Agenzia nazionale per le nuove tecnologie,
l'energia e lo sviluppo economico sostenibile

MINISTERO DELLO SVILUPPO ECONOMICO

Ricerca di Sistema elettrico

Studio e sviluppo di algoritmi per l'ausilio al pilota e l'elaborazione dei dati sensoriali di bordo

Silvello Betti, Stefano Chiesa

Dipartimento di
Ingegneria Elettronica

Report RdS/PAR2017/065

STUDIO E SVILUPPO DI ALGORITMI PER L'AUSILIO AL PILOTA E L'ELABORAZIONE DEI DATI DEI DATI SENSORIALI DI BORDO

Silvello Betti, Stefano Chiesa (Dipartimento Ingegneria Elettronica, Università di Roma "Tor Vergata")

Settembre 2018

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Annuale di Realizzazione 2017
Area: Efficienza energetica e risparmio di energia negli usi finali elettrici e interazione con altri vettori energetici
Progetto: D.6 Sviluppo di un modello integrato di smart district urbano
Obiettivo: d - Sicurezza infrastrutture critiche e monitoraggio aereo dello Smart District - Sotto-obiettivo d.2: Monitoraggio aereo dello smart district
Responsabile del Progetto: Claudia Meloni, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione "Studio e sviluppo di algoritmi per l'ausilio al pilota e l'elaborazione dei dati sensoriali di bordo"
Responsabile scientifico ENEA: Dr. Sergio Taraglio
Responsabile scientifico Università di Roma "Tor Vergata": Prof. Silvello Betti

Indice

Sommario

Questo documento descrive le attività svolte ed i risultati conseguiti dal Dipartimento di Ingegneria Elettronica dell'Università di Roma "Tor Vergata", nell'ambito dell'Accordo di Collaborazione con ENEA volto allo studio e sviluppo di algoritmi per l'ausilio al pilota e l'elaborazione dei dati sensoriali di bordo.

Gli ambiti di attività si sono articolati in aspetti teorici e di specifica, in aspetti di implementazione e programmazione e in test funzionali ed operativi.

Gli aspetti teorici e di specifica sono stati indirizzati allo studio di un algoritmo di separazione tra aeromobili sviluppato nel Laboratorio di Robotica dell'ENEA per studiarne la portabilità in un diverso ambito, quello dell'ausilio ad un pilota di RPAS (Remotely Piloted Aerial System, meglio noto come drone). Una volta chiarita la parte teorica ci si è rivolti alla parte implementativa dove sono stati realizzati i moduli software per rendere operativo l'algoritmo all'interno del simulatore robotico realizzato nel corso del PAR 2016, disegnato per l'addestramento dei piloti di drone. L'implementazione è stata condotta nell'ambito del *framework* ROS (Robot Operating System) entro cui è già operante il simulatore.

Ultimata l'implementazione del software si è passati ad una fase di test durante la quale si sono verificate le rispondenze con le specifiche. I test sono stati condotti in collaborazione con il Laboratorio di Robotica dell'ENEA e hanno previsto una serie di voli simulati di più droni contemporaneamente, fino ad un massimo di tre, intorno ad un edificio dello Smart Village del Centro Ricerche della Casaccia. La configurazione di test prevedeva il volo pilotato di due dei droni in modo completamente privo di vincoli, mentre il pilota del terzo drone riceveva suggerimenti dall'algoritmo allo scopo di non mettersi in situazioni di pericolo con l'edificio o i droni.

Un secondo filone di attività si è diretto verso la calibrazione ottica di telecamere termiche. La necessità di una tale attività deriva dal fatto che una parte delle attività del PAR 2017 sono volte alla realizzazione di modelli 3D sia nella banda del visibile che in quella dell'infrarosso termico. Per ottenere modelli più accurati può essere utile conoscere i parametri ottici delle telecamere utilizzate.

E' quindi stata realizzata una procedura per la modellazione dei parametri di una telecamera termica, creando pattern 'termici' e applicando alcune elaborazioni alle immagini riprese per un efficace calcolo dei parametri ottici.

1 Introduzione

Il presente Report descrive le attività svolte dal Dipartimento di Ingegneria Elettronica dell'Università di Roma "Tor Vergata", nell'ambito dell'Accordo di Collaborazione con ENEA volto allo studio e sviluppo di algoritmi per l'ausilio al pilota e l'elaborazione dei dati sensoriali di bordo di un drone multi elica, Figura 1.

