100	600	100	20

Tabella 5.3: Tabella relativa a tutte le simulazioni eseguite in questo elaborato nel caso della presenza di tre *target* nell'area sorvegliata

.

Di seguito vengono riportate i risultati delle simulazioni più rilevanti sia per il caso singolo, che per quello con tre *target*.

5.2.1 Simulazione singolo target CFAR=8

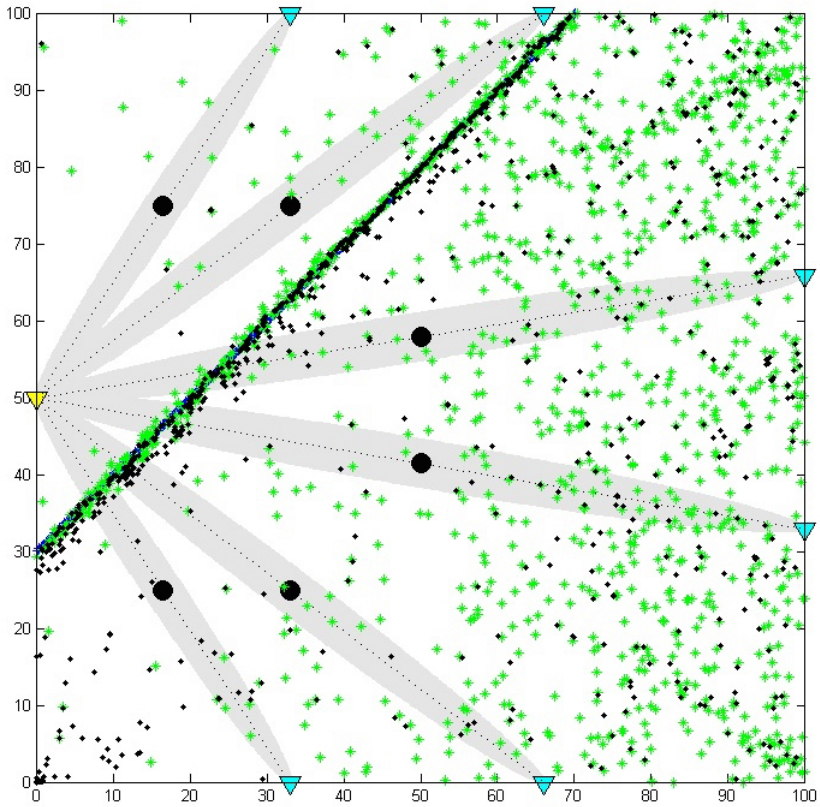


Figura 5.3: Simulazione con un target e soglia del CFAR impostata a 8

.

Dalla figura 5.3 si nota come, per una soglia di CFAR bassa (è impostata ad 8 in questo caso), la presenza di falsi allarmi è molto elevata e quindi si necessita un aumento di questo valore. Questo problema è sicuramente dovuto all'eliminazione del *median filter* rispetto alle simulazioni di partenza (come quella vista in Figura 5.2).

5.2.2 Simulazione singolo target CFAR=9

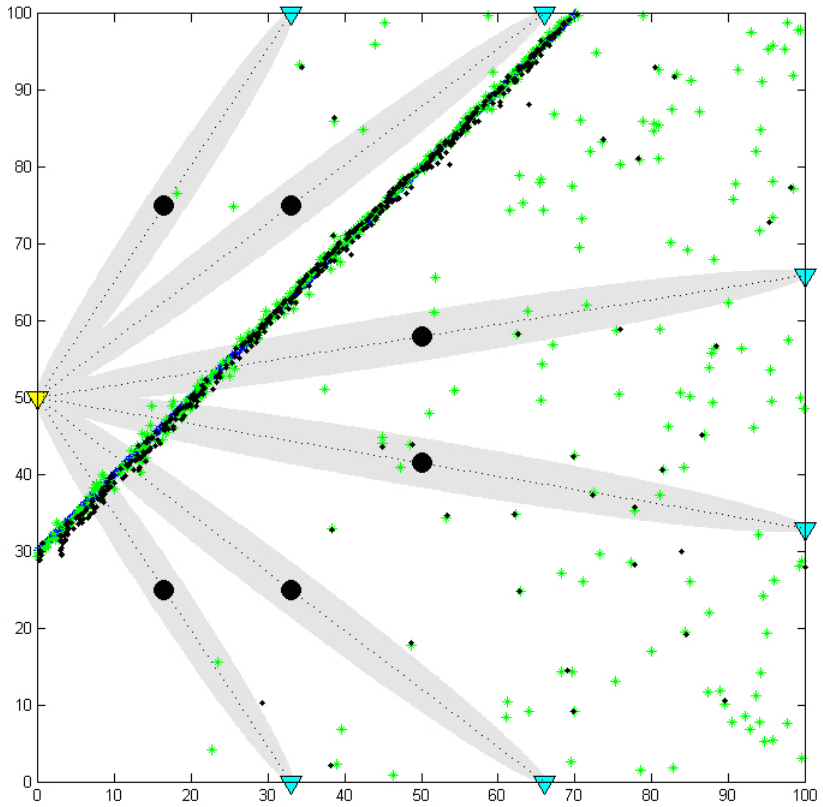


Figura 5.4: Simulazione con un target e soglia del CFAR impostata a 9

.

In questa simulazione (Figura 5.4) è lampante come già solo aumentando di un'unità il valore del CFAR il numero di falsi allarmi diminuisce, sia in prossimità, che distanti dalla traiettoria del target.

5.2.3 Simulazione singolo target CFAR=11

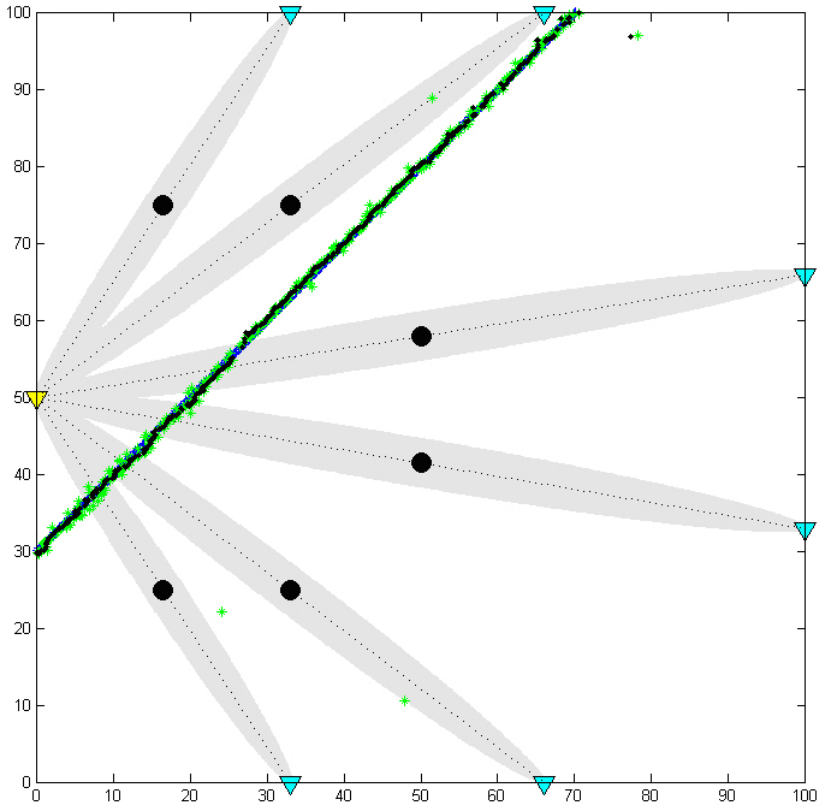


Figura 5.5: Simulazione con un target e soglia del CFAR impostata a 11

.

Diversamente dalle altre simulazioni, per quella in Figura 5.5 si nota che il numero di falsi allarmi è tendente allo zero, confermando la loro diminuzione con l'aumento del CFAR che in questo caso vale 11. Bisogna però considerare che non si può aumentare troppo il CFAR poiché la possibilità che alcuni punti relativi al target siano sotto soglia e quindi non rilevati dagli algoritmi implementati.

5.2.4 Simulazione tre target CFAR=11

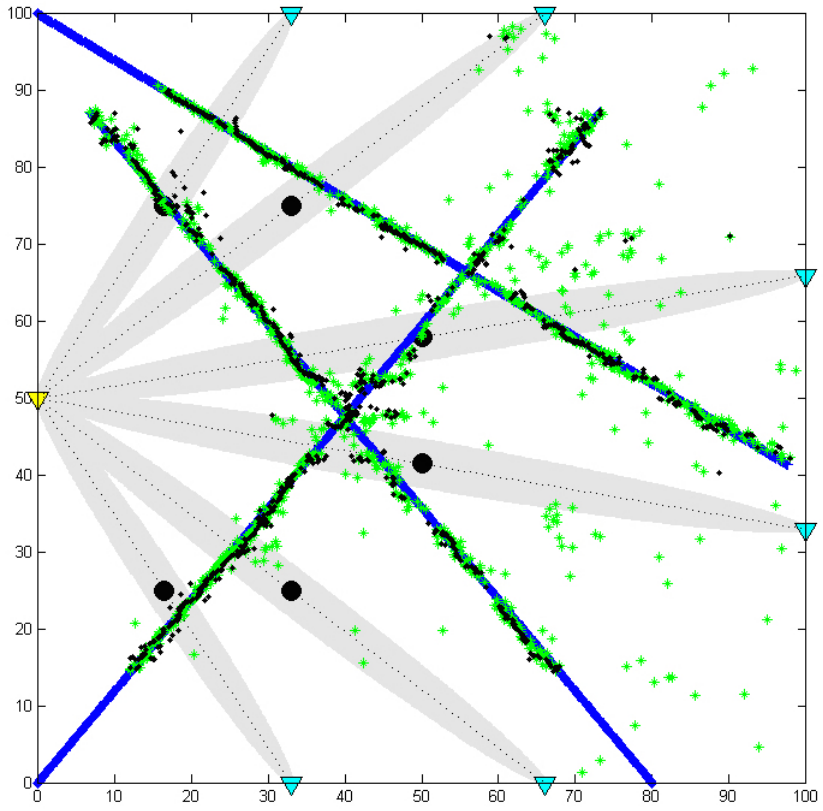


Figura 5.6: Simulazione con tre target e soglia del CFAR impostata a 11

.

Nel caso di tre target che si muovono in contemporanea nell'area di sorveglianza (Figura 5.6), si nota che, rispetto a caso di singolo target, il simulatore purtroppo ha ancora molti falsi allarmi, ecco perchè sono state effettuate ulteriori simulazioni al fine di diminuirne il numero.

5.2.5 Simulazione tre target CFAR=17

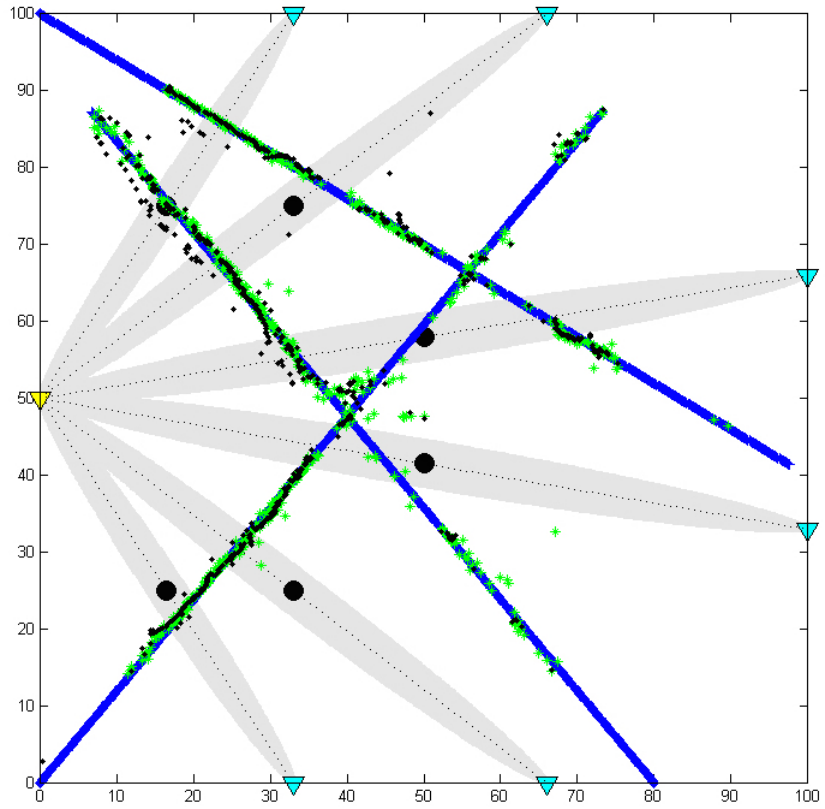


Figura 5.7: Simulazione con tre target e soglia del CFAR impostata a 17

.

Come previsto, aumentando il CFAR, il numero di falsi allarmi cala drasticamente e le traiettoie sono comunque abbastanza nitide, quindi la soglia non è troppo elevata da eliminare la traccia stessa del target.

Capitolo 6

Time Domain PulsOn 410

Il PulsOn 410 della Time Domain Corporation è un dispositivo radar che utilizza forme d'onda UWB propagate attraverso il canale a Radio Frequenza. Esso può essere impostato con varie configurazioni, a seconda dell'applicazione che si vuole implementare; infatti può essere utilizzato come Radar Monostatico, Bistatico o Multistatico. In questo capitolo vengono analizzate le caratteristiche principali di questi dispositivi.

6.1 Caratteristiche principali

Il P410 è un dispositivo molto piccolo ($76 \times 80 \times 19mm$), leggero (58 grammi), che permette di implementare applicazioni radar a corto raggio, a basso costo e a basso consumo di potenza. La sostanziale differenza, rispetto ai vecchi nodi P220 della Time Domain, sta in primis nella notevole riduzione delle dimensioni e del peso, ma soprattuto nell'utilizzo della sola uscita USB per lo scambio di dati, seppur mantenendo i protocolli di trasmissione a pacchetto delle informazioni. L'utilizzo e l'analisi delle forme d'onda generate dai vari nodi P410 invece possono essere sfruttati mediante tre nuovi *software* messi a disposizione dalla Time Domain stessa:

- **Monostatic Radar Module (MRM)** : è il *software* utilizzato per sfruttare i P410 come radar monostatici, quindi tramite l'utilizzo di due antenne Broadspec nello stesso dispositivo (una trasmittente e l'altra ricevente, come si può riscontrare in Figura 6.1), vi è la possibilità di fare una stima molto accurata della distanza fra un eventuale *target* in movimento e le antenne [21].

Figura 6.1: PulsOn 410 Time Domain Corporation

.

- **Ranging and Comunication Module (RCM)** : è il *software* che permette di fare misure di distanza *peer-to-peer* con un accuratezza molto buona (anche fino a 2 cm di errore massimo in ambienti *indoor* in linea di vista, o addirittura può anche fare misure di distanza attraverso i muri con un errore minimo dell'ordine di 0.5 metri) [20] [22].

- **Channel Analysis Tool (CAT)** : è il *software* che permette di analizzare, misurare e salvare le forme d'onda UWB lanciate da un P410 trasmittente nell'ambiente. Il tutto implementato in un ambiente con radar multi o bistatico [23].

Dalla Figura 6.2 si può riscontrare nello specifico come il P410 si interfaccia all'esterno; sono presenti: due porte per l'innesto di una o due antenne (a seconda dell'applicazione che si vuole utilizzare), un'interfaccia per l'alimentazione, quattro LED (due relativi all'accensione e allo stato di funzionamento posizionati di fianco all'alimentazione e altri due relativi alla trasmissione/ricezione di pacchetti, all'interno della scheda stessa), una presa mini-usb per il trasporto delle informazioni

verso l'*host* e per la configurazione del nodo.

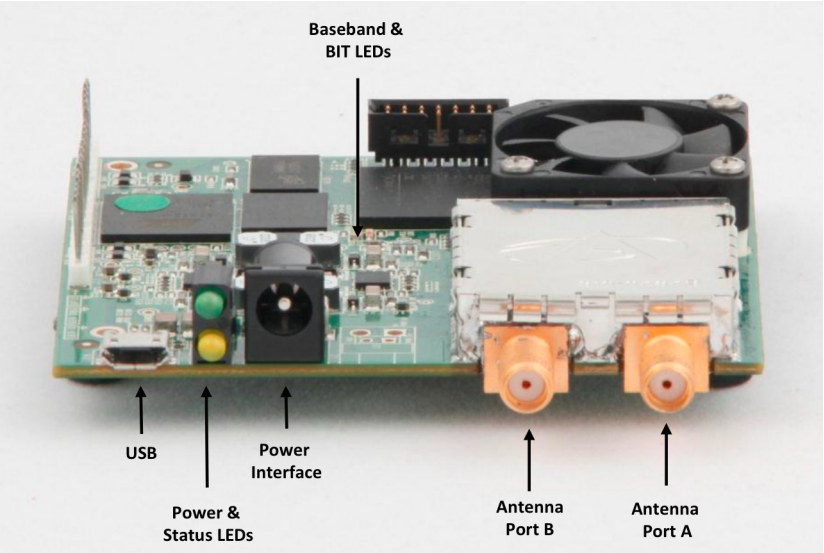


Figura 6.2: Dettaglio del PulsOn 410 Time Domain Corporation [24]

.

Le specifiche di trasmissione principali per questi nodi sono specificate nel *Data Sheet* forniti[25] e riassunti in Tabella 6.1.

Specifiche P410	**Valore**
Maximum Power Consumption	3.9 Watts (as Radar)
Operating Band RF	[3.1 ÷ 5.3] GHz
Center Frequency	4.3 GHz
Transmit power	[-31.6 ÷ -12.64] dBm
Noise Figure	4.8 dB
Transmit Pulse Repetition Rate	10.1 MHz

Tabella 6.1: Tabella delle principali specifiche del P410

.

6.2 Parametri per l'impostazione dei nodi

Impostare i parametri corretti del P410 è fondamentale per avere una corretta trasmissione/ricezione. In questo paragrafo vengono considerate le principali impostazioni per il caso *Channel Analysis Tool (CAT)*

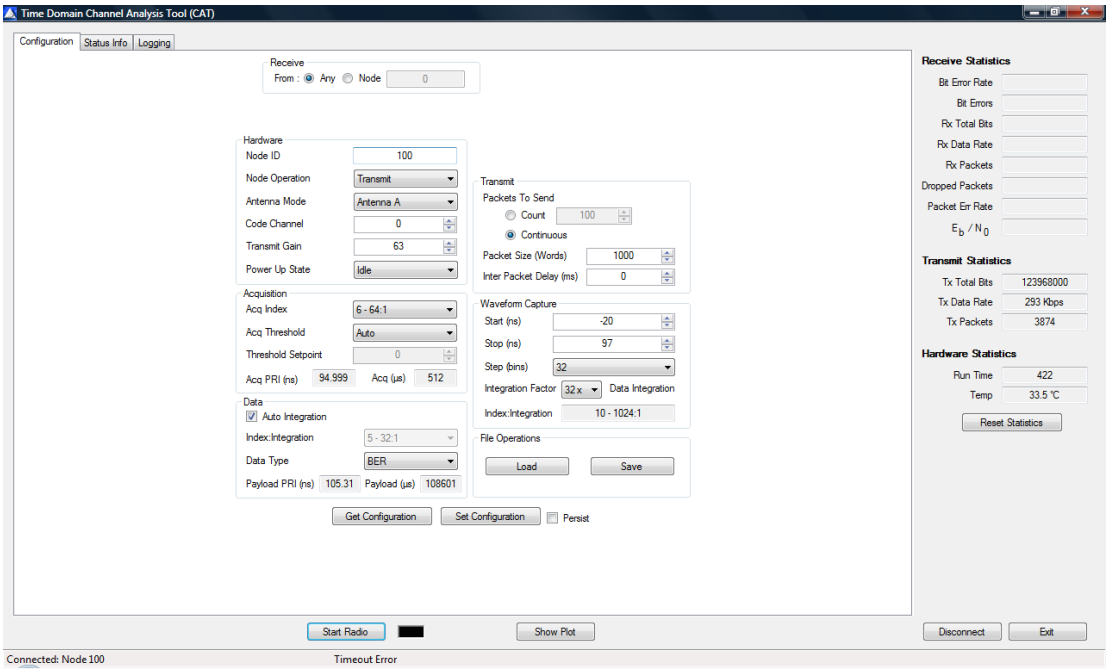


Figura 6.3: GUI nel caso di un nodo trasmittente nel caso CAT per radar multistatico

.

così come si possono vedere in Figura 6.3 (che è il caso analizzato per le misurazioni sperimentali):

- ***Node ID*** : codice identificativo del nodo (di solito fornito di default dal costruttore).