Figura 1. Il quadricottero

Il drone è utilizzato nell'ambito della Ricerca di Sistema Elettrico nel Piano Annuale di Realizzazione 2017, nel progetto "D.6 - Sviluppo di un modello integrato di Smart District urbano", in particolare nel sotto progetto "d - Sicurezza infrastrutture critiche e monitoraggio aereo dello Smart District". Il drone multi elica ha infatti lo scopo di monitorare uno Smart District, ovvero un certo numero di edifici già sensorizzati all'interno allo scopo di misurarne i consumi energetici, anche dall'esterno sia in termini di dispersione energetica che di qualità dell'aria. Quest'ultima, infatti, è fortemente influenzata dalle emissioni di impianti energetici quali ad esempio le caldaie per il riscaldamento o la produzione di acqua sanitaria e quindi rappresentano un'utile integrazione secondaria ai dati energetici primari.

A tale scopo il drone è equipaggiato con una termo camera nella banda infrarossa per il monitoraggio delle dispersioni dagli edifici e di un insieme di sensori per la misurazione di inquinanti legati alle attività energetiche dell'uomo (autotrazione, riscaldamento, etc.), allo scopo di monitorare la qualità dell'aria. Per ulteriori dettagli sulle caratteristiche del drone si rimanda a al rapporto [1].

Il drone è interfacciato ad un Sistema di Supporto alle Decisioni (Decision Support System, DSS) con finalità operative, basato su una Infrastruttura di Dati Territoriali (SDI, Spatial Data Infrastructure) in modo da poter presentare i dati raccolti dal drone in modo georeferenziato.

Le attività di ricerca e sviluppo si sono indirizzate verso due distinti filoni: da una parte lo sviluppo e l'implementazione di algoritmi per l'ausilio al pilota e dall'altra alla calibrazione della termocamera di bordo per migliorare i modelli 3D degli edifici illustrati nel rapporto [2].

2 Descrizione delle attività svolte e risultati

Il presente documento descrive le attività svolte, classificabili in tre linee di lavoro:

- studio e sviluppo di un algoritmo per l'ausilio al pilota;
- estensione del simulatore realizzato nel corso del PAR 2016 al fine di effettuare i test relativi al funzionamento dell'algoritmo;
- calibrazione di telecamere sensibili nell'infrarosso termico.

2.1 Studio e sviluppo di un algoritmo per l'ausilio al pilota

2.1.1 Introduzione

Il sistema per il monitoraggio energetico e chimico sviluppato nell'abito del PAR 2016 è, come detto, basato su di una piattaforma volante: un drone. Un aspetto assolutamente rilevante dell'utilizzo di velivoli pilotati remotamente è senz'altro quello della sicurezza. I RPAS (Remotely Piloted Aerial System) possono rappresentare un fattore di rischio sia nella gestione del traffico aereo che nella sicurezza di persone e cose a terra.

Questa linea di lavoro è stata indirizzata quindi allo studio e allo sviluppo di un algoritmo di ausilio al pilota, con lo scopo di ridurre il rischio. Fermo restando che la decisione finale spetta al pilota, questo algoritmo è in grado di suggerire un percorso che sia il più sicuro possibile, tenuto conto delle condizioni al contorno. Queste condizioni sono rappresentate dalla eventuale presenza di altri droni e/o velivoli nell'area e dalla presenza degli edifici da monitorare.

La base da cui partire è stata rappresentata da un algoritmo sviluppato dal Laboratorio di Robotica dell'ENEA (DTE-SEN-IDRA) [3]. Esso è stato partner di una serie di progetti europei e regionali (MACRO, EUROSTARS ARCA, SARA, OLGA) volti all'ideazione, sperimentazione e perfezionamento di un algoritmo che permettesse ad un velivolo *unmanned* autonomo un volo sicuro per sé e per il traffico limitrofo. Questa serie di progetti è stata sviluppata in collaborazione con la PMI italiana Deep Blue s.r.l.. Tema comune di questi progetti è la realizzazione di un sistema sia hardware che software che possa essere alloggiato a bordo di un UAV (Unmanned Aerial Vehicle) e che sia in grado di permettere al pilota del velivolo, sia esso automatico o umano, di evitare eventi di mancata separazione tra aeromobili. Ovvero di segnalare al pilota la manovra evasiva che permetta al veicolo di non avvicinarsi mai ad altro aeromobile entro un dato raggio di sicurezza (per i velivoli di linea cinque miglia nautiche, circa 8 km) prescritto dalle regole del controllo del traffico aereo.

Nel caso degli RPAS l'algoritmo segnala al pilota la migliore traiettoria, ma la decisione ultima rimane comunque in capo al pilota.

L'algoritmo è basato sulla Teoria dei Giochi che analizza matematicamente situazioni di conflitto tra agenti diversi e ne ricerca soluzioni competitive e/o cooperative tramite modelli matematici [4]. Studia cioè le decisioni individuali in situazioni dove ci siano interazioni tra più soggetti interdipendenti, ovvero dove le decisioni di uno possano influire sui risultati conseguibili da un altro, finalizzate al massimo guadagno del soggetto considerato. Nata per analizzare le strategie per vincere nei giochi, si è poi estesa ai campi più disparati, in modo particolare in economia.

L'algoritmo in realtà sfrutta un'estensione di questa teoria denominata Satisficing Game Theory (SGT) [5]. Il neologismo *satisficing* è l'unione di *satisfy* (soddisfare) e *suffice* (essere sufficiente). Tale estensione prende in considerazione, mediante l'uso di opportuni strumenti matematici, i pro e i contro di una decisione di un singolo agente (nel caso in esame il drone), tenendo però anche in conto le possibili decisioni che possono essere prese dagli altri agenti che operano nel contesto considerato, secondo una scala condivisa di priorità. La decisione finale di ogni agente sarà dunque il frutto di un bilanciamento tra i propri desideri e le esigenze degli altri agenti con cui si confronta, dando così luogo all'instaurazione di un comportamento collettivo di tipo anche altruistico. Questa strategia quindi fornisce una soluzione adeguata alle esigenze di tutti gli agenti in gioco, ma, conseguentemente, non necessariamente quella ottimale per tutti gli agenti.

L'algoritmo è basato sulle informazioni di posizione, velocità e di direzione di marcia degli agenti presenti nella zona. A questi dati viene qui dato, per comodità, il nome di dati ADSB (Automatic Dependent Surveillance – Broadcast, [6]), che sono quelli inviati in modalità broadcast via radio da tutti gli aeromobili commerciali con i propri dati di posizione, velocità, direzione, etc.. Gli agenti considerati dall'algoritmo possono essere di diverso tipo: altri velivoli, edifici, eventi meteo, terreno. Durante la presente annualità di lavoro sono state implementate le prime tre classi di agenti ovvero altri velivoli, edifici e meteo, realizzando però il software in modo da poter essere facilmente scalato per poter tenere in conto anche le altre classi.

L'algoritmo è stato implementato come una libreria dinamica. Questa libreria può essere richiamata da un programma esterno per calcolare una possibile traiettoria che consenta a un velivolo di evitare altri aereomobili, situazioni meteo critiche o edifici. Come detto, un possibile utilizzo di questa libreria è quello di suggerire al pilota una rotta da seguire per ridurre le possibilità di conflitto.

Il codice è stato scritto in C++ all'interno dell'ambiente di sviluppo Eclipse, utilizzando come sistema operativo linux Ubuntu 18.04 LTS ed è contenuto nella cartella compressa EneaAlgorithm.zip.

2.1.2 Architettura generale del sistema

In Figura 2 è riportata una porzione dell'architettura generale del sistema complessivo, ovvero come l'algoritmo si rapporta con il simulatore.