- ***Node Operation*** : indica in che stato sta operando quel nodo specifico; tipicamente o ricevitore, o trasmettitore.

- ***Code Channel*** : canale nel quale si vuole trasmettere; tipicamente è consigliato utilizzare i canali dallo 0 al 6 compresi, ovviamente per far si che la trasmissione vada a buon fine, trasmettitore e ricevitore devono essere impostati sullo stesso canale.

- ***Transmit Gain*** : è il guadagno di trasmissione il quale può andare da un valore di 0 ad un massimo di 63 nel range visto in Tabella 6.1 (quindi 0 sarà riferito a $-31.6dBm$, mentre 63 ai $-12.64dBm$).

- ***Acq Index*** : l'*Acquisition Pulse Integration Index* determina il *range* al quale la trasmissione può essere acquisita, è quindi parte fondamentale del fattore di integrazione utilizzato.
- ***Auto Integration*** : se attivo, imposta il valore d'integrazione dati ad un valore più basso rispetto all' *Acq Index*.
- ***Start (ns)*** : è il punto di partenza dal quale avviene la cattura della forma d'onda ricevuta in nanosecondi.
- ***Stop (ns)*** : è il punto finale nel quale avviene la cattura della forma d'onda ricevuta in nanosecondi.
- ***Step (bins)*** : è il numero di *bins* (1 *bin* aprossimativamente è pari a $1.9073ps$) e quindi di secondi che intercorrono tra una misurazione e l'altra di una scansione.
- ***Integration Factor*** : valore che definisce il *Pulse Integration Index* da utilizzare nel ricevimento delle forme d'onda (ovviamente andrà ad aggiungersi al fattore già impostato in trasmissione con l'*Acq Index*).
- ***Index:Integration*** : questo valore mostra il fattore di integrazione globale che l'utente ha scelto tramite *Acq Index* e *Integration Factor*.

Capitolo 7

Misure Sperimentali

In questo capitolo vengono affrontate tutte le problematiche riguardanti la messa in pratica di uno scenario reale di un'area sorvegliata, composto da quattro antenne (una trasmittente e tre riceventi), le quali devono rilevare un intruso che si muove per una traiettoria interna al perimetro. L'obiettivo finale è di confrontare le misure in presenza del *target*, con quelle in assenza (*Empty room*) al fine di rilevare, nelle forme d'onda acquisite dai nodi, un riscontro dell'intruso nell'area di osservaizione.
Vengono di seguito descritti: lo scenario utilizzato, il *stetup* dei nodi PulsOn 410 (già descritti nel Capitolo 6), i target utilizzati e la relativa traccia ed infine le modalità di salvataggio ed elaborazione dati delle forme d'onda acquisite dai ricevitori.

7.1 Scenario Utilizzato

I quattro nodi P410 sono stati installati all'interno del laboratorio Droni della Scuola di Ingegneria e Architettura (Campus di Cesena), in un'area di forma quadrata con lati di 10 metri, con un'impalcatura che ha permesso di posizionare le antenne a metà di ogni lato (come si può notare dalla Figura 7.1 che rappresenta lo scenario utilizzato). Tutti i nodi sono stati posizionati ad un'altezza di circa $2.04m$ da terra, sporgenti $10cm$ dall'impalcatura e fissati ad essa tramite una canalina a sua volta basata su un supporto di legno (vedi Figura 7.2). Tutto ciò è stato fatto per cercare di ridurre l'effetto di riflessioni dal pavimento e di un'eventuale riflessione dall'impalcatura metallica. Questo perché, grazie a preliminari misurazioni, è stato riscontrato come le antenne poste a terra o a circa $0.30cm$ su delle sedie, generino un segnale rice-

Figura 7.1: Immagine dello scenario reale in cui sono state fatte le misure di questo elaborato.

vuto affetto gravemente dal cammino multiplo.

Figura 7.2: Immagine della struttura d'appoggio e del posizionamento dei P410 sull'impalcatura nell'area di misurazione.

7.2 *Target* e traiettoria

Per le misurazioni sono stati valutati vari tipi di *target* (ad esempio è stato utilizzato un attaccapanni con una giacca, oppure un vecchio armadio per *router*, i quali però non hanno riscontrato risultati significativi nelle misure) giungendo all'utilizzo di un oggetto a forma sferica con diametro di circa $40cm$, ricoperto di stagnola, posto all'altezza di $1.73m$. Si è giunti a questa soluzione (visibile in Figura 7.3) poichè fra gli oggetti utilizzati è quello con *Radar Cross Section* maggiore (ma comunque inferiore a quella umana) e quindi quello maggiormente rilevabile dai radar.

La traiettoria percorsa dal target utilizzato per le misure (Figura 7.3)

Figura 7.3: Immagine del *target* utilizzato per le misurazioni nell'area.

può essere vista nella simulazione fatta con Matlab e riportata in Figura 7.4. Si nota che la traiettoria è rettilinea, lunga $9.05m$ e attraversa l'area d'osservazione delle antenne.

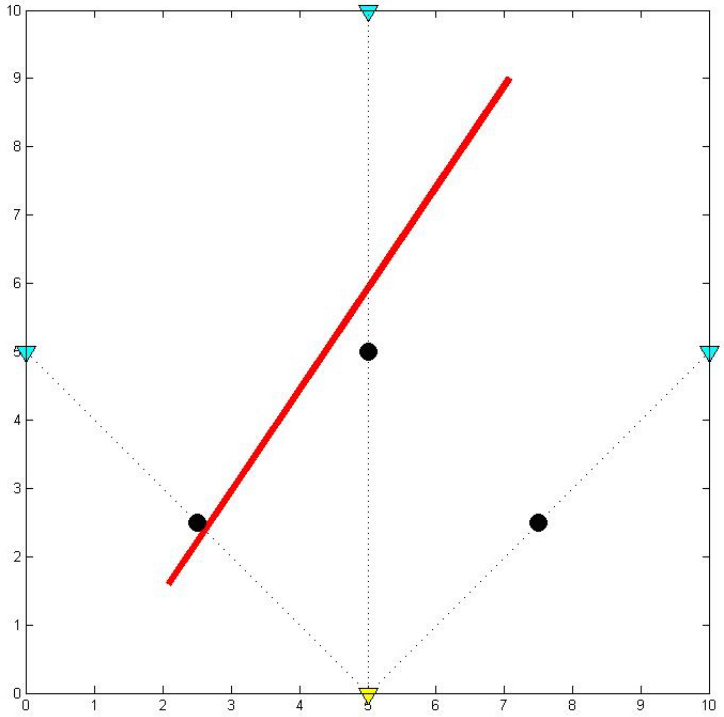


Figura 7.4: Simulazione che rappresenta lo scenario delle misure composto dal trasmettitore (triangolo giallo), i tre ricevitori (triangoli azzurri) e la traccia percorsa dal *target*.

7.3 Misure

Le misure sperimentali derivano innanzitutto dal lavoro di tracciamento della traiettoria stabilita per *target* migliore (ossia quella vista in Figura 7.4). Si è scelto di acquisire i dati relativi alle tre antenne riceventi, spostando il target ogni 5 cm, così da percorrere tutti i 9.05 m in 182 $misure$. Si hanno quindi, per ognuna delle 182 posizioni del *target*, tre acquisizioni distinte in relazione ai tre i ricevitori. Le misure sono state eseguite spostando per ogni posizione il *target*, mantenendo la posizione del target perpendicolare rispetto alla linea della traccia percorsa (vedi Figura 7.5).

Figura 7.5: Dettaglio della posizione del target: in questo caso si nota come il target sia a 8.05 m dal punto di partenza quindi in posizione 162.

7.4 Configurazione dei nodi tramite GUI

In questo paragrafo viene descritto come impostare ogni singolo nodo P410 (Vedi anche appendice A) in trasmissione o ricezione e prepararlo all'acquisizione delle forme d'onda UWB, tramite i parametri forniti grazie all'interfaccia grafica del *Channel Analysis Tool (CAT)*.
Innanzitutto, aprendo il *Software* CAT, bisogna connettere all'*host* ogni singolo nodo utilizzato: nel caso di studio analizzato in questa tesi, si considera lo scenario composto da un trasmettitore e tre ricevitori, di conseguenza, prima di ogni elaborazione, vengono connessi i 4 nodi al PC con 4 GUI aperte tramite CAT (sono stati quindi utilizzati: il nodo 100 come trasmettitore, il nodo 101 come ricevitore 1, il nodo 102 come ricevitore 2 e il nodo 103 come ricevitore 3).
Le impostazioni per trasmettitore e ricevitori sono sostanzialmente le stesse, a parte il parametro *Node Operation* il quale identifica appunto se il nodo deve trasmettere (*Transmit*) o ricevere (*Receive*). Sono tutte riportate di seguito nella Tabella 7.1 e possono essere viste in configurazione di trasmissione, in Figura 6.3.

Impostazione P410	**Valore**
Code Channel	0
Transmit Gain	63
Acq Index	6 - 64 : 1
Acq Threshold	Auto
Waveform Capture	
Start (ns)	- 20
Stop (ns)	97
Step	32
Ingegration Factor	32 x

Tabella 7.1: Tabella delle impostazioni utilizzate sui nodi P410 per le misure sperimentali.

Si nota quindi che con i parametri impostati in Tabella 7.1, il fattore di integrazione finale è di 1024, di conseguenza vengono collezionate e mediate 1024 forme d'onda in una per ogni singolo salvataggio.

7.5 *Start Radio* e salvataggio forme d'onda

Passaggio importante per fare partire il segnale UWB, è l'avvio della trasmissione radio tramite il tasto *Start Radio*[24] della GUI del trasmettitore e dei ricevitori (vedi Figura 6.3). Lanciando così l'invio delle forme d'onda fra TX e RX, è possibile visualizzare le forme d'onda ricevute direttamente da ogni ricevitore tramite la funzione *Show Plot* (vedi Figura 6.3), ottenendo il tipico segnale generato dalle P410 visibile in Figura 7.6.

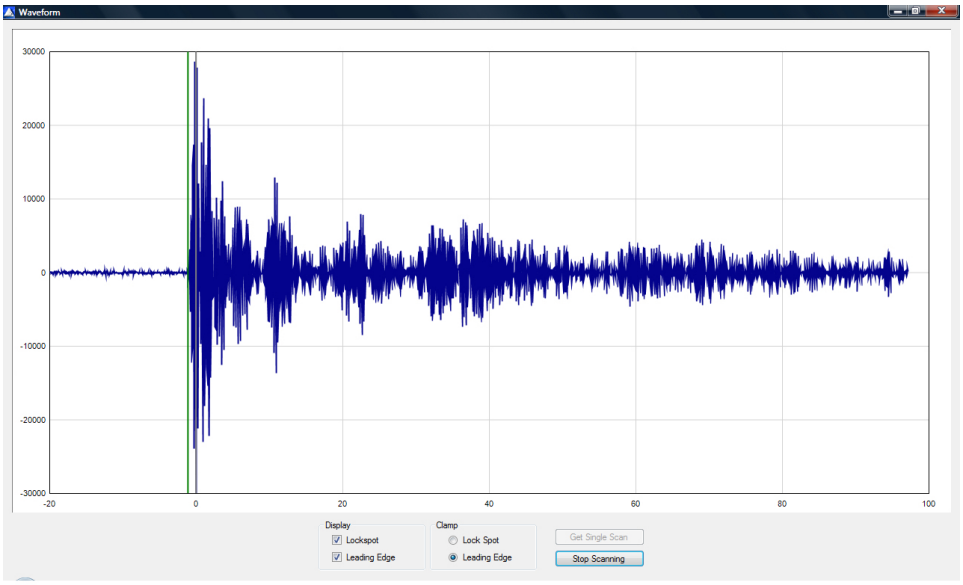


Figura 7.6: Forma d'onda ricevuta per uno dei ricevitori nel caso di scansione continua (*Start Scanning*) e in presenza del target.

Nella forma d'onda (Figura 7.6) sono quindi presenti: il segnale UWB che viene dal nodo trasmettitore, eventuali riflessioni dovute al clutter e infine, in mezzo a tutto ciò, un piccolo *spike* che identifica la presenza del *target* (come verrà mostrato nella successiva analisi dell'elelaborato di tesi).
Per salvare tutte le forme d'onda dei segnali ricevuti, dalla GUI di ogni ricevitore, si seleziona il menù a tendina *Logging* (vedi Figure 6.3 e 7.7) e si fa partire lo *Start Logging*. Questa funzione permette di salvare, nel percorso selezionato direttamente dall'utente (vedi Figura 7.7)[24], tutti i dati relativi ai campioni ricevuti dal ricevitore in modo poi da poterli riprodurre in Matlab. I file salvati (un esempio è in Figura 7.8) sono di tipo *csv* ossia in forma di tabella e contengono un'intestazione iniziale che è relativa alle impostazioni della forma d'onda (ossia relative alle specifiche utilizzate nel capitolo 7.4) e successivamente tutti i dati relativi alla forma d'onda vista in Figura 7.6 al fine di poterla riprodurre.

Vengono quindi salvati i file relativi ai tre ricevitori in ogni posizione del target, per cui vengono ripetuti i salvataggi per 182 volte, più un'ulteriore volta nel caso *Empty room* (ossia nel caso di assenza di eventuali *target* nell'area di sorveglianza dei nodi).
Raccolte tutte le misurazioni per ogni caso, si può procedere con l'a-

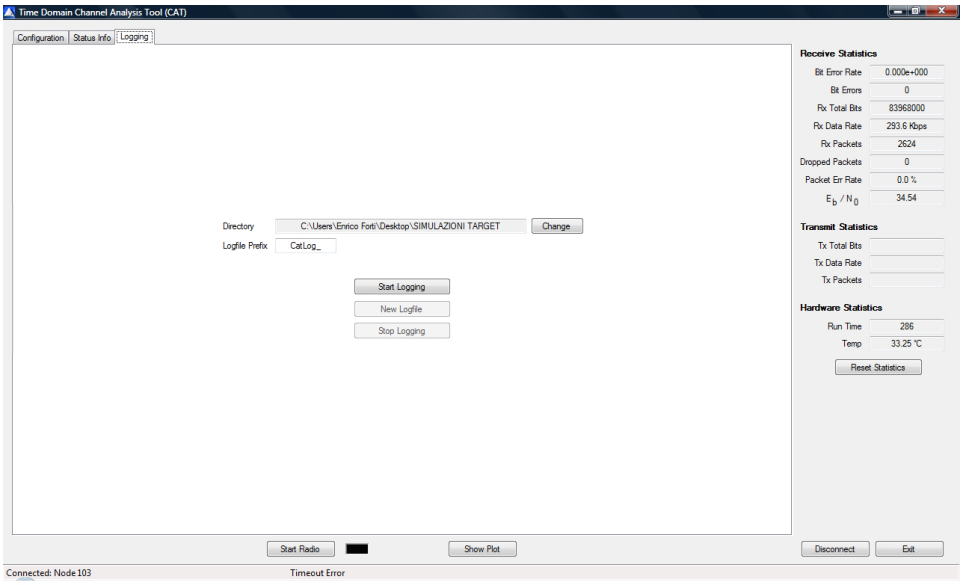
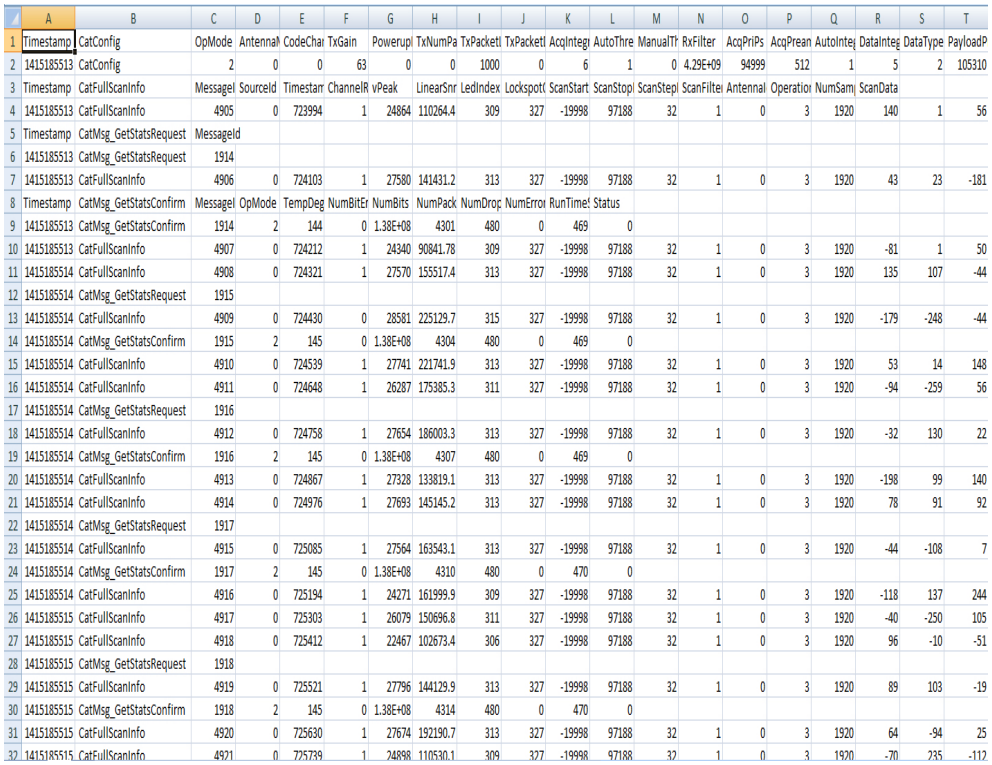


Figura 7.7: GUI nel caso in cui si debbano salvare le forme d'onda ricevute.

nalisi e l'elaborazione dei dati.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Timestamp	CatConfig	OpMode	Antennal	CodeChar	TxGain	Powerup	TxNumPa	TxPacketI	TxPacketI	AcqIntegr	AutoThre	ManualTh	RxFilter	AcqPriPs	AcqPrean	AutoInte	DataInteg	DataType	PayloadP
2	1415185513	CatConfig	2	0	0	63	0	0	1000	0	6	1	0	4.29E+09	94999	512	1	5	2	105310
3	Timestamp	CatFullScanInfo	MessageI	SourceId	Timestam	ChannelR	vPeak	LinearSnr	LedIndex	Lockspot(	ScanStart	ScanStop	ScanStepI	ScanFilte	Antennal	Operatio	NumSam	ScanData		
4	1415185513	CatFullScanInfo	4905	0	723994	1	24864	110264.4	309	327	-19998	97188	32	1	0	3	1920	140	1	56
5	Timestamp	CatMsg_GetStatsRequest	MessageId																	
6	1415185513	CatMsg_GetStatsRequest	1914																	
7	1415185513	CatFullScanInfo	4906	0	724103	1	27580	141431.2	313	327	-19998	97188	32	1	0	3	1920	43	23	-181
8	Timestamp	CatMsg_GetStatsConfirm	MessageI	OpMode	TempDeg	NumBitEr	NumBits	NumPack	NumDrop	NumErro	RunTimeS	Status								
9	1415185513	CatMsg_GetStatsConfirm	1914	2	144	0	1.38E+08	4301	480	0	469	0								
10	1415185513	CatFullScanInfo	4907	0	724212	1	24340	90841.78	309	327	-19998	97188	32	1	0	3	1920	-81	1	50
11	1415185514	CatFullScanInfo	4908	0	724321	1	27570	155517.4	313	327	-19998	97188	32	1	0	3	1920	135	107	-44
12	1415185514	CatMsg_GetStatsRequest	1915																	
13	1415185514	CatFullScanInfo	4909	0	724430	0	28581	225129.7	315	327	-19998	97188	32	1	0	3	1920	-179	-248	-44
14	1415185514	CatMsg_GetStatsConfirm	1915	2	145	0	1.38E+08	4304	480	0	469	0								
15	1415185514	CatFullScanInfo	4910	0	724539	1	27741	221741.9	313	327	-19998	97188	32	1	0	3	1920	53	14	148
16	1415185514	CatFullScanInfo	4911	0	724648	1	26287	175385.3	311	327	-19998	97188	32	1	0	3	1920	-94	-259	56
17	1415185514	CatMsg_GetStatsRequest	1916																	
18	1415185514	CatFullScanInfo	4912	0	724758	1	27654	186003.3	313	327	-19998	97188	32	1	0	3	1920	-32	130	22
19	1415185514	CatMsg_GetStatsConfirm	1916	2	145	0	1.38E+08	4307	480	0	469	0								
20	1415185514	CatFullScanInfo	4913	0	724867	1	27328	133819.1	313	327	-19998	97188	32	1	0	3	1920	-198	99	140
21	1415185514	CatFullScanInfo	4914	0	724976	1	27693	145145.2	313	327	-19998	97188	32	1	0	3	1920	78	91	92
22	1415185514	CatMsg_GetStatsRequest	1917																	
23	1415185514	CatFullScanInfo	4915	0	725085	1	27564	163543.1	313	327	-19998	97188	32	1	0	3	1920	-44	-108	7
24	1415185514	CatMsg_GetStatsConfirm	1917	2	145	0	1.38E+08	4310	480	0	470	0								
25	1415185514	CatFullScanInfo	4916	0	725194	1	24271	161999.9	309	327	-19998	97188	32	1	0	3	1920	-118	137	244
26	1415185515	CatFullScanInfo	4917	0	725303	1	26079	150696.8	311	327	-19998	97188	32	1	0	3	1920	-40	-250	105
27	1415185515	CatFullScanInfo	4918	0	725412	1	22467	102673.4	306	327	-19998	97188	32	1	0	3	1920	96	-10	-51
28	1415185515	CatMsg_GetStatsRequest	1918																	
29	1415185515	CatFullScanInfo	4919	0	725521	1	27796	144129.9	313	327	-19998	97188	32	1	0	3	1920	89	103	-19
30	1415185515	CatMsg_GetStatsConfirm	1918	2	145	0	1.38E+08	4314	480	0	470	0								
31	1415185515	CatFullScanInfo	4920	0	725630	1	27674	192190.7	313	327	-19998	97188	32	1	0	3	1920	64	-94	25
32	1415185515	CatFullScanInfo	4921	0	725739	1	24898	110530.1	309	327	-19998	97188	32	1	0	3	1920	-70	235	-112

Figura 7.8: Esempio di LogFile contenente tutte le informazioni relative alla forma d'onda (tutti i dati fino alla colonna Q) e successivamente i campioni della forma d'onda stessa (*ScanData* che inizia alla colonna R).