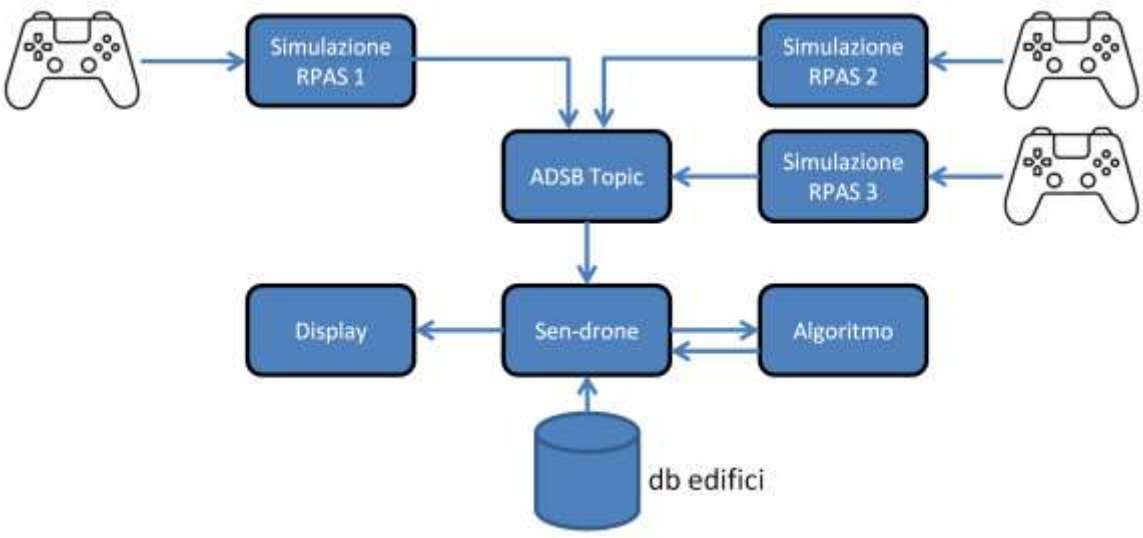

Figura 2. L'architettura generale dell'uso dell'algoritmo nel simulatore

In Figura 2 sono presenti tre RPAS simulati, di questi è il numero 1 che fruisce dell'algoritmo. Il modulo *sen_drone_node* costituisce un adattatore che consente di utilizzare l'algoritmo all'interno del framework di simulazione. Esso riceve in ingresso i dati ADSB dei velivoli presenti dall' ADSB_Topic e attinge ad un

data base degli edifici della zona. Utilizzando questi dati invoca l'algoritmo per calcolare una traiettoria sicura per il drone 1 e mostrarla sul display quale suggerimento al pilota dello RPAS 1.

Tutto il sistema di simulazione è stato implementato nell'ambito di ROS (Robot Operating System) [7], riutilizzando e adattando nodi software preesistenti e realizzandone di nuovi.

2.1.3 Dettaglio dell'implementazione dell'algoritmo

Come detto l'algoritmo è stato incapsulato in una libreria richiamabile a *runtime*. Essa espone una interfaccia descritta nel file di *header* EneaAlgorithm.h.

La classe *EneaAlgorithm* costituisce il punto di accesso alla libreria e può essere istanziata invocando il costruttore *EneaAlgorithm::EneaAlgorithm(...)* con una serie di parametri che permettono di controllare l'esecuzione dell'algoritmo:

- n_step Numero di step nel futuro per i quali l'algoritmo deve effettuare il calcolo della traiettoria,
- delta_t Intervallo di tempo tra gli n_step nel futuro eseguiti dall'algoritmo (in secondi),
- h_sep Separazione orizzontale (in miglia nautiche),
- v_sep Separazione verticale (in piedi).

Tramite i metodi *ADSB_in()*, *weather_in()*, *building_in()* è possibile impostare i dati relativi a altri velivoli, situazioni meteo avverse e edifici dai quali si intende mantenere un certo livello di separazione. *ADSB_self()* consente invece di passare alla classe le informazioni relative al velivolo per il quale si intenda calcolare la traiettoria.

Invocando il metodo *compute_route()* su una istanza della classe è possibile richiedere il calcolo della traiettoria date le condizioni correnti al contorno.

Il calcolo della traiettoria viene effettuato nel seguente modo.

Si considerino due tempi: il *wall time* ovvero il tempo reale della simulazione indicato con τ ed il tempo di algoritmo per il calcolo della traiettoria indicato con t. Si assuma che sia τ_0 il tempo corrente, congeliamo l'evoluzione temporale della simulazione a questo punto. Il drone per il quale si calcola la traiettoria avrà una data posizione ($x(\tau_0)$), velocità e direzione. Sulla base dei dati degli edifici e degli altri droni l'algoritmo sarà in grado di calcolare puntualmente la virata che comporta i maggior benefici (avvicinamento al prossimo waypoint, ovvero al punto di arrivo) e i minori costi (allontanamento da situazioni di conflitto con altri droni o edifici, ovvero passaggi troppo ravvicinati).

Con questa nuova direzione è possibile calcolare dove sarà il drone considerato ad un tempo $t_1>\tau_0$ nel futuro ($x_d(t_1)$) con $t_1=\tau_0+delta\_t$ indicando con $x_d(t_1)$ la posizione calcolata del drone all'istante t_1. Allo stesso modo, mantenendo inalterati i dati degli altri attori (droni ed edifici) è possibile calcolare dove essi saranno al tempo t_1; ovviamente a muoversi saranno solamente i droni. A questo punto si avrà la configurazione ipotetica di tutti gli attori al tempo t_1, ipotetica in quanto si è assunto che a manovrare sia stato solo il drone in considerazione, mentre gli altri attori hanno agito come se non avessero fatto alcuna manovra e, soprattutto, in realtà la simulazione è sempre congelata al tempo τ_0.