Capitolo 8

Elaborazione dei dati

Il *Post-Processing* dei dati acquisiti con le misure sperimentali è fondamentale per la rilevazione del *target*, l'allineamento di tutte le forme d'onda, la rimozione del clutter. In questo capitolo vengono quindi riportate tutte le possibili sequenze di elaborazione analizzate e anche i risultati che ne conseguono. Tutto ciò è stato implementato utilizzando Matlab, grazie ad alcune tecniche descritte dettagliatamente nelle Appendici C, D, E.
L'acquisizione di tutte le misure nei file *.csv* (vedi Capitolo 7) permette la successiva elaborazione dei dati. Innanzitutto bisogna estrapolare le forme d'onda dalle informazioni iniziali dell'onda trasmessa e porre tutto in delle matrici (passaggi fondamentali per le successive elaborazioni con Matlab che vengono descritti nel dettaglio nell'Appendice B). Queste matrici saranno quindi composte da 1920 campioni relativi alle scansioni del *target* nelle 182 posizioni, e dell'*Empty Room*. L'*Empty Room* è una misura sperimentale effettuata dai nodi P410, nel caso in cui, nell'area di sorveglianza, non vi sia alcun *target*; le forme d'onda relative a questo caso, vengono quindi salvate per essere poi confrontate con quelle in presenza di intrusi, al fine della rimozione del *clutter* e della *detection*. Le forme d'onda in presenza del *target* e quelle dell'*Empty Room*, vengono elaborate dalle operazioni descritte nei prossimi paragrafi e possono essere viste di seguito.
Per semplicità le rappresentazioni di questo Capitolo sono riferite ai risultati relativi al solo "Ricevitore 1", che (come si può vedere in Figura 7.4) è quello alla destra del trasmettitore.

8.1 Allineamento, Rimozione *Clutter* e Inviluppo (*AT-RC-I*)

Nella prima elaborazione, si esegue in sequenza: l'allineamento delle forme d'onda salvate, poi la rimozione del *Clutter* ed infine l'inviluppo (demodulazione non coerente). Di seguito vengono riportate le figure più significative relative a questa sequenza di approccio dell'elaborazione dei dati.

8.1.1 *Post* Allineamento

Per capire bene come funzionano queste elaborazioni, è importante riportare le figure delle singole scansioni per vederne i miglioramenti. L'Allineamento è quindi fondamentale per sincronizzare nel tempo le forme d'onda nel caso di presenza e di assenza del *target* (vedi Appendice C).
In Figura 8.1 vengono riportate un esempio di forma d'onda acquisita in presenza del *target* (in blu), e in assenza di esso (*Empty Room*, in verde).

Dal solo allineamento (Figura 8.1) si può già notare che intorno ai

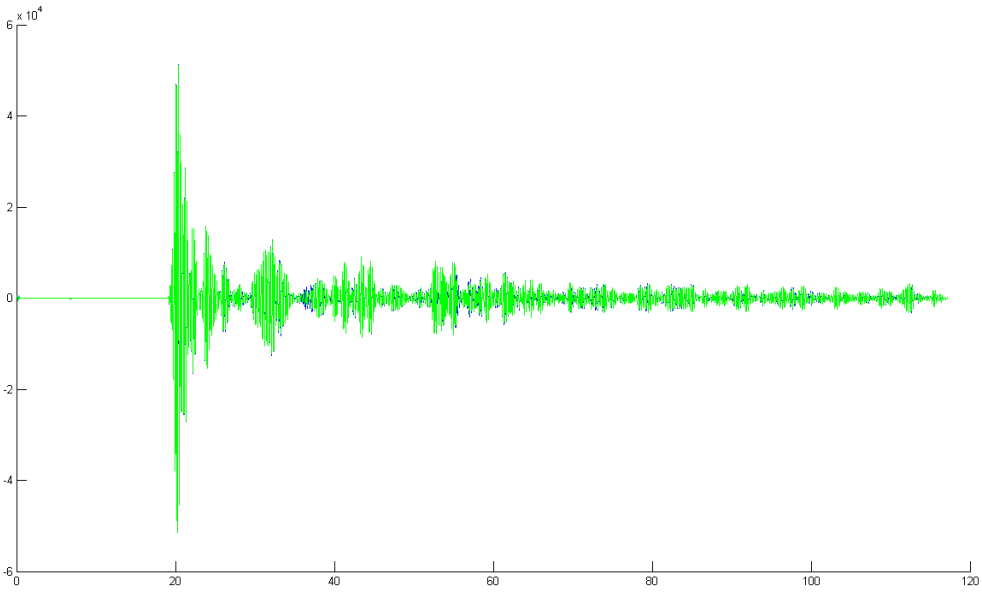


Figura 8.1: Rappresentazione dell'allineamento fra segnale con *target* (in blu) e quello con *Empty Room* (vedi Appendice F).

36 *ns* vi è una piccola differenza fra le due forme d'onda: questa è la posizione del target che, secondo i calcoli fatti e la scansione considerata, deve apparire in corrispondenza di quell'esatto ritardo temporale (vedi Appendice F).

8.1.2 *Post* Rimozione del *Clutter*

Dopo aver allineato le forme d'onde, il passo successivo è quello di sottrarre da quella del *target* (in blu in Figura 8.1), quella relativa all'*Empty Room* (in verde in Figura 8.1). La sottrazione netta campione per campione allineati, può essere vista in Figura 8.2.

Dalla Figura 8.2, si nota come è presente il picco relativo al target

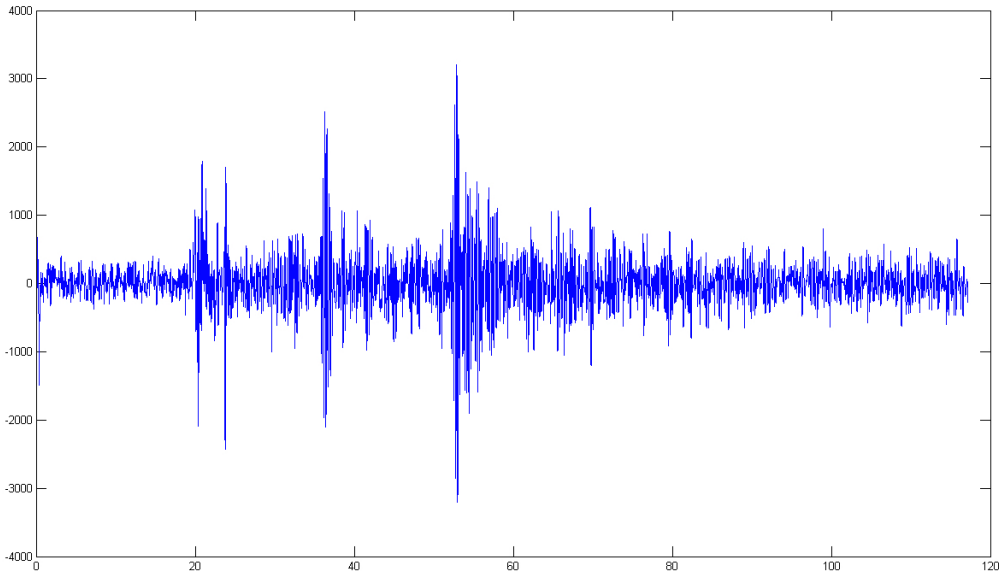


Figura 8.2: Rappresentazione della Rimozione del Clutter: differenza fra segnale con *target* ed *Empty Room*.

(intorno ai 36 *ns*), ma purtroppo anche altri picchi dovuti al *clutter* (all'incirca a 54 *ns*) e alla non perfetta rimozione del cammino diretto (intorno ai 20 *ns*).

8.1.3 *Post* Inviluppo

L'inviluppo è l'ultimo passaggio per quest'elaborazione che permette di avere una forma d'onda più pulita e che dovrebbe rendere l'immagine

e quindi il picco del *target* più visibile (vedi Figura 8.3).

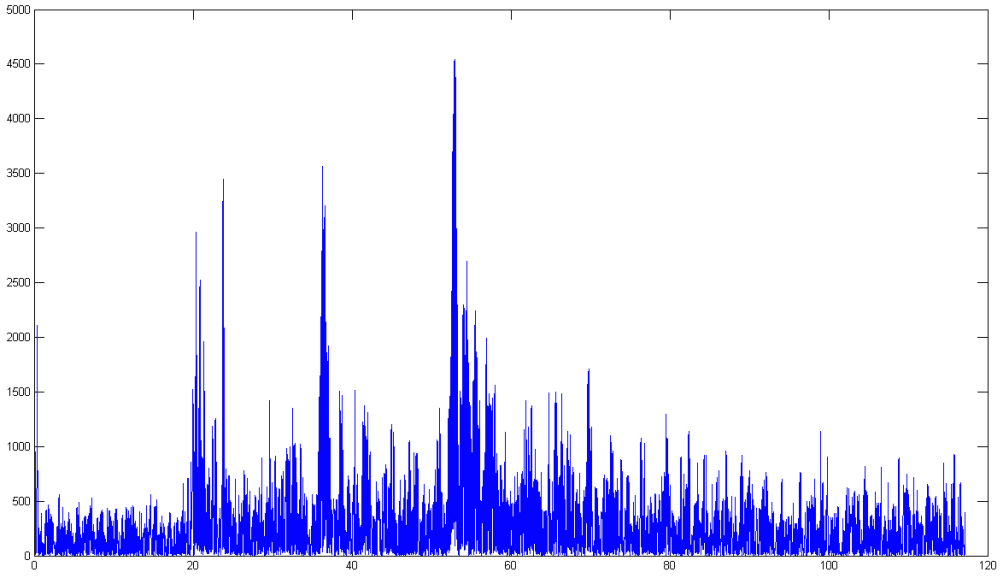


Figura 8.3: Rappresentazione della forma d'onda finale dopo: allineamento, rimozione clutter e inviluppo.

8.1.4 Rappresentazione finale *AT-RC-I*

Nell'ultimo passaggio dell'elaborazione si vuole rappresentare, in un'unica figura, tutte le 182 scansioni e quindi rilevazioni del *target*, nel tempo, riscontrando un suo effettivo movimento. Tramite la funzione *'imagesc'* di Matlab è possible visualizzare tutto ciò in Figura 8.4.

In Figura 8.4, si nota la presenza del *target* che, all'aumentare della sua posizione (dall'alto verso il basso), prima si avvicina al cammino diretto (intorno ai 20 ns) e successivamente si allontana. Questo è stato previsto e riscontrato grazie a questa figura, poichè effettivamente l'intruso utilizzato per le simulazioni (vedi figura 7.3), percorre una traiettoria che si avvicina e poi si allontana dalla congiungente dei due nodi (vedi Figura F.1).

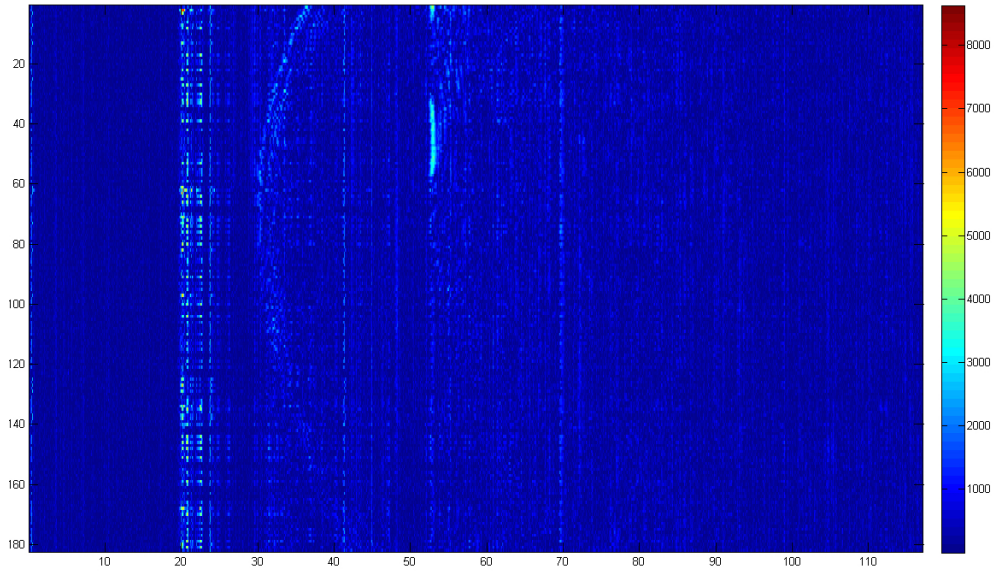


Figura 8.4: Rappresentazione del movimento del *target* nel tempo nel caso *AT-RC-I*.

8.2 Allineamento, Inviluppo e Rimozione *Clutter* (*AT-I-RC*)

In questo caso la prima parte dell'elaborazione è simile alla precedente; infatti si parte con l'allineamento delle forme d'onda, per poi passare all'inviluppo e solo infine alla rimozione del *Clutter*. Verranno quindi tralasciate le figure e le considerazioni in merito al "*Post* allineamento" poiché sono identiche al caso visto nel paragrafo 8.1.1.

8.2.1 *Post* Inviluppo

Dopo l'allineamento delle due forme d'onda relative al *target* e all'*Empty Room*, esse vengono elaborate separatamente ottenendone l'inviluppo per ciascuna di esse.

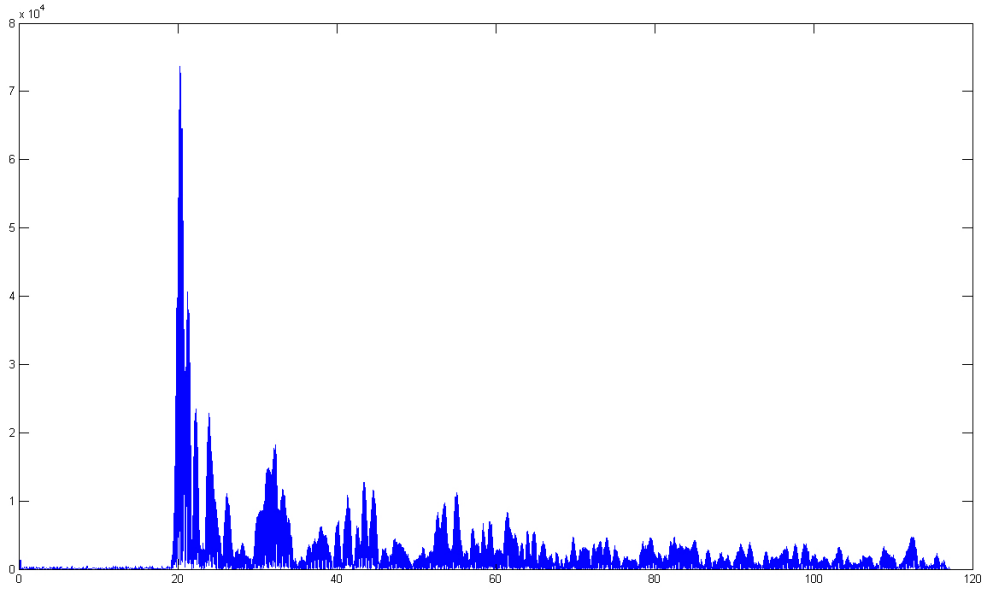


Figura 8.5: Rappresentazione della forma d'onda nel caso di presenza del *target*, avendo eseguito le elaborazioni di allineamento ed inviluppo.

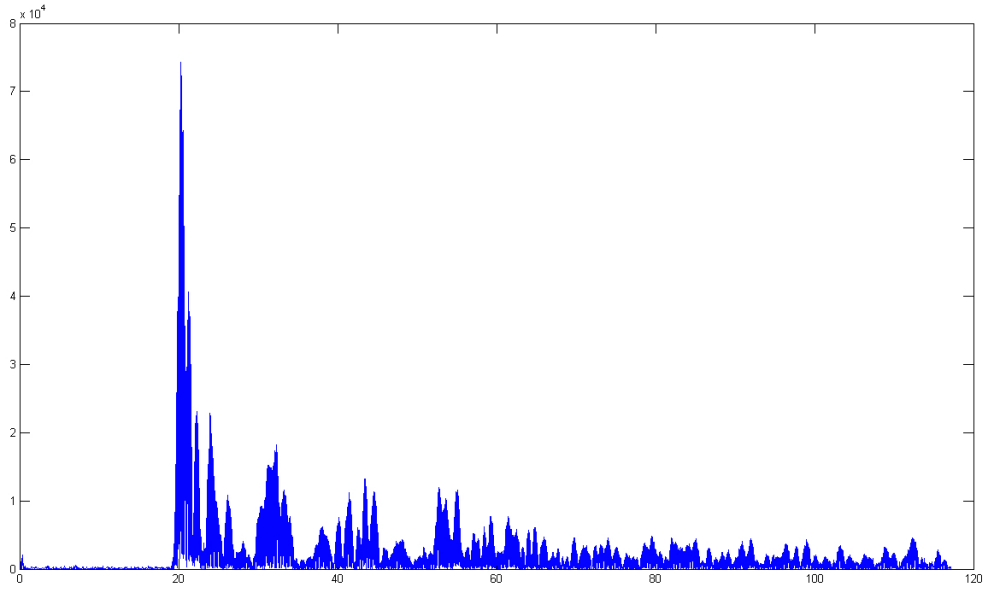


Figura 8.6: Rappresentazione della forma d'onda nel caso di assenza del *target* (*Empty Room*), avendo eseguito le elaborazioni di allineamento ed inviluppo.

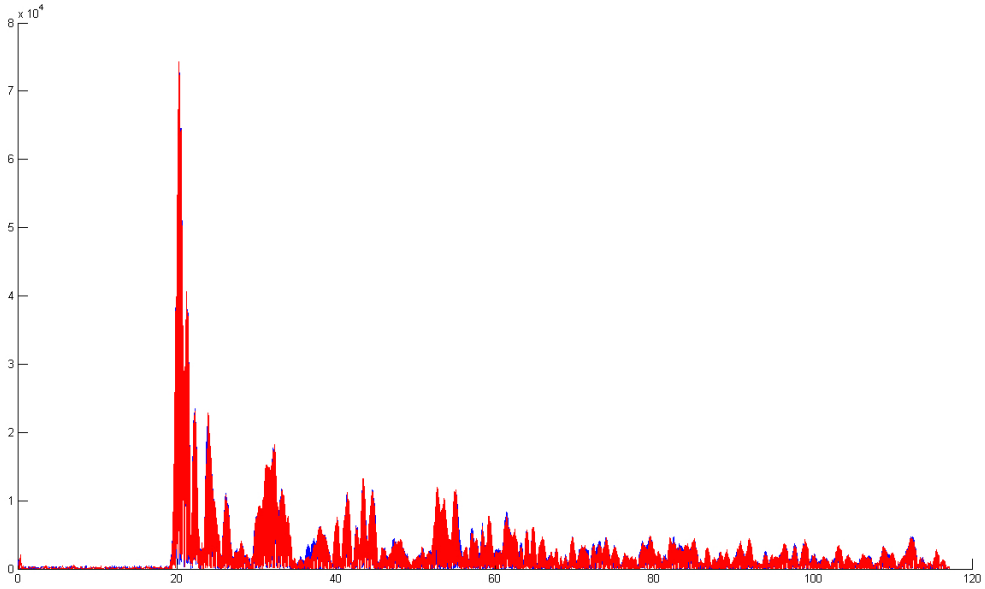


Figura 8.7: Rappresentazione delle due forme d'onda assieme (*target* in blu, *Empty Room* in rosso).

Anche in questo caso (come succede nel paragrafo 8.1.2) è riscontrabile una piccola discrepanza fra le due curve; infatti, in Figura 8.7 sempre in prossimità dei 36 *ns*, si nota un picco di colore blu (relativo alla forma d'onda dell'inviluppo del *target*) che spicca su quella dell'*Empty Room* (in rosso).

8.2.2 *Post* Rimozione del *Clutter*

L'ultima elaborazione della sequenza consiste nel sottrarre la curva di Figura 8.5, con quella di Figura 8.6, per far risaltare il picco relativo alla presenza del *target* di Figura 8.7.
Come nel caso precedente (Figura 8.4), anche in questo, oltre al picco relativo al *target*, vi sono altri *spike* relativi al cammino diretto e ai residui del *Clutter*.

8.2.3 Rappresentazione finale *AT-I-RC*

Come in precedenza, vengono visualizzati tutti i risultati finali per ogni scansione tramite la Figura 8.9.
Purtroppo in relazione ai risultati del caso *AT-RC-I*, non vi sono visto-

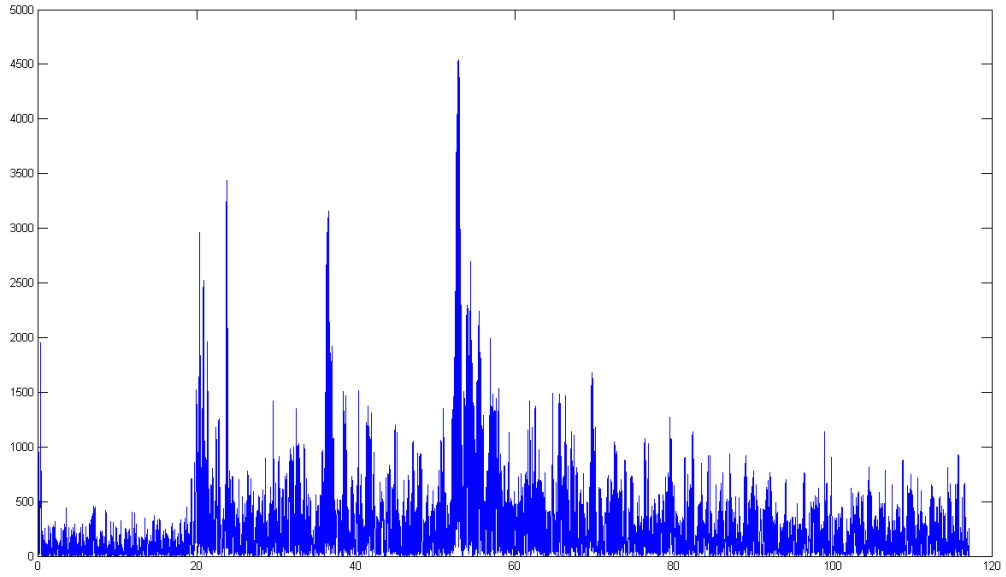


Figura 8.8: Rappresentazione finale dopo: allineamento, inviluppo e rimozione del *Clutter*.

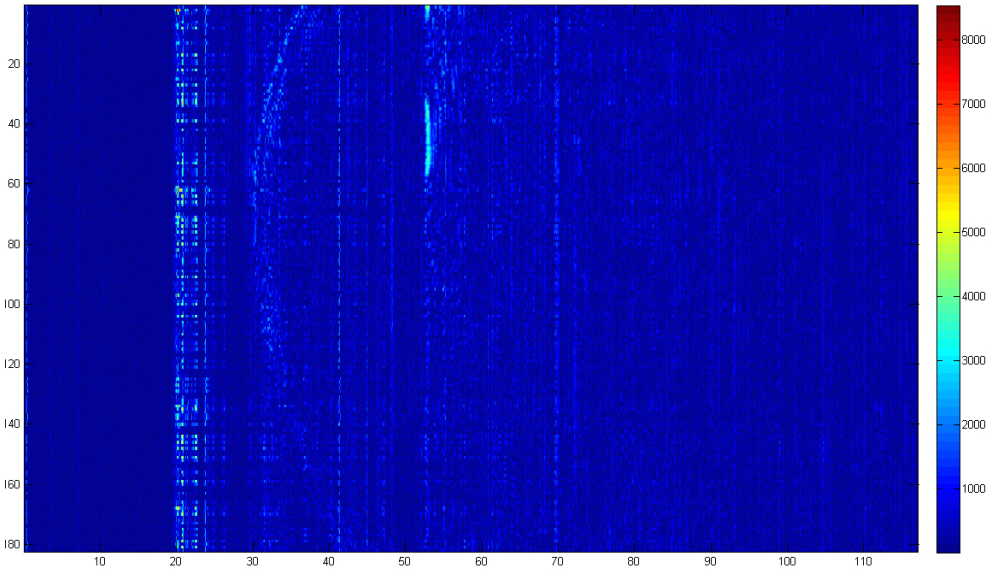


Figura 8.9: Rappresentazione del movimento del *target* nel tempo nel caso *AT-I-RC*.

si miglioramenti, indi per cui si è passati al terzo tipo di elaborazione.

8.3 Inviluppo, Allineamento e Rimozione del *Clutter* (*I-AT-RC*)

Nell'ultimo caso analizzato in questa tesi, cambia la sequenza di esecuzione delle elaborazioni: si parte con l'inviluppo delle due forme d'onda separatamente, poi si allineano fra di loro gli inviluppi ed infine avviene la rimozione del *Clutter*.

8.3.1 *Post* Inviluppo

In prima istanza viene quindi ottenuto l'inviluppo sulle forme d'onda salvate direttamente dalle misure sperimentali, per cui si ottengono due segnali non allineati nel tempo che si riferiscono al *target* (in blu) e all'*Empty Room* (in rosso)

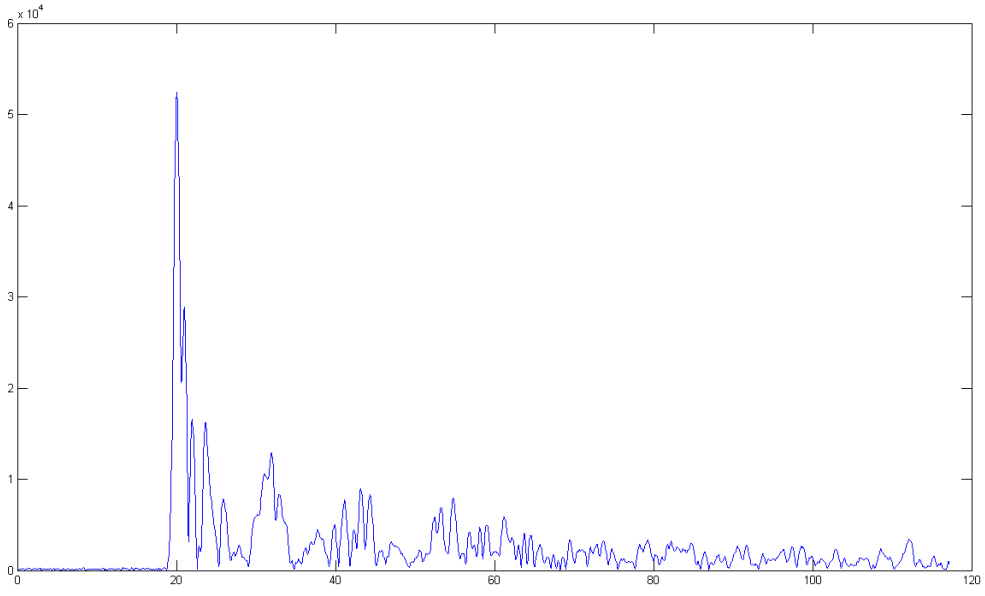


Figura 8.10: Rappresentazione nel tempo dell'inviluppo del segnale con la presenza del *target*.

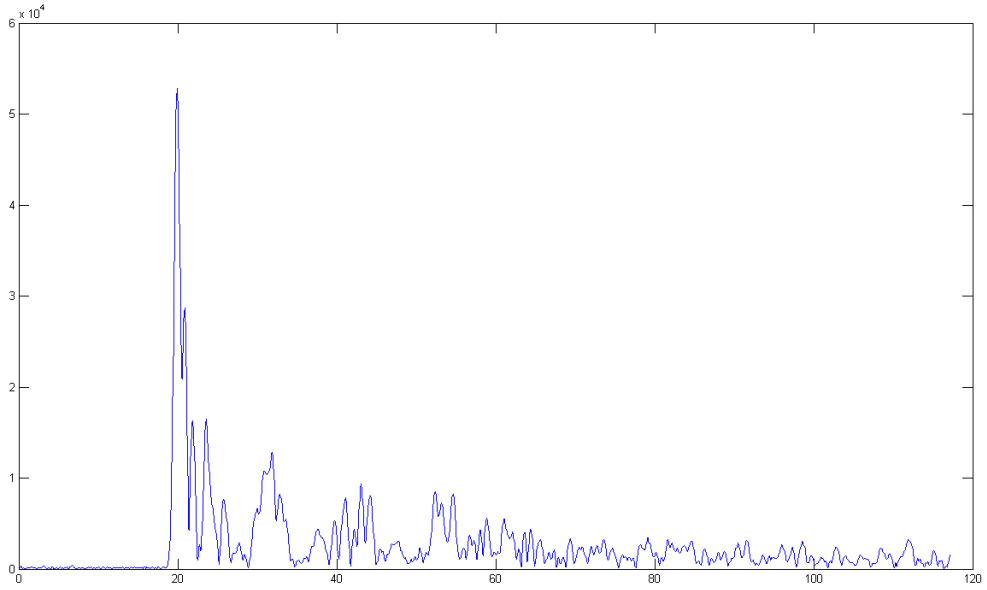


Figura 8.11: Rappresentazione nel tempo dell'inviluppo del segnale nel caso dell'*Empty Room*.

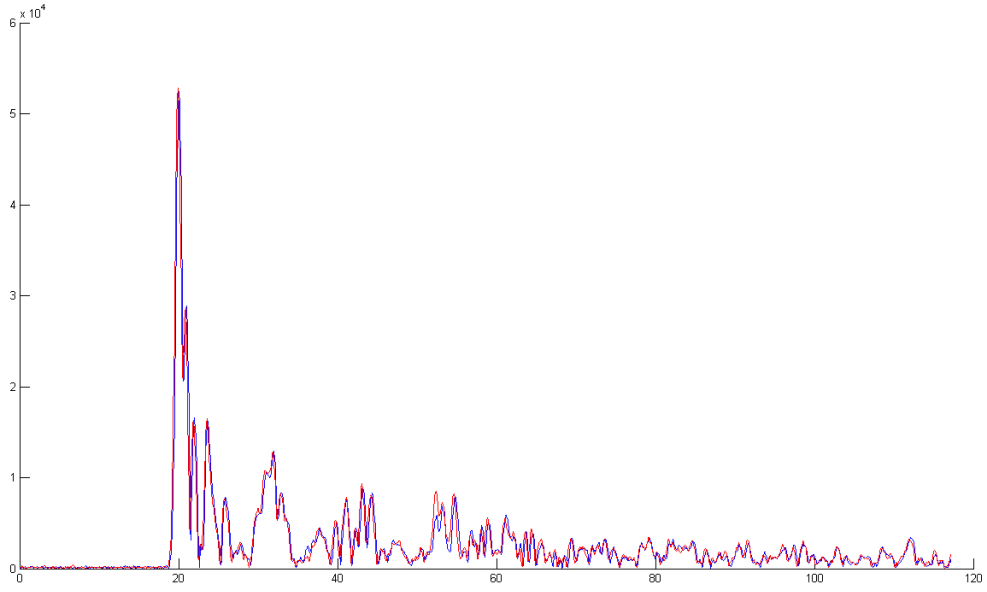


Figura 8.12: Rappresentazione delle due forme d'onda d'inviluppo assieme; si nota che non vi è ancora l'allineamento fra le due; in rosso è rappresentata l'*Empty Room*, in blu il *target*.

Si nota come, anche in questo caso (Figura 8.12) intorno ai 36 ns, è distinguibile la presenza del *target*, la quale verrà enfatizzata con le successive elaborazioni.

8.3.2 *Post* Allineamento

In questo caso l'allineamento avviene fra l'inviluppo delle due forme d'onda e non più attraverso il segnale salvato direttamente dai file *.csv*. In Figura 8.13 si può notare il quasi perfetto allineamento delle due forme d'onda rispetto al caso Pre-Allineamento della Figura 8.12.

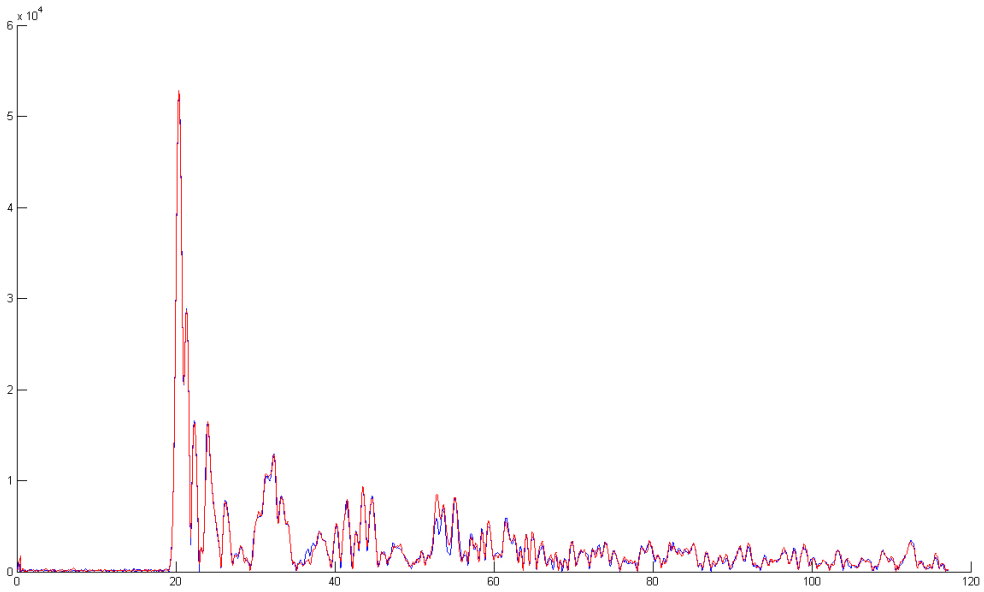


Figura 8.13: Rappresentazione delle due forme d'onda d'inviluppo assieme dopo aver eseguito l'allineamento; in rosso è rappresentata l'*Empty Room*, in blu il *target*.

8.3.3 *Post* Rimozione del *Clutter*

La rimozione del *Clutter* avviene quindi tramite la sottrazione diretta dell'inviluppo delle due forme d'onda allineate viste in Figura 8.13.

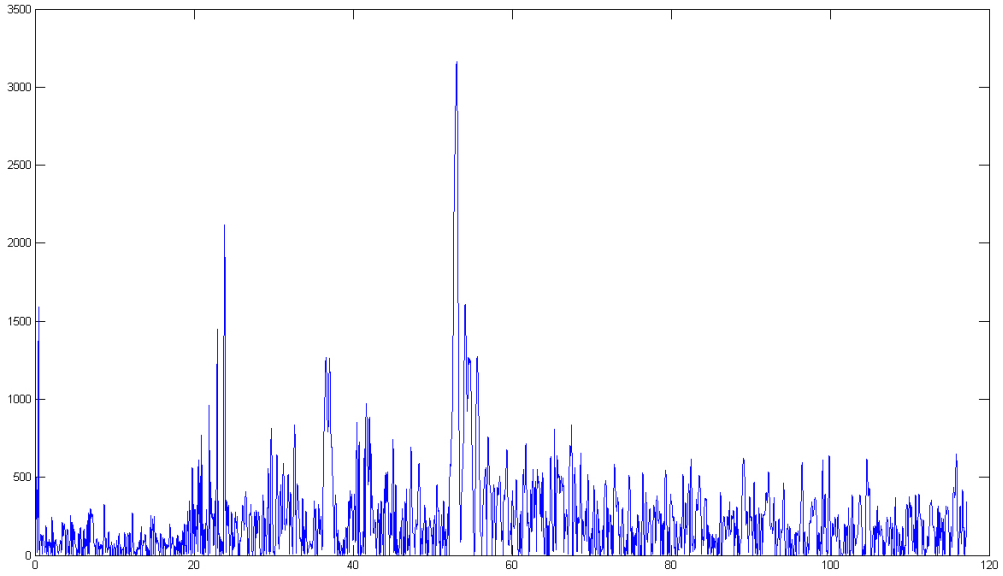


Figura 8.14: Rappresentazione dell'immagine della singola forma d'onda dopo tutta l'elaborazione (*I-AT-RC*).

8.3.4 Rappresentazione finale *I-AT-RC*

Vengono infine visualizzati i risultati finali per tutte le scansioni del target nel caso di questa sequenza di elaborazione (vedi Figura 8.15).
Si nota che in questo caso il risultato è leggermente migliore rispetto alle altre due sequenze di elaborazione.

8.4 Allineamento in Finestra e Raddrizzatore

Dai risultati delle tre sequenze di elaborazioni (Figure 8.4, 8.9 e 8.15) si possono notare alcuni problemi che potrebbero essere rilevanti nella *detection* e nel successivo *tracking* del *target* nell'area d'osservazione.
In primis vi è la presenza ancora molto rilevante del cammino diretto intorno ai 20 ns, inoltre vi sono dei picchi a circa 53 ns per le realizzazioni che vanno dalla posizione 35 alla 60; tutto ciò è derivato dalla rimozione del clutter che nei capitoli precedenti è stata fatta brutalmente sottraendo le due forme d'onda (*target* - *Empty Room*) e rappresentandone il valore assoluto. Quest'ultima operazione non è del tutto perfetta, poiché potrebbe esservi un leggero non allineamento fra le due forme d'onda, il quale determina dei picchi negativi. Intorno ai

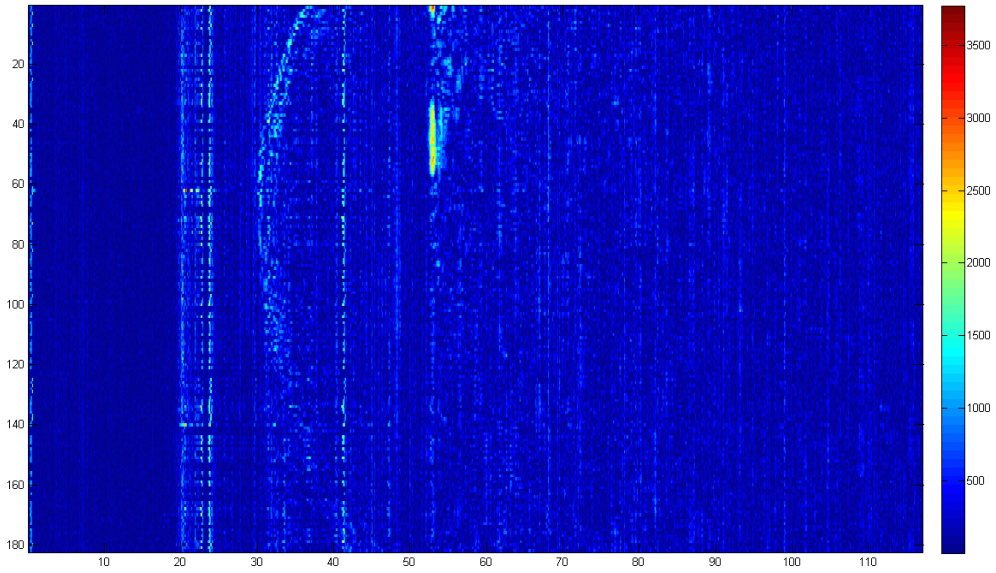


Figura 8.15: Rappresentazione del movimento del *target* nel tempo nel caso *I-AT-RC*.