Con questi dati sarà ora possibile calcolare una nuova virata per il drone considerato nella posizione $x_d(t_1)$, che renda la rotta più conveniente e che porterà il drone nella posizione $x_d(t_2)$ con $t_2=t_1+delta\_t$.

Si ripeterà ora il ragionamento per il drone portato in posizione $x_d(t_2)$ e così via, fino a calcolare una traiettoria $[x(\tau_0), x_d(t_1), x_d(t_2),\ldots, x_d(t_{n\_step})]$ con $t_{n\_step} = n\_step*delta\_t$.

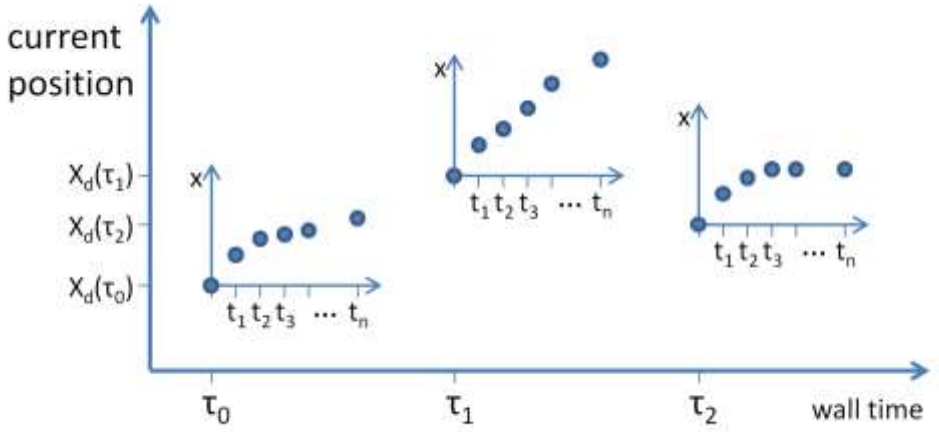

Figura 3. I due diversi riferimenti temporali: wall time e tempo di algoritmo

Questa sarà la traiettoria più conveniente al tempo τ_0 e la si potrà mostrare al pilota quale suggerimento. Terminato il calcolo della rotta suggerita, si potrà fare evolvere tutto il sistema fino al prossimo passo temporale τ_1 (wall time), congelare nuovamente il sistema e ripetere la sequenza di calcoli sopra esposta.

Quindi per ogni step temporale del *wall time* (τ) si potrà avere una traiettoria suggerita che potrà cambiare ad ogni step successivo, al variare della configurazione degli agenti presenti nell'area.

In Figura 3 è mostrato in forma grafica quanto esposto: il sistema di riferimento più grande rappresenta lo scorrere del tempo nella simulazione (*wall time*) ed i sistemi di riferimento più piccoli rappresentano il calcolo di una traiettoria ciascuno, rappresentata da posizioni in funzione dei tempi di algoritmo $t_1,\ldots,t_{n\_step}$. E' ora più chiaro il significato dei primi due parametri della classe EneaAlgorithm: *n_step* è il numero di step temporali con cui calcolare la traiettoria (tempo di algoritmo), ovvero per quanti step simulare l'evoluzione temporale della configurazione di agenti ad ogni passo temporale reale τ_k; *delta_t* definisce l'intervallo che intercorre tra due generici istanti temporali t_i e t_j con *j=i+1* relativi al tempo di algoritmo nei quali viene effettuata la scelta della manovra più conveniente da effettuare. Quindi *n_step* x *delta_t* è il numero di secondi di volo che rappresenta la traiettoria suggerita.

In definitiva ad ogni step temporale del *wall time* della simulazione, il sistema mostrerà su di un display una traiettoria suggerita di *n_step* passi temporali da *delta_t* secondi. Al passo temporale successivo la traiettoria sarà ricalcolata sulla base dei dati aggiornati. L'intervallo temporale tra un calcolo di traiettoria ed il successivo è un parametro del software il cui valore è di 3 secondi. Si tenga presente che il segnale ADS-B trasmesso dagli aerei di linea ha una frequenza di 1 Hz. Nel presente caso le velocità in gioco sono molto minori ed un intervallo di questa entità è ragionevole.

In Figura 4 è mostrato il diagramma di flusso relativo all'esecuzione dell'algoritmo. Come descritto in precedenza, l'algoritmo è un evoluzione del lavoro presentato in [3] e adattato ai sistemi SAPR. A differenza dell'algoritmo originale, che tiene in considerazione esclusivamente altri velivoli, la versione qui descritta tiene conto anche della presenza di edifici e del meteo. Nella versione originale i velivoli sono ordinati sulla base di un meccanismo di priorità basata su vari fattori (es. tempo di volo, quantità di carburante, etc.). L'algoritmo cerca di evitare solo i velivoli con priorità maggiore del velivolo per cui si calcola la traiettoria, altri quelli a più bassa priorità vengono ignorati. Nel caso di sistemi SAPR si è scelto di adottare la soluzione più conservativa possibile, attribuendo a tutte le altre entità presenti una priorità maggiore di quella del drone.

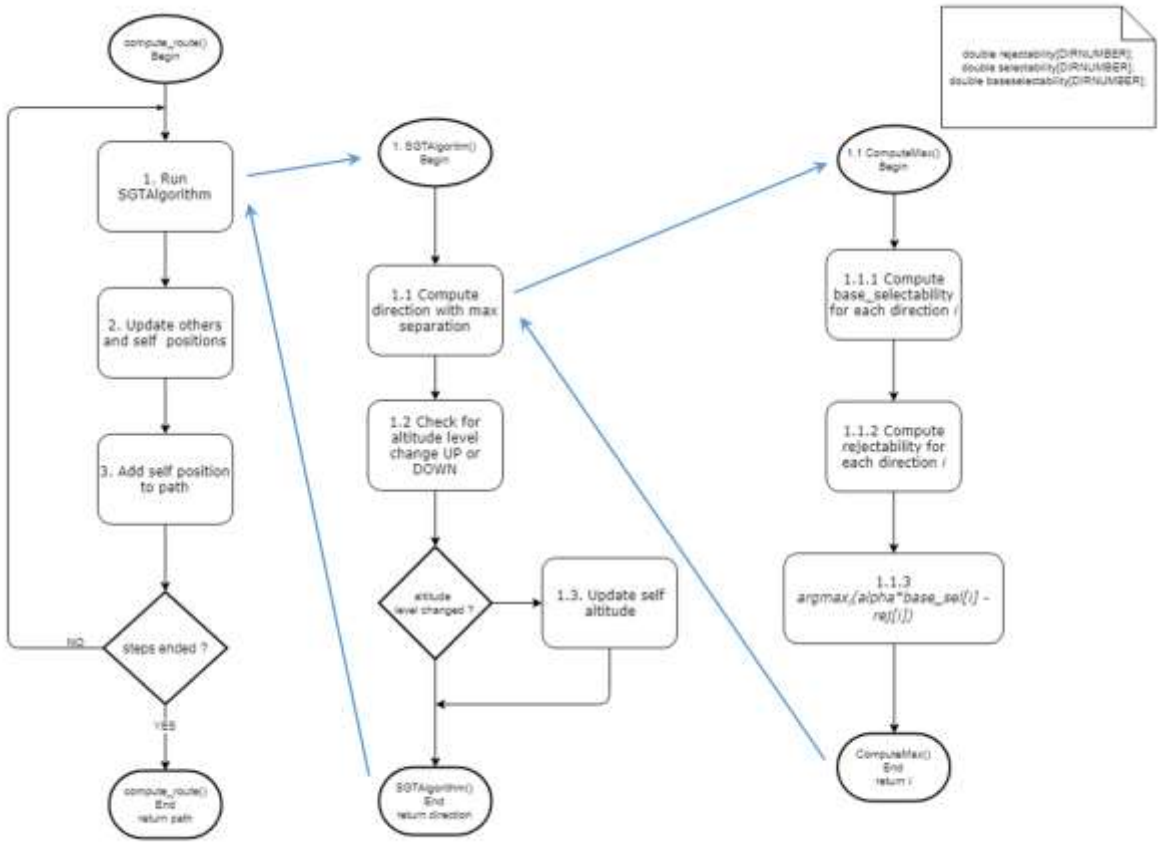

Figura 4. Diagramma di flusso relativo all'esecuzione dell'algoritmo

L'algoritmo implementa un flusso logico di tre passi mostrati nella colonna più a sinistra della Figura 4 (funzione *compute_route()*):

1. si calcola sulla base della Teoria dei Giochi la direzione di volo migliore (passo 1);
2. si aggiornano le posizioni di tutti gli attori presenti (drone, altri velivoli, etc.) (passo 2);
3. si