20 ns il cammino diretto presenta alcuni di questi picchi, così come, a circa 53 ns, una parte del *clutter* viene oscurato dalla presenza del *target*. Quindi, la sottrazione delle forme d'onda in presenza e assenza del *target* e la successiva rappresentazione in valore assoluto, determinano dei picchi nelle scansioni.
Per risolvere questi problemi ed avere quindi un allineamento migliore, sono stati implementati due fondamentali modifiche ai risultati ottenuti:

- **Allineamento in Finestra**;
- **Raddrizzatore**.

8.4.1 Allineamento in Finestra

Questo metodo ha come obiettivo quello di effettuare un allineamento migliore rispetto al caso utilizzato nei paragrafi precedenti. Infatti, come si nota in Appendice C.2, l'allineamento fatto finora prende in considerazione tutta la forma d'onda dai -20 ns ai 97 ns. Invece con questo nuova implementazione (descritta in Appendice C.3) si considera un allineamento nel solo cammino diretto (quindi nella finestra $[-10 \div 20]$ ns). Trovato lo scostamento delle forme d'onda del *target*

con l'*Empty Room* nella sola finestra, si riporta poi il tutto all'intera forma d'onda al fine di avere l'allineamento globale da quello nella finestra del cammino diretto.

8.4.2 Raddrizzatore

Questa elaborazione è tanto semplice quanto efficace; infatti l'utilizzo di un raddrizzatore assieme al nuovo allineamento a finestra, porta ad un miglioramento netto dei risultati delle Figure 8.4, 8.9 e 8.15. Il raddrizzatore viene applicato in tutte le elaborazioni (viste nei paragrafi 8.1 8.2 8.3) nella parte relativa alla rimozione del *Clutter*. Questa elaborazione elimina i picchi negativi dovuti alla sottrazione *target* con *Empty Room* i quali non sono rilevanti nel rilevamento del segnale di riflessione dell'oggetto nell'area, anzi ne peggiorano la visualizzazione (vedi paragrafo 8.4).
Il raddrizzatore sostanzialmente funziona con queste poche righe di codice:

```
diff=Target-EmptyRoom; %diff = differenza campione per campione
    fra target e empty room
DiffRadd=(abs(diff)+diff)/2; %Qui invece si ha il raddrizzatore
    che pone a zero i campioni che sono sotto zero della
    differenza
```

8.4.3 Rappresentazione finale Allineamento in Finestra e Raddrizzatore

In questo paragrafo vengono mostrate le figure finali del percorso del target nei casi di sequenza di elaborazione d'interesse (*AT-RC-I*, *AT-I-RC* e *I-AT-RC*).

Dalle Figure 8.16, 8.17 e 8.18 si nota come parte del cammino diretto e del *Clutter* (del quale si discuteva nel paragrafo 8.4) viene fortemente attenuato rispetto ai risultati senza Raddrizzatore ed Allineamento in Finestra (vedi rispettivamente le Figure 8.4, 8.9 e 8.15). É importante anche notare come la singola forma d'onda sia nettamente cambiata e migliorata con l'elaborazione derivata dal nuovo Allineamento a Finestra e dal Raddrizzatore. In Figura 8.19 si nota molto bene come, tramite l'implementazione del Raddrizzatore e dell'Allineamento in Finestra, ad esempio il picco a 53 ns sia praticamente annullato e come invece la presenza del *target* (a circa 37 ns) sia messa in rilievo.

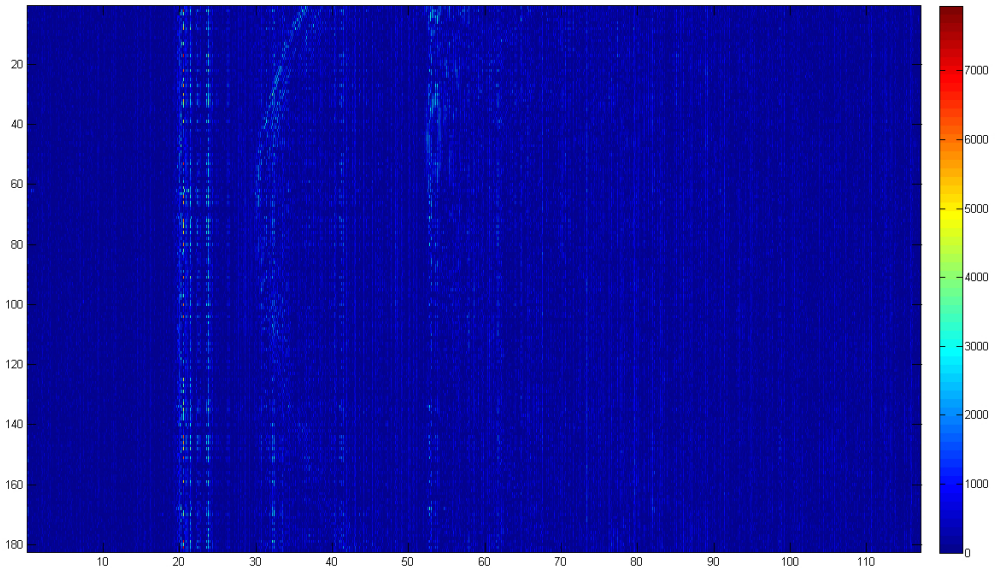


Figura 8.16: Rappresentazione del movimento del *target* nel tempo nel caso *AT-RC-I* con Allineamento in finestra $[-10 \div 20]$ *ns* e Raddrizzatore.

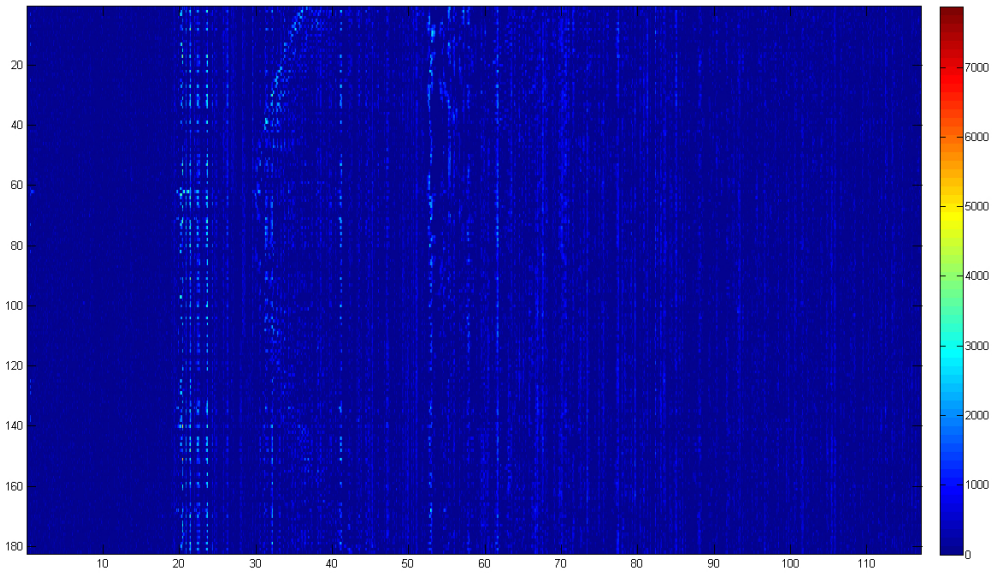


Figura 8.17: Rappresentazione del movimento del *target* nel tempo nel caso *AT-I-RC* con Allineamento in finestra $[-10 \div 20]$ *ns* e Raddrizzatore.

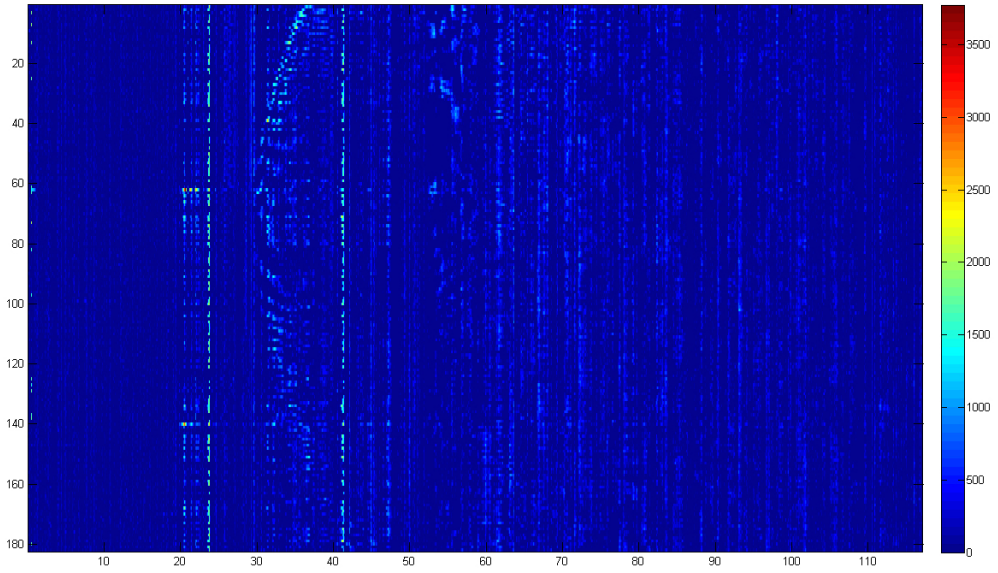


Figura 8.18: Rappresentazione del movimento del *target* nel tempo nel caso *I-AT-RC* con Allineamento in finestra $[-10 \div 20]$ ns e Raddrizzatore.

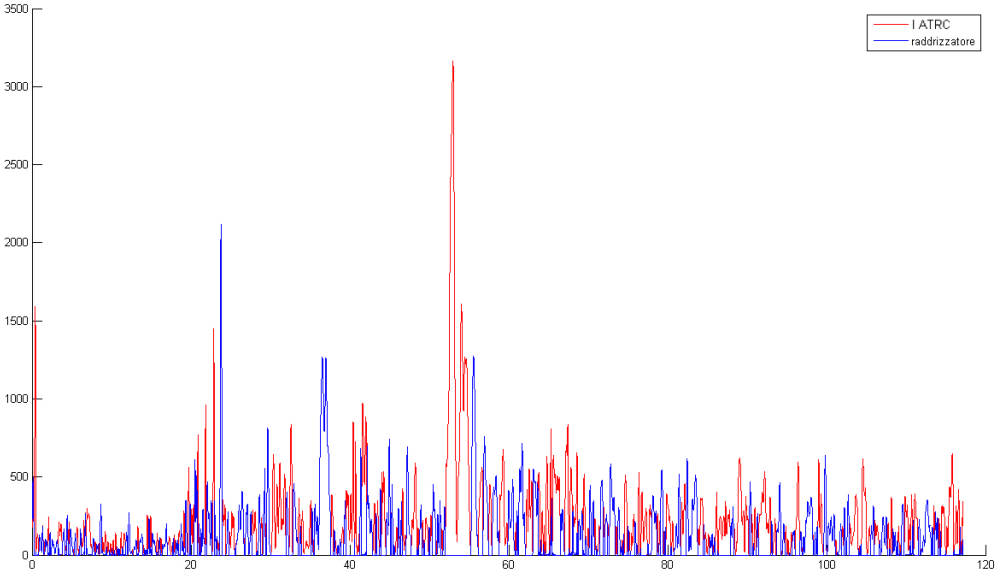


Figura 8.19: Singola scansione alla Posizione zero che confronta la forma d'onda con raddrizzatore e allineamento in finestra (blu) e quella senza (rosso).

8.5 Integrazione di un Numero Maggiore di Forme d'Onda

I risultati ottenuti nei precedenti paragrafi cominciano ad essere rilevanti per la visualizzazione del *target*, però, in questo elaborato di tesi, è stato scelto di implementare un ulteriore miglioramento alle forme d'onda viste nel precedente paragrafo. Infatti, durante la fase di acquisizione delle misure sperimentali, sono state salvate più forme d'onda relative alla singola posizione del *target* e dell'*Empty Room*. Per cui si è scelto di integrare 20 di queste scansioni in un'unica al fine di aumentare ulteriormente di circa 13 dB il rapporto segnale-rumore (SNR) della forma d'onda salvata. Quindi per prima cosa vengono considerate le 20 scansioni, vengono allineate (tramite l'algoritmo descritto nello specifico in Appendice C.1) e infine si ha in uscita un'unica scansione che è la media di tutte quelle considerate.

8.5.1 Risultati con Integrazione di venti scansioni

In questo paragrafo vengono riportati i risultati di tutte implementazioni effettuate finora in questo capitolo, per i tre casi di sequenze d'elaborazione. Le Figure 8.20, 8.21 e 8.22, sono riferite alle sequenze di elaborazione analizzate in questo paragrafo, quindi considerando un'integrazione di venti scansioni in una per ogni posizione, allineamento temporale in una finestra sul cammino diretto e il raddrizzatore.
Si nota come, grazie all'integrazione, il *Clutter* e soprattutto il cammino diretto, vengano nettamente attenuati a fronte di una sempre migliore visibilità della presenza del *target* che si muove (come viene descritto anche in Appendice C.1 infatti si ha un aumento del rapporto segnale rumore).

8.6 CFAR e *Clustering*

In questo paragrafo si considerano tutti i miglioramenti progressivamente raggiunti durante le varie elaborazioni esaguite durante il lavoro di tesi e descritte in questo capitolo, al fine di implementare su di essi gli algoritmi utilizzati nelle simulazioni (vedi Capitolo 3). Infatti si è scelto di prendere i risultati migliori ottenuti dalle elaborazioni fatte finora e di applicarne l'algoritmo CFAR sui campioni *soft* (vedi Capitolo 3.1) ed infine l'algoritmo di *Clustering* creato da zero per questo elaborato (vedi Capitolo 3.2.3).
Per le successive conclusioni si è quindi scelto di utilizzare la sequenza

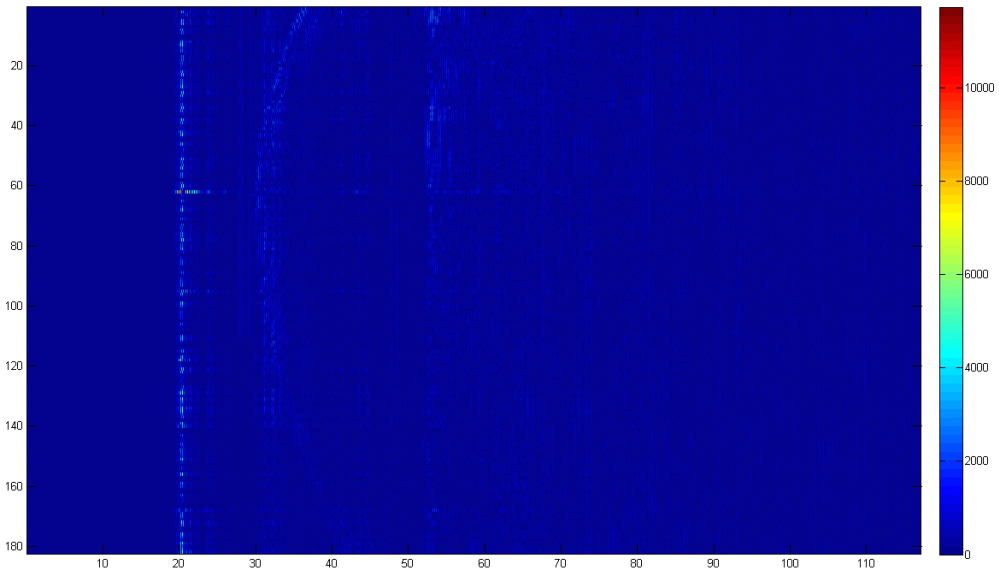


Figura 8.20: Rappresentazione del movimento del *target* nel tempo nel caso *AT-RC-I* con Integrazione di 20 forme d'onda, Allineamento in finestra $[-10 \div 20]$ ns e Raddrizzatore.

di elaborazioni Inviluppo, Allineamento Temporale e Rimozione Clutter (*I-AT-RC*) poichè dai precedenti risultati (vedi Figura 8.22), si è riscontrato che è la miglior rappresentazione riscontrabile del *target*.

8.6.1 Algoritmo CA-CFAR

Considerando quindi le misure sperimentali elaborate (rappresentate in Figura 8.22), si applica ad esse l'algoritmo CA-CFAR (vedi Capitolo 3.1) al fine di avere una matrice di soli 0 e 1, come avveniva nel caso delle simulazioni effettuate in precedenza (vedi Capitolo 3).

Dalla Figura 8.23 si nota come la maggior parte del *Clutter* venga rimosso e come sia messo in rilievo il percorso temporale che il *target* compie nell'area. Per cui grazie a questo algoritmo si ha una matrice composta da '0' dove non si stima la presenza del *target* (in Figura 8.23 vengono rappresentati con il colore bianco) e da '1' dove invece si è rilevato un *target* (o un residuo di *clutter*, in nero).
Si è quindi riusciti ad avere un riscontro uguale al caso simulato visto nel Capitolo 3, ottenendo matrici con scansioni tali da contenere solo un bit (o uno o zero) in base al supermento o meno della soglia im-

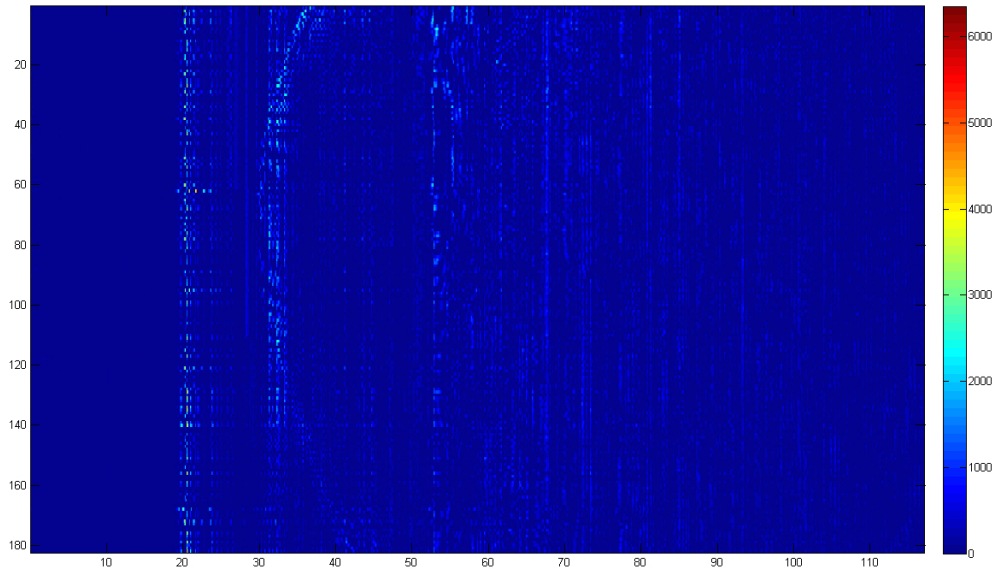


Figura 8.21: Rappresentazione del movimento del *target* nel tempo nel caso *AT-I-RC* con Integrazione di 20 forme d'onda, Allineamento in finestra $[-10 \div 20]$ ns e Raddrizzatore.

postata dal *Cell Averaging Constant False Alarm Rate*. In particolare, tramite l'esecuzione di numerose simulazioni, è stato riscontrato che i parametri migliori per il rilevamento del target (per queste misure sperimentali) sono quelli riferiti in Tabella 8.1. In questo caso i parametri impostati, sono relativi all'*Oversampling*, ossia il caso in cui si ha un ricampionamento di 1 campione in 10. Questa elaborazione viene fatta dall'allineamento mediante l'algoritmo LED (vedi Appendice C).

Impostazioni	**Valori**
Campioni di guardia (N_GUARD)	160
Campioni di riferimento (N_CFAR)	3200
Soglia CFAR (T_CFAR)	30

Tabella 8.1: Tabella delle impostazioni dell'algoritmo CA-CFAR per 19200 campioni.

.

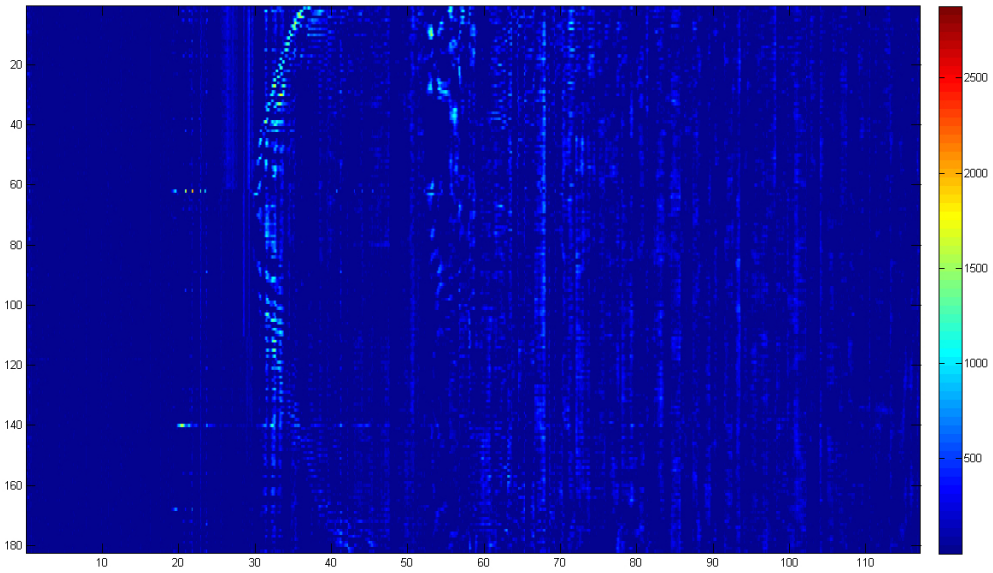


Figura 8.22: Rappresentazione del movimento del *target* nel tempo nel caso *I-AT-RC* con Integrazione di 20 forme d'onda, Allineamento in finestra $[-10 \div 20]$ ns e Raddrizzatore.

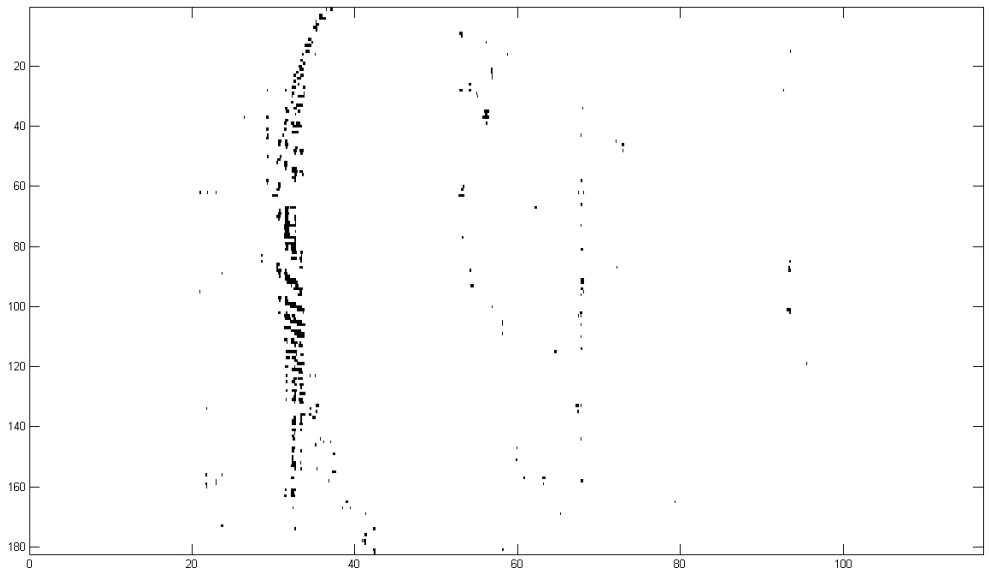


Figura 8.23: Rappresentazione del Ricevitore 1 dopo aver utilizzato l'algoritmo CA-CFAR.

8.6.2 *Clustering*

L'ultima elaborazione effettuata in questa tesi, è quella appunto di utilizzare l'algoritmo visto per le simulazioni (vedi Capitolo 3), sulle misure sperimentali, al fine di vedere come potesse modificare i dati reali in relazione ai dati teorici.
Per applicare quindi l'algoritmo di *Clustering Monodimensionale* (vedi Capitolo 4), si è quindi considerata la matrice delle misure sperimentali relative al *target* dopo il CA-CFAR, cercando di raggruppare tutti gli '1' relativi al percorso dell'intruso in un unico bit posto ad uno. Si nota come dalla Figura 8.24 la traccia del target non è molto più

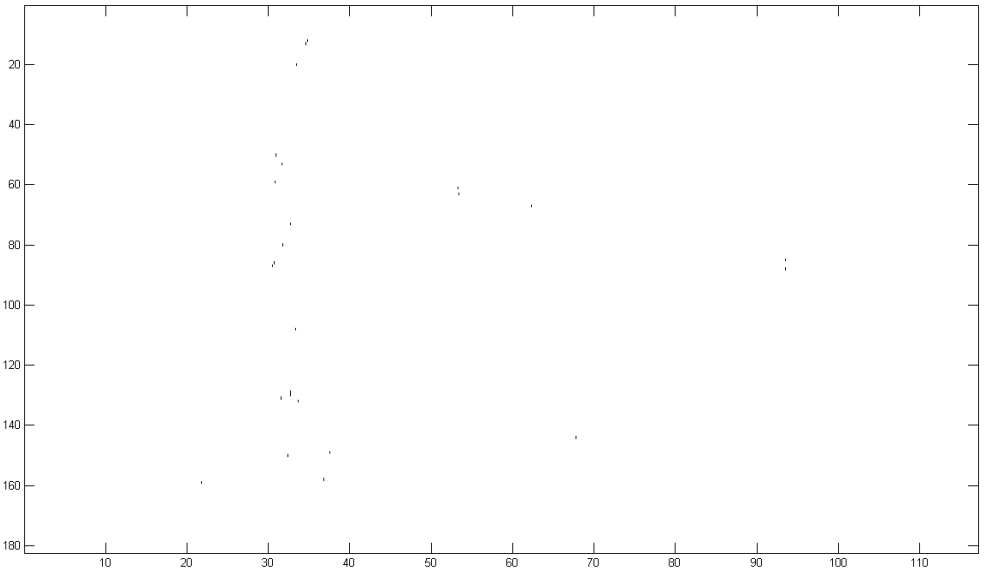


Figura 8.24: Rappresentazione del Ricevitore 1 dopo aver utilizzato l'algoritmo CA-CFAR e il *Clustering* Monodimensionale nel caso di matrici $[182 \times 19200]$.

visibile, almeno non in tutta la sua interezza. Questo è dovuto al fatto che le matrici utilizzate per questa elaborazione sono molto grandi ($[182 \times 19200]$ campioni) di conseguenza Matlab non riesce a far visualizzare tutti gli '1' presenti nella matrice oltre a rendere insostenibile il tempo impiegato per calcolo del *Clustering*. Infatti ingrandendo questa figura si può notare come i punti corrispondenti alla presenza del target, sono presenti nella matrice (vedi Figura 8.25).

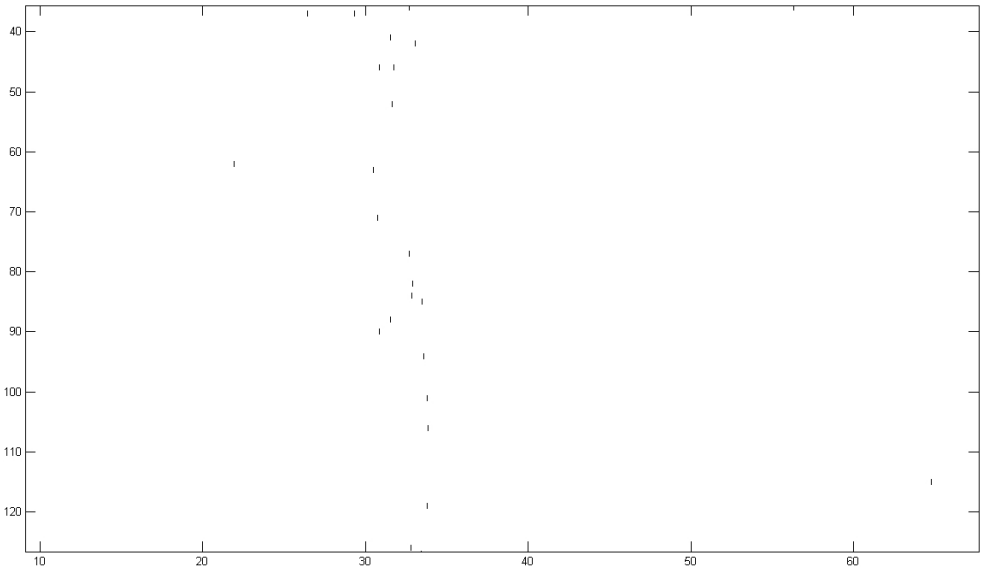


Figura 8.25: Rappresentazione del Ricevitore 1 dopo aver utilizzato l'algoritmo CA-CFAR e il *Clustering* Monodimensionale nel caso di matrici $[182 \times 19200]$, ingrandendo la figura per rilevare la presenza degli '1' relativi al *target*.

Per risolvere il problema del tempo di elaborazione troppo elevato di Matlab e della scarsa visualizzazione dovuta all'elevato numero di campioni, si è scelto di applicare un *Down-sampling* su ogni singola scansione, al fine di ridurre di un fattore 10 il numero di campioni.

8.7 *Down-sampling*, CFAR e *Clustering*

Il *Down-sampling* è stato applicato in modo analogo ma inverso, al ricampionamento visto per l'allineamento delle forme d'onda (vedi Appendice C), subito dopo le elaborazioni di Inviluppo, Allineamento Temporale e Rimozione del Clutter (ossia dopo le modifiche per le quali si ottiene la Figura 8.22). Si nota come, dopo questa riduzione del numero di campioni, le Figure 'Post CA-CFAR' (vedi Figura 8.26) siano sostanzialmente uguali a quelle nel caso di 19200 campioni (vedi Figura 8.23). Per il caso *Down-sampling*, sono state utilizzate le impostazioni che possono essere viste in Tabella 8.2, con un riscontro visivo nelle figure succesive.

Viene quindi ora implementato l'algoritmo di *Clustering* Monodimensionale al fine ultimo di avere un'unica traccia del *target* nel tempo.

Impostazioni	Valori
Campioni di guardia (N_GUARD)	16
Campioni di riferimento (N_CFAR)	320
Soglia CFAR (T_CFAR)	30
Soglia di Clustering (thr)	70

Tabella 8.2: Tabella delle impostazioni dell'algoritmo CA-CFAR per 1920 campioni.

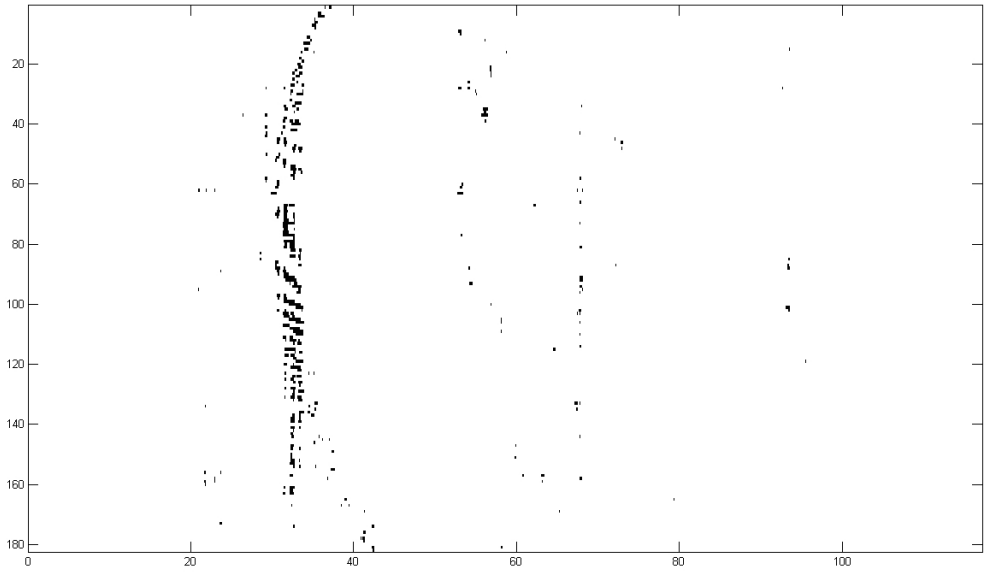


Figura 8.26: Rappresentazione del Ricevitore 1 dopo aver utilizzato l'algoritmo CA-CFAR nel caso di matrici [182×1920].

In Figura 8.27 si nota come, dopo il *Down-sampling*, CA-CFAR e *Clustering*, la traccia del *target* sia molto più visibile e netta grazie alla riduzione del numero di campioni da mettere a schermo, oltre alla riduzione del tempo di elaborazione di Matlab.

8.8 Risultati Finali

In ultima analisi vengono considerati i risultati finali, relativi a tutte le elaborazioni analizzate in questo documento, utilizzando le misure sperimentali dei ricevitori 1 e 2 (vedi Figura 8.27 per il ricevitore 1).

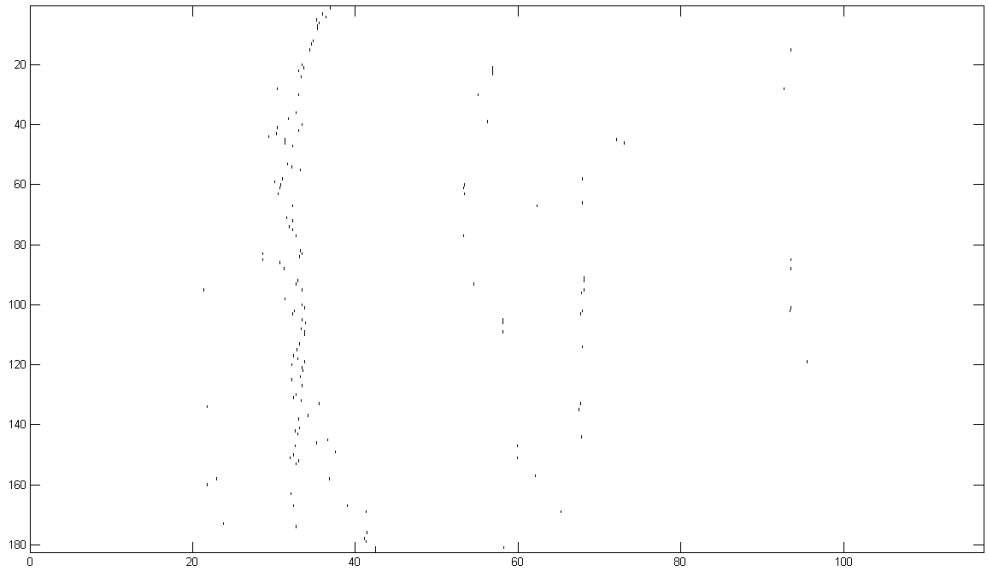


Figura 8.27: Rappresentazione del Ricevitore 1 dopo aver utilizzato l'algoritmo CA-CFAR e il *Clustering* Monodimensionale nel caso di matrici $[182 \times 1920]$.

Conoscendo la posizione del *target* e la sua distanza dai ricevitori e dal trasmettitore, è possibile calcolare il ritardo temporale dell'intruso, dal cammino diretto (vedi Appendice F). Le misure sperimentali elaborate per i due ricevitori, vengono quindi confrontate con questo risultato, nelle Figure 8.28 e 8.29. Si può quindi notare che i risultati di tutte le elaborazioni (rappresentati dalle tracce in nero), tendono a seguire la stima del ritardo del *target* (rappresentata mediante una traccia di colore rosso), confermando quindi la presenza dell'intruso in quel punto.

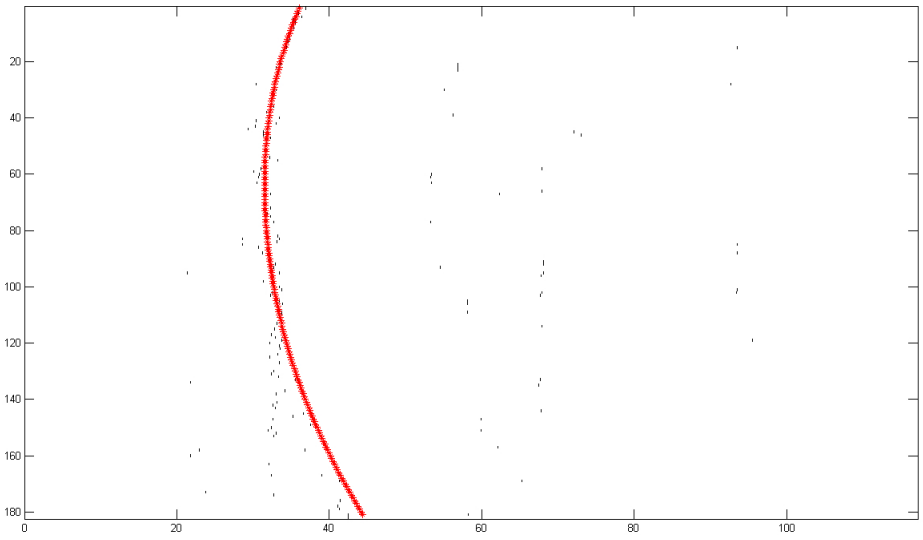


Figura 8.28: Rapresentazione della differenza fra misure sperimentali per il ricevitore 1 (in nero) dopo le elaborazioni sviluppate in questo elaborato e la posizione temporale reale stimata in base alla posizione del *target* (in rosso).

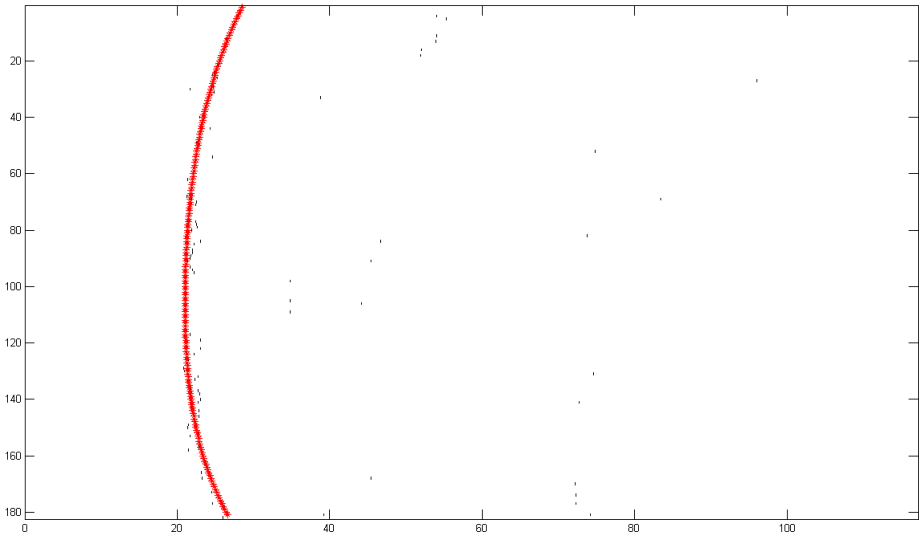


Figura 8.29: Rapresentazione della differenza fra misure sperimentali per il ricevitore 2 (in nero) dopo le elaborazioni sviluppate in questo elaborato e la posizione temporale reale stimata in base alla posizione del *target* (in rosso).

Capitolo 9

Conclusioni

In questo elaborato sono stati trattati molteplici argomenti relativi ad uno scenario di radar multistatico basato su tecnologia UWB, sia in ambito di simulazioni al calcolatore, che in ambito reale mediante misure sperimentali.
In prima analisi sono state considerate le simulazioni e l'implementazione di nuovi algoritmi di *Clustering* mono e bidimensionali per la *dection* di intrusi in un sistema radaristico. Mediante l'utilizzo di Matlab è stato possibile elaborare i risultati dell'utilizzo dei nuovi algoritmi nelle simulazioni, variando la soglia del CFAR e riscontrando i miglioramenti del tracciamento delle traiettorie dei target.
In seconda analisi invece sono state effettuate delle misure sperimentali, cercando di riprodurre uno scenario simile alle simulazioni, considerando quattro antenne PulsOn P410 (un trasmettitore e tre ricevitori), su un'area di osservazione quadrata, nella quale si muove un *target*. Tutti i dati derivanti dalle misure sono stati successivamente elaborati mediante numerosi algoritmi, sempre implementati grazie all'utilizzo di Matlab. A questo punto viene implementato l'algoritmo di *Clustering* sulle Misure sperimentali elaborate al fine di avere una matrice binaria che abbia un'unica traccia della stima del *ToA*, quindi del ritardo relativo al *target* in base alla posizione in cui si trova.
Conoscendo la posizione precisa dell'intruso nell'area si può quindi confrontare il risultato ottenuto con tutte le elaborazioni effettuate durante questa tesi, con il ritardo temporale in cui ci si aspetta sia il *target* (come viene descritto in Appendice F). Nelle Figure 8.28 e 8.29 vengono quindi riportati i risultati finali relativi al confronto fra traccia temporale prevista e i dati direttamente derivati dall'ultima elaborazione di *Clustering* monodimensionale.

Appendice A

Impostazione nodi P410

Le impostazioni utilizzate in tutti i nodi per le simulazioni e per le misure sperimentali sono state scelte al fine di cercar di avere i migliori risultati per l'area in cui sono stati posizionati.

A.1 Scelta della finestra di osservazione

Per scegliere che finestra catturare con le misure sperimentali, si è ipotizzato un caso peggiore che potrebbe accadere posizionando sia l'antenna trasmittente che quella ricevente ad un angolo del quadrato dell'area di osservazione (di grandezza $10 \times 10m$) e un eventuale *target* all'angolo opposto (il percorso che il segnale UWB dovrebbe percorre è disegnato in rosso in Figura A.1).

Conoscendo il lato (l) dell'area d'interesse si può calcolarne la diagonale (d) ed in particolare il tempo che impiega il segnale UWB per percorrere l'andata e il ritorno; la diagonale misura quindi:

$$d = l\sqrt{2} = 10\sqrt{2} \simeq 14.14\ m. \tag{A.1}$$

Il tempo di percorrenza del segnale (T_p) è quindi definito come due volte la diagonale (d) fratto la velocità della luce (c) ed è pari a:

$$T_p = \frac{2d}{c} \simeq 94.35\ ns. \tag{A.2}$$

Si riscontra quindi dal risultato di Formula A.2, che per rilevare nel caso peggiore la presenza di un target nell'area $10 \times 10m$ (utilizzata per le misure in questa tesi), bisogna acquisire una forma d'onda per almeno un intervallo temporale T che soddisfi la relazione:

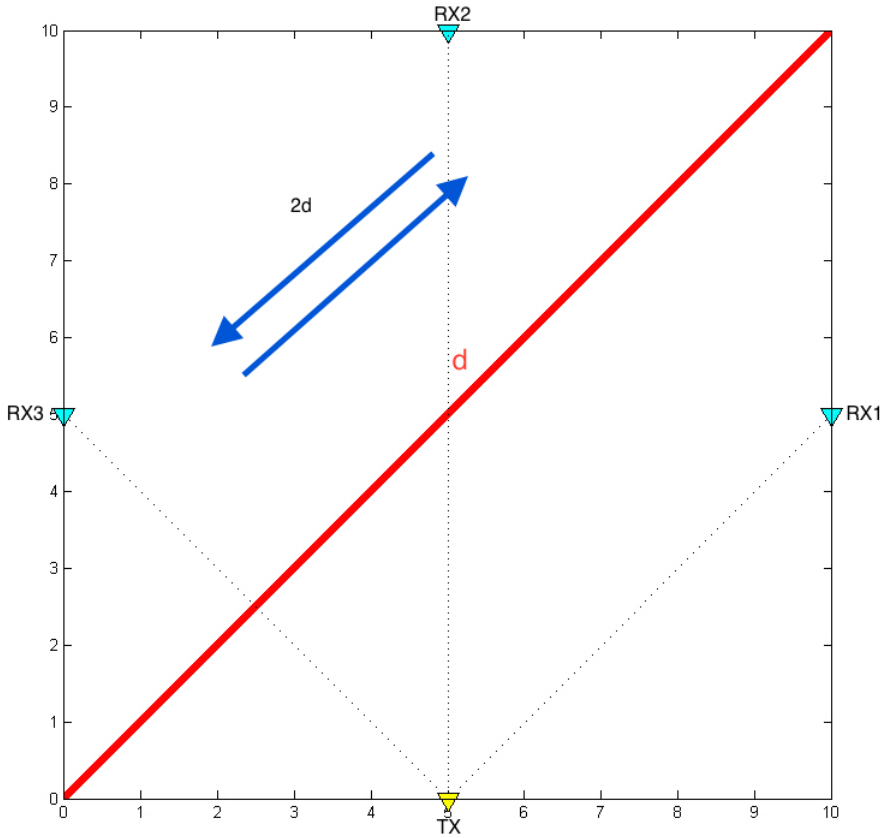


Figura A.1: Rappresentazione grafica del percorso massimo che il segnale deve percorrere nel caso peggiore di massima lontananza massima del target (in blu è rappresentata l'andata e il ritorno, in rosso la diagonale).

$$T > T_p \qquad T > 94.35\ ns. \tag{A.3}$$

Per questo è stato scelto come valore di *Stop (ns)*, 97 ns (come si può riscontrare in Tabella 7.1), oltre ad essere partiti con l'acquisizione (*Start (ns)*) a -20 ns per essere certi di salvare correttamente tutta la forma d'onda ricevuta.

A.2 Altre impostazioni

Altre configurazioni rilevanti per i nodi P410 sono quelle relative alla potenza trasmessa (Vedi *Transmit Gain* in Tabella 7.1) che è impostato al valore massimo (63) per cercare di migliorare la possibilità di

rilevare il *target*.
Il numero di *Bins* utilizzati per ogni acquisizione è 32 (valore impostato in *Step* nella Tabella 7.1) per cui (vedi[23]), conoscendo la duranta di ogni singolo bin si può ricavare il tempo di simbolo (T_s):

$$1\ bin \simeq 1.9073\ ps \tag{A.4}$$

$$32\ bins = 32 * 1\ bin \simeq 61.0336\ ps \tag{A.5}$$

$$T_s = 61.0336\ ps \tag{A.6}$$

Infine *Acq Index* è stato impostato in base alla Figura A.2; in particolare è stato scelto il caso *PII 6* ossia in cui i nodi hanno un *range* di utilizzo che può arrivare ai 60 m, sempre mantenendo *Data Integration* in automatico, ossia ad un valore minore di un'unità rispetto all'*Acq Index*.
Secondo le misure dell'area sorvegliata poteva essere scelto anche un

Acquisition PII	**DATA PII**	**Max Range (meters)**	**Raw Data Rate: (bps)**
5	4	35	632k
6	5	60	316k
7	6	88	158k
8	7	125	79k
9	8	177	39.5k
10	9	250	19.7k
11	10	354	9.86k

Figura A.2: Tabella delle possibili impostazioni del valore di *Acq Index* e relativi *range* e *Data Rate*.

Acq Index pari a 5 (ossia con *Range* massimo pari a 35 m), però si è scelta l'altra opzione per mantenere un fattore di integrazione finale un po' più alto (*Index : Integration*) e pari a 1024.

Appendice B

Acqusizione dati per l'elaborazione

Come detto nel Paragrafo 7.5, la GUI del *software CAT*, permette di salvare in file *.csv* tutte le forme d'onda con relative informazioni (come visto in Figura 7.8); in questa appendice verrà descritta la procedura di caricamento dei *Logging File* creati in Matlab.
I file salvati sono organizzati in cartelle composte da tre *file* (in base che si riferiscano al primo 1.*csv*, al secondo 2.*csv*, o al terzo ricevitore 3.*csv*) e nominate in base alla distanza dal punto di partenza (esempio: la cartella 'dist120-int32' si riferisce alla misurazione alla distanza di 120 *cm* dal punto di partenza del target). Dopo aver salvato ed organizzato tutto, è stato creato un *file* in codice Matlab al fine di caricare una delle scansioni di uno specifico ricevitore in un unica matrice di grandezza [182 $\times$ 1920], dove 182 sono le misure delle rispettive posizioni del target lungo i 9.05 m della traiettoria, mentre 1920 sono il numero di campioni salvati ad ogni scansione. Nel codice riportato di seguito viene infatti caricato in un'unica matrice dal nome *SingleData* la prima scansione di tutte le posizioni per il target, nel caso del ricevitore 1 e quella ripetuta nel caso dell'*Empty Room*. Con le elaborazioni successive si giungerà a confrontare le misure con *target* con quelle dell'*Empty room* per giungere alla detection e ad una visualizzazione della presenza del target.

```
i=0;
j=1;
FilePath='C:\Users\e\Desktop\SIMULAZIONI PALLA\dist';
while j<=182
    istr=num2str(i);
    FilePath1=strcat(FilePath,istr,'-int32\1.csv');
```

```
    eval(sprintf('scan%d = xlsread(FilePath1)',i));
    eval(sprintf('ScanData%d = ScanDataPure(scan%d)',i,i));
    eval(sprintf('SingleData(j,:) = ScanData%d(1,:)',i));
    i=i+5;
    j=j+1;
end
scan_empty=xlsread('C:\Users\e\Desktop\SIMULAZIONI PALLA\empty-
    int32\1.csv');
Scan_empty_Data=ScanDataPure(scan_empty);
SinglEmpty=Scan_empty_Data(1,:);
j=1;
while j<=182
    SinglEmpty182(j,:)=SinglEmpty(j,:);
    j=j+1;
end
```

Alla fine dell'elaborazione si possono quindi riscontrare nel *Workspace* di Matlab le due matrici fondamentali per le elaborazioni dei dati successivi (vedi Figura B.1).

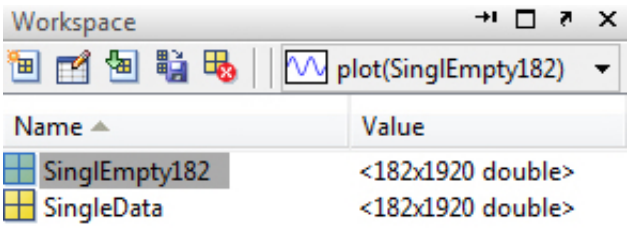


Figura B.1: Immagine del *Workspace* dopo il caricamento di tutte le misure sperimentali: vi sono due matrici fondamentali per le elaborazioni successive.

Appendice C

Allineamento

Per allineamento si intende la sincronizzazione di due forme d'onda nel dominio del tempo in base al ToA del cammino diretto. Per elaborare bene i dati e quindi per la successiva rimozione del *Clutter* è fondamentale che le due forme d'onda siano perfettamente allineate.
Per eseguire questa operazione è stato necessario l'utilizzo di alcuni passaggi di elaborazione della forma d'onda in ingresso; l'allineamento considera un algoritmo *Leading Edge Detection* o *LED*[26] (utilizzato anche indirettamente in alcune applicazioni della Time Domain). Questo algoritmo prende come riferimento una forma d'onda della quale ricerca il campione che, elevato al quadrato, è maggiore di 1/3 del massimo picco del segnale stesso. Il *Leading Edge* è il primo valore dopo lo zero temporale che supera questa soglia. Tutte le altre forme d'onda vengono poi allineate a quella di riferimento tramite la funzione di correlazione. Di seguito vengono riportate tutte le varianti di questo algoritmo che sono state utilizzate in questo elaborato di tesi.

C.1 Allineamento di 20 scansioni *(LED_alg_20_in_1)*

L'esempio di codice Matlab riportato in questo paragrafo è riferito all'allineamento di venti scansioni della stessa misura sperimentale utilizzato quindi per l'integrazione di altrettante forme d'onda in un'unica (vedi Capitolo 8.5). Questa implementazione è stata eseguita al fine di aumentare il rapporto segnale-rumore di un fattore pari a circa $13dB$, quindi per migliorare la visibilità del target e per cercare di eliminare il più possibile la rilevanza del *clutter* nelle misure sperimentali.

```
picco=(max(xp(1,:)))^2; %Si calcola il picco come valore massimo
    del campione al quadrato
soglia=picco/3; %Si calcola la soglia da confrontare
for i_led=1:length(xp(1,:))
    if (xp(1,i_led))^2 > soglia
        leadingEdge=i_led; %Viene trovato il Leading Edge
        break;
    end
end
pos_led=i_led-328; %Lo 0 temporale `e sul campione 328
xp(1,:)=circshift(xp(1,:), [1 -pos_led]);
while l<=20 % In questo caso vengono allineate le 20 successive
    scansioni
    xp(l,:)=x(l,:);
    [a b]=max(xcorr(xp(1,:),xp(l,:))); %Viene allineata la l-esima
         scansione con la prima
    pos=length(xp(l,:))-b; %Viene trovata la posizione dei
        campioni per l'allineamento
    xp(l,:)=circshift(xp(l,:), [1 -pos]); %Si allinea la forma d'
        onda l-esima con la prima
    l=l+1;
end
```

C.2 Allineamento *Target/Empty Room (LED_alg)*

Nel caso dell'allineamento per la rimozione del clutter (vedi Capitolo 8) il codice è praticamente identico a quello visto nel paragrafo precedente, salvo per il fatto che l'allineamento viene fatto solo tra due forme d'onda: quella con la presenza del target e l'altra relativa all'*Empty Room*. L'unica discrepanza si ha nel fatto che, per migliorare la precisione di allineamento, la forma d'onda è stata ricampionata di un fattore pari a 10; ciò può essere visto nel codice riportato in seguito.

```
x=resample(x,10,1); %Ricampionamento Segnale con target
picco=(max(x))^2;
soglia=picco/3;
for i_led=1:length(x)
%ricampiono con un fattore 10
%calcolo il picco al quadrato
%calcolo la soglia
    if (x(i_led))^2 > soglia
        leadingEdge=i_led;
        break;       %termino il ciclo for se trovo il leadingEdge
    end
end
pos_led=i_led-3278; %lo 0 temporale `e sul campione 3145
x=circshift(x, [1 -pos_led]);
y=resample(y,10,1); %Ricampionamento Segnale Empty Room
```

```
[a b]=max(xcorr(x,y));
pos=length(y)-b;
y=circshift(y, [1 -pos]);
```

C.3 Allineamento in finestra ridotta *(LED_alg_window)*

Per migliorare ulteriormente la prestazioni si è scelto di eseguire l'allineamento delle forme d'onda direttamente sulla finestra temporale relativa al cammino diretto del segnale UWB e non su tutti i 117 ns salvati. Per cui si è considerata la finestra che va da -10 ns dallo zero temporale (nella Figura C.1 lo zero temporale si riferisce ai 20 ns poiché, come descritto nel Capitolo 7, la forma d'onda catturata va dai -20 ns ai 97 ns) ai 20 ns, poichè in essa è presente, con buon margine, tutto il cammino diretto.
Di seguito viene quindi riportato come il codice Matlab è stato modificato per considerare i campioni riferiti alla nuova finestra su cui fare l'allineamento e come riportare i risultati sulla forma d'onda completa.

```
i=164;
while i<656
    wind(:,i-163)=x(:,i); %Viene creata la finestra per le misure
        relative al target
    windemp(:,i-163)=y(:,i);%Viene creata la finestra per le
        misure relative all'Empty Room
    i=i+1;
end
wind=resample(wind,10,1); %Ricampionamento della finestra del
    target
picco=(max(wind))^2;
soglia=picco/3;
for i_led=1:length(wind)
    if (wind(i_led))^2 > soglia
        leadingEdge=i_led;
        break;
    end
end
pos_led=i_led-1640;
wind=circshift(wind, [1 -pos_led]);
windemp=resample(windemp,10,1); %Ricampionamento della finestra
    dell'Empty Room
[a b]=max(xcorr(wind,windemp));
pos=length(windemp)-b; %Indice di allineamento fra le due forme d'
    onda finestrate
windemp=circshift(windemp, [1 -pos]);
x=resample(x,10,1); %Ricampionamento della forma d'onda intera del
     target
```

```
y=resample(y,10,1); %Ricampionamento della forma d'onda intera
    dell'Empty Room
x_out=circshift(x, [1 -pos_led]); %Allineamento target
y_out=circshift(y, [1 -pos]); %Allineamento Empty Room
```

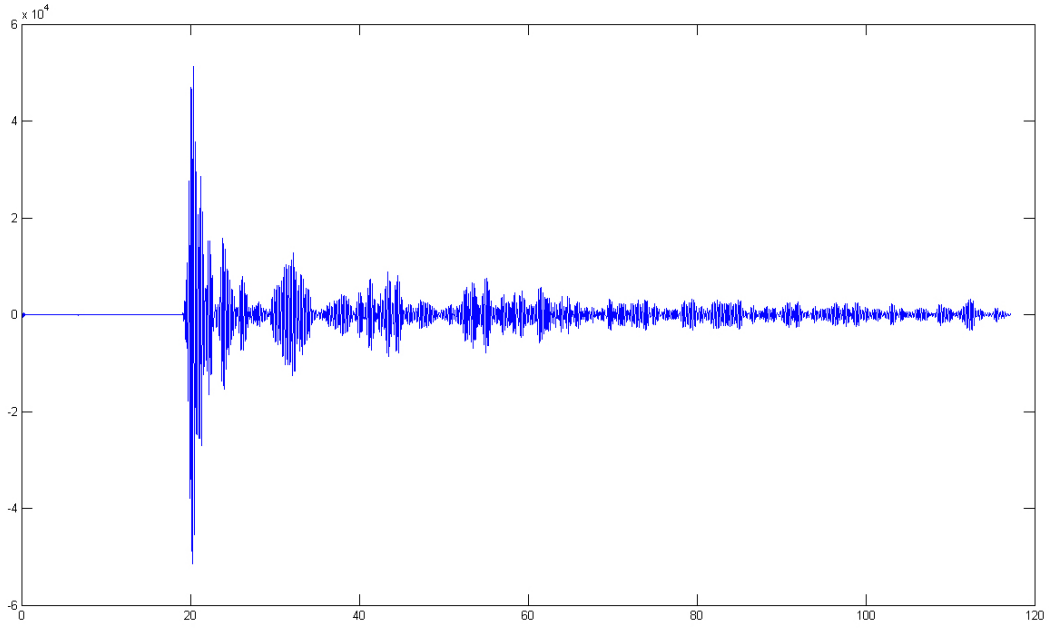


Figura C.1: Figura di una acquisizione salvata e successivamente aperta ed elaborata tramite Matlab. Nota bene il cammino diretto (picco intorno ai 20 ns che è lo zero temporale).

Appendice D

Inviluppo

L'inviluppo è una parte dell'elaborazione eseguita sulle forme d'onda, derivate dalle misure sperimentali, al fine di facilitare la manipolazione del segnale stesso. Come si nota dalla Figura C.1 (che rappresenta la visualizzazione di una singola scansione in presenza di target dopo l'acquisizione in Matlab tramite la procedura descritta nell'Appendice B), la forma d'onda dei PulsOn 410 può essere vista come un'oscillazione modulata a prodotto e descritta quindi dalla formula

$$s(t) \simeq a(t)\ \cos(2\pi f_0 t\ +\ \theta). \tag{D.1}$$

Considerando le impostazioni del segnale generato direttamente dai nodi della Time Domain[25] (poichè purtroppo non vi sono specifiche complete sulla tipologia di segnale utilizzato), indicativamente il segnale $s(t)$ possiede la struttura (D.1). L'inviluppo sfrutta alcune tecniche di demodulazione per giungere ai risultati sperati cercando di migliorare il segnale in ingresso anche attraverso il filtraggio. I parametri impostati sono stati presi direttamente dalle informazioni rilevate dal *Datasheet* dei P410:

- f_0 è la frequenza centrale del segnale ed è impostata a: 4.3 GHz.
- La banda operativa del segnale è invece $[3.1 \div 5.3]GHz$ di conseguenza di larghezza pari a 2.2 GHz.

L'estrazione dell'inviluppo si compone quindi di alcuni passaggi fondamentali; cosiderata la forma d'onda in ingresso (Formula D.1), la prima elaborazione effettuata riguarda l'elevamento al quadrato, ottenendo quindi

$$s'(t) \simeq a^2(t)\ \cos^2(2\pi f_0 t\ +\ \theta). \tag{D.2}$$

Conoscendo le formule goniometriche di bisezione, si ricava che il coseno al quadrato può essere scritto come

$$\cos^2(\alpha) = \frac{1}{2}\ +\ \frac{1}{2}\cos[2\alpha]. \tag{D.3}$$

Di conseguenza la forma d'onda dopo l'elaborazione e l'utilizzo delle formule goniometriche diventa

$$s'(t) \simeq \frac{1}{2}a^2(t)\ +\ \frac{1}{2}a^2(t)\cos[2(2\pi f_0 t\ +\ \theta)] \tag{D.4}$$

Si può quindi notare dalla (D.4) appena trovata, che vi è una componente in continua ($\frac{1}{2}a^2(t)$) e un'altra alla frequenza doppia di f_0. Di conseguenza la prossima elaborazione consiste nel filtrare la componente ad alta frequenza per avere come risultato la sola componente alla frequenza zero. Per fare ciò è stato quindi implementato un filtro FIR (*Finite Impulse Response*) passa-basso, tramite la funzione *fdatool* di Matlab, che tagli via le componenti ad alta frequenza e mantenga la continua con l'informazione del segnale derivande dalle misure sperimentali.
In Figura D.1 si possono vedere le impostazioni del filtro FIR e la sua rappresentazione grafica tramite *fdatool*; le principali impostazioni sono riferite al segnale in ingresso che deve essere tagliato almeno dopo 1.1 GHz rispetto alla continua e prima dei 7.5GHz:

- Viene utilizzato un filtro FIR *Equiripple*.
- Ordine del filtro : 10.
- *Passband Frequency* : 1.2 GHz.
- *Stopband Frequency* : 7.4 GHz.
- Frequenza di Campionamento : 17.2 GHz.
- *Density Factor* : 20.
- *Passband Weight* : 1.
- *Stopband Weight* : 1.

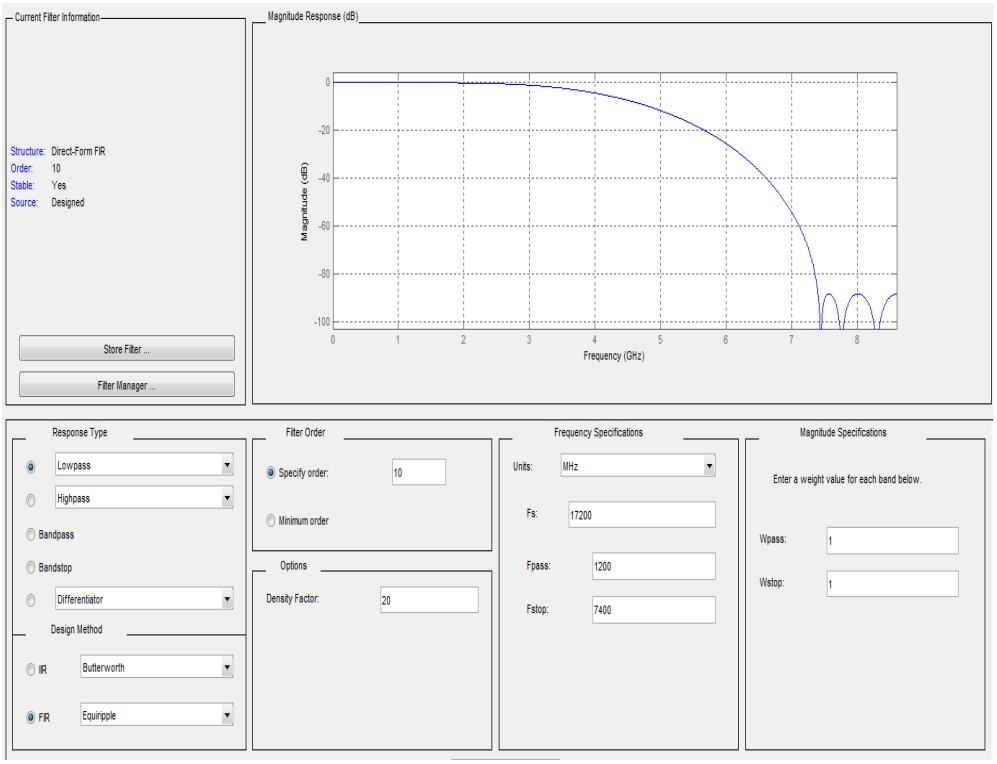


Figura D.1: Rappresentazione del filtro FIR utilizzato per le elaborazioni delle misure sperimentali tramite Matlab.

Trovato quindi il segnale in (D.4), viene dato in pasto al filtro appena generato al fine di tagliare la parte ad alta frequenza del segnale, ottenendo

$$s''(t) \simeq \frac{1}{2}a^2(t). \tag{D.5}$$

A questo punto, per ottenere l'inviluppo $|a(t)|$, basta moltiplicare tutto per 2 ed estrarre la radice quadrata.
Tutte queste operazioni fanno parte dell'Inviluppo sviluppato ed utilizzato in questa tesi per elaborare le forme d'onda direttamente ottenute con misure sperimentali dai nodi P410 ed è stato implementato tramite Matlab creando la funzione *InvilFilter*:

```
function [x_inv_filt] = InvilFilter(x,Num)
i=1;
```

```
while i<=182
    x2 = (x(i,:)).^2; %Elevamento al quadrato del segnale
    x2filt = FILTFILT(Num,1, x2); %Filtraggio FIR tramite i suoi
        coefficienti (Num)
    clear x2 %Pulizia variabili
    x2filt2 = x2filt*2; %Moltiplicazione per due
    clear x2filt %Pulizia variabili
    x_inv_filt(i,:)=sqrt(x2filt2); %Radice quadrata dei valori
    clear x2filt2 %Pulizia variabili
    i=i+1;
end
end
```

Appendice E

Rimozione *Clutter*

Il segnale salvato dalle misure sperimentali è corrotto da rumore ma anche da un contributo di riflessione dovuto a tutti gli oggetti presenti nell'ambiente di rilevazione: il *Clutter*. Questo effetto va quindi a peggiorare il segnale che deve essere poi utilizzato nelle elaborazioni al fine di trovare o meno la presenza di un eventuale *target*.
Esistono numerose tecniche per la rimozione più o meno parziale del *Clutter* (analizzate anche in [27]), tra le quali la più intuitiva, molto efficace e semplice da implementare è l'*Empty room*[26][27][28]. Come

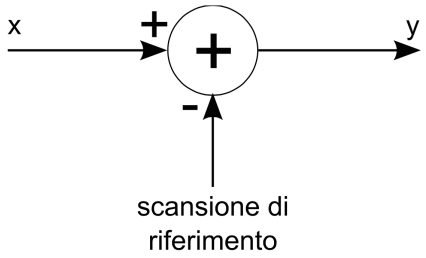


Figura E.1: Principio di funzionamento della tecnica *Empty Room*[28].

si nota dalla Figura E.1, il principio di funzionamento di questa elaborazione è molto semplice; infatti essa attua un confronto e quindi una sottrazione del segnale utile (ossia con presenza del *target*), con una forma d'onda rilevata in assenza di esso. Perciò questo algoritmo ha bisogno di una fase di *set-up* nella quale vengono salvate una serie di scansioni relative all'area di interesse, nel caso in cui non vi siano intrusi (da quale appunto deriva lo stesso nome *Empty Room*). Si intuisce che è fondamentale il fatto che l'area di interesse rimanga invariata poichè altrimenti l' "istantanea" salvata dell'*Empty Room* risulta obsoleta al confronto con la forma d'onda in presenza del *target*. In relazione a

ciò, le misure sperimentali eseguite e delle quali si discute nel Capitolo 7, sono state realizzate e salvate tutte nello stesso scenario cercando di mantenere tutto invariato e muovendo soltanto il *target* nell'area di osservazione.

Appendice F

Stima della posizione temporale del *target*

Al fine di riscontrare la presenza o meno del *target* nei segnali salvati, è possibile stimare una posizione temporale in cui ci si aspetta la presenza dell'intruso rispetto alla forma d'onda dell'*Empty Room*, semplicemente considerando posizione del target, relative distanze fra ricevitori e trasmettitori e la velocità della luce.
La scansione considerata in Figura 8.1 è quella relativa alla posizione di partenza del *target* (posizione zero), di conseguenza si può stimare a che distanza, in termini temporali, ci si aspetta di trovare una differenza dall'*Empty Room* che determina la presenza del target.
Dalla Figura F.1 si possono eseguire i calcoli e stimare dove dovrebbe essere il picco del *target* nelle scansioni allineate (Figura 8.1). Innanzitutto si calcola la distanza Trasmettitore-Ricevitore1 (Equazione F.1):

$$d_{TX-RX1} \simeq 6.86\ m \tag{F.1}$$

Poi si calcola la distanza fra il punto considerato e i nodi che devono rilevarlo (TX e RX1):

$$d_{PuntoZero} \simeq 3.26\ m + 8.43\ m = 11.69\ m \tag{F.2}$$

Si calcola ora la differenza fra le due distanze:

$$d_{Differenza} \simeq 11.69\ m - 6.86\ m = 4.83\ m \tag{F.3}$$

Giunti a questo punto, si calcola, stimando la velocità del segnale con

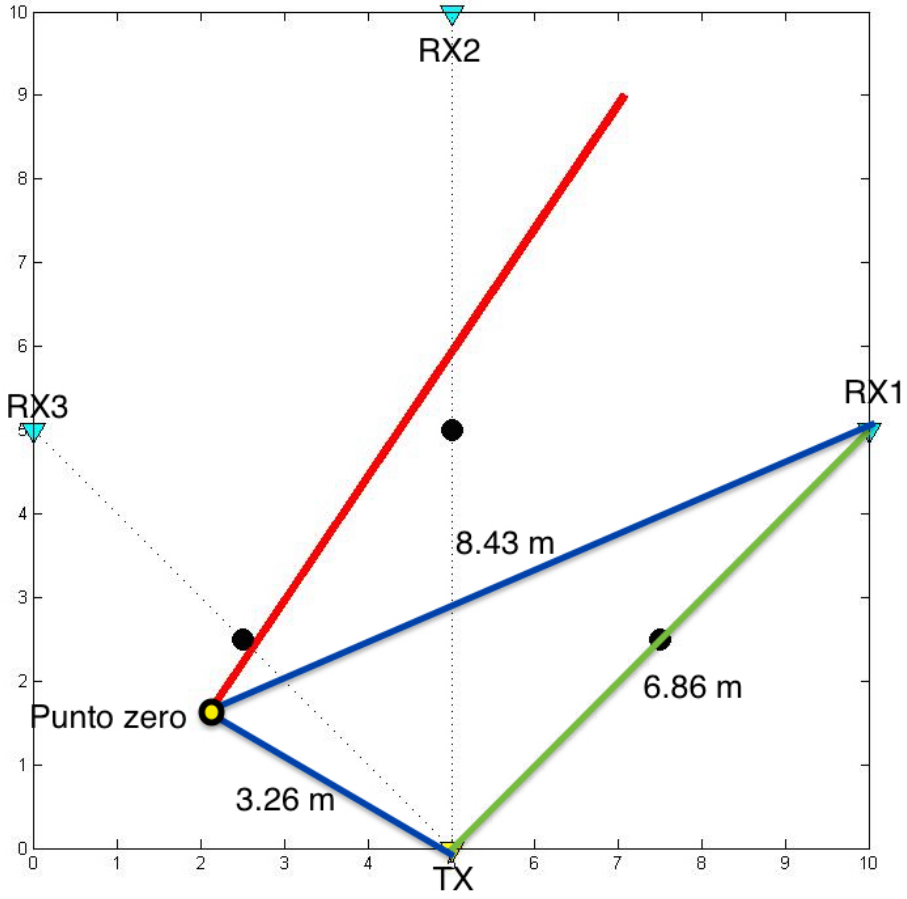


Figura F.1: Rappresentazione delle distanze da tenere in conto per l'elaborazione nel caso di Posizione Zero per il *target* e per il Ricevitore 1.

quella della luce, a che distanza temporale, rispetto al cammino diretto, dovrebbe trovarsi il target:

$$t_{Distanza} = \frac{4.83\ m}{3 \cdot 10^8 \frac{m}{s}} \simeq 16.1\ ns \tag{F.4}$$

Di conseguenza da questo risultato si nota che ci si aspetta il target a circa 16.1 ns di distanza temporale dal cammino diretto; il cammino diretto (come si può notare dalle impostazioni nel Capitolo 7 utilizzate in fase di misurazione) si trova allo zero temporale delle forme d'onda salvate, di conseguenza dopo 20 ns (poiché la forma d'onda salvata sta in un range $[-20 \div 97]$ ns). Di conseguenza ci si aspetta il target a circa $20 + 16.1\ ns = 36.1\ ns$ come è riscontrabile in Figura 8.1.

Bibliografia

[1] Xiaohui Huang, Yunming Ye, Haijun Zhang, *Extensions of Kmeans-Type Algorithms: A New Clustering Framework by Integrating Intracluster Compactness and Intercluster Separation*, IEEE Transaction on Neural Networks and Learning Systems, VOL. 25, NO. 8, August 2014.

[2] *http://www.mathworks.it/it/help/stats/hierarchical-clustering.html*

[3] Nicola Bagnolini, *Tracking di Target Multipli in Reti di Sensori Radar*, Tesi di Laurea Magistrale in Ingegneria Elettronica e Telecomunicazioni per lo Sviluppo Sostenibile, relatore Prof. Ing. Andrea Giorgetti, 2012-2013.

[4] Bita Sobhani, Enrico Paolini, Andrea Giorgetti, Matteo Mazzotti e Marco Chiani, *Target Tracking for UWB Multistatic Radar Sensor Networks*, IEEE Journal of Selected Topics in Signal Processing, VOL. 8, NO. 1, February 2014.

[5] Moe Z. Win, Robert A. Scholtz, *Ultra-Wide Bandwidth Time-Hopping Spread-Spectrum Impulse Radio for Wireless Multiple-Access Communications*, IEEE Transactions on Communications, Vol. 48, No. 4, April 2000.

[6] Federal Communications Commission, *Revision of part 15 of the commission's rules regarding ultra-wideband transmission systems*, first report and order (ET Docket 98-153) April 2002.

[7] Commission of the European Communities, *Commission decision of 21 february 2007 on allowing the use of the radio spectrum for equipment using ultra-wideband technology in a harmonised manner in the community (2007/131/EC)*, Official Journal of the European Union, February 2007.

[8] Enrico Paolini, Andrea Giorgetti, Marco Chiani, Riccardo Minutolo e Mauro Montanari, *Localization Capability of Cooperative*

Anti-Intruder Radar Systems, Hindawi Publishing Corporation, EURASIP Journal on Advances in Signal Processing, Vol. 2008.

[9] Luca Fontana, *Misure Sperimentali su Sustemi Radar Multistatici a Banda Ultralarga (UWB)*, Tesi di Laurea Specialistica in Ingegneria Elettronica e delle Telecomunicazioni, Relatore Prof. Ing. Marco Chiani, 2009-2010.

[10] *http://www.elettra2000.it/vdegliesposti/Dispense%20Propagazione%20e%20Pianificazione%20LM/20_uwb.pdf*, Corso di Propagazione e Pianificazione nei Sistemi d'Area LM, Prof. Ing. Vittorio Degli Esposti.

[11] *http://www.elettra2000.it/vdegliesposti/Dispense%20Tecniche%20LS/Sistemi_terrestri_localizzazione.pdf*, Corso di Tecniche Elettromagnetiche per la Localizzazione e il Controllo Ambientale LM, Prof. Ing. Gabriele Falciasecca.

[12] Xiaomin Chen e Sayfe Kiaei, *Monocycle Shapes for Ultra Wideband System*, Dept. of Electrical Engineering, Arizona State University.

[13] Adrian Popa, *An Optimization of Gaussian UWB Pulses*, 10th International Conference on Development and Application Systems, Suceava, Romania, May 2010.

[14] Eva H.K Yeung and John Mitchell, *Modelling of Ultra-wideband (UWB) Radio System*, Department of Electrical Engineering, University College London.

[15] Leonardo Calandrino e Marco Chiani, *Lezioni di comunicazioni elettriche*, Pitagora Editore, 2004.

[16] Marco Chiani, Andrea Giorgetti, Matteo Mazzotti, Riccardo Minutolo e Enrico Paolini, *Target Detection Metrics and Tracking for UWB Radar Sensor Networks*, ICUWB 2009, September 2009.

[17] Bita Sobhani, Matteo Mazzotti, Enrico Paolini, Andrea Giorgetti e Marco Chiani, *Multiple Target Detection and Localization in UWB Multistatic Radars*, IEEE International Conference on Ultrawideband, Paris, France, September 2014.

[18] Weixian Liu, Yilong Lu Member e Jeffrey S. Fu, *CFARData Fusion of Multistatic Radar System under Homogeneous and Nonhomogeneous Backgrounds*, Radar Lab Nanyang Technological University, Singapore.

[19] Ian Oppermann, Matti Hämäläinen, Jari Iinatti, *UWB: Theory and Applications*, Wiley, September 2004.

[20] *http://www.timedomain.com/p400.php*.

[21] *Monostatic Radar Module (MRM) User Guide*, www.timedomain.com, August 2012.

[22] *Ranging and Comunication Module (RCM) User Guide*, www.timedomain.com, August 2012.

[23] *Channel Analysis Tool (CAT) User Guide*, www.timedomain.com, August 2012.

[24] *Quick Start Guide PulsON Channel Analysis Tool (CAT)*, www.timedomain.com, August 2012.

[25] *Data Sheet PulsON 410*, www.timedomain.com, June 2012.

[26] Maicol Pollini, *Sviluppo di un applicativo per il rilevamento e la localizzazione basati su radar UWB*, Tesi di Laurea Magistrale in Ingegneria Elettronica e Telecomunicazioni, relatore Prof. Ing. Andrea Giorgetti, 2008/2009.

[27] Angelo Facondini, *Sistemi di Localizzazione UWB: Algoritmi di Rivelazione per Radar Multistatico*, Tesi di Laurea Specialistica in Ingegneria Elettronica e Telecomunicazioni, Relatore Prof. Ing. Marco Chiani, 2006/2007.

[28] Andrea Urbini, *Sistemi di Localizzazione UWB: Algoritmi di Rimozione del Clutter per Radar Multistatico*, Tesi di Laurea Specialistica in Ingegneria Elettronica e Telecomunicazioni, Relatore elatore Prof. Ing. Marco Chiani, 2006/2007.

Ringraziamenti

Senza dubbio, giunto a questo grande traguardo per me, dovrebbe essere facile elencare e ricordarmi tutte le persone che mi sono state accanto in questi ultimi anni e che soprattutto mi hanno sopportato, ma purtroppo lo sapete, non sono molto bravo a ricordarmi tutto e tutti, anzi.
Le prime persone che ringrazio e alle quali devo la proposta, lo svolgimento, l'aiuto nell'elaborazione di questa tesi, sono: il mio Relatore, il Prof. Enrico Paolini e i miei Correlatori, il Prof. Andrea Giorgetti e l'Ing. Matteo Mazzotti; senza i loro consigli, le loro proposte, gli incontri per discutere i risultati, la loro più totale disponibilità, non sarei mai stato in grado di portare avanti e concludere questo lavoro di tesi.
Vorrei ringraziare anche: il Prof. Marco Chiani, il quale mi ha permesso di utilizzare i nodi Time Domain PulsOn 410, senza i quali non sarei mai riuscito a giungere alle conclusioni di questo elaborato; il Prof. Lorenzo Marconi e il Prof. Roberto Naldi per la completa disponibilità nell'utilizzo del Laboratorio Droni per le mie misure sperimentali.
Per continuare, ringrazio i miei genitori, le mie sorelle e i miei parenti, senza il loro supporto e la loro pazienza, non ce l'avrei mai fatta a finire anche la Magistrale.
Un grazie molto sentito a Mario, Gabrio e Sandra, sempre presenti, sempre disponibili, sempre pronti a darmi una mano, nonostante le difficoltà, nonostante la distanza, nonostante tutto. Grazie davvero.
Grazie a Malto, Cava, Mambo, Giacomo, Pole, Pietro, che mi hanno sopportato durante questi anni di studio e serate assieme.
Grazie Matteo, Josè, Claudia, Paolo, Max, che hanno condiviso con me questi ultimi due anni di alti e bassi con gli esami e mi hanno aiutato non poco nel raggiungere questo obiettivo.
Grazie a Puc, Zol, Mic, Tonno, Fabio e Giorg, che purtroppo vedo troppo poco rispetto a quanto vorrei. Nonostante ciò "si potrebbe fare un libro su tutto ciò che abbiamo combinato ma è giusto tenere questi ricordi per noi!"; Puc ho voluto e dovuto citarti perché questa frase ci rispecchia troppo.
Grazie a Vale per gli anni di studio assieme e soprattutto per le foto di

questa tesi.
Grazie a Giorgia, Alice, Francesca ed Elena che hanno condiviso con me le ore passate in biblioteca, le infinite pause caffè e soprattutto i miei strippi.
Grazie a Barbara, Monica, Chiara e Cesare, che, nonostante la lontananza, nonostante le mille difficoltà nell'incontrarci, nonostante la Monica (ti voglio bene Tappo, ma lo sai che dovevo scriverti qualcosa), ogni volta che riusciamo a vederci è come ritornare indietro nel tempo.
Ultimi ma non meno importanti grazie a Felix, Cicci, Mirko, Valbo, Boc, Bobo, Miki, Frency, Riki e a tutti i miei amici di Forlimpopoli che ovviamente ho dimenticato di menzionare.
Insomma grazie a tutti, soprattutto a chi ho dimenticato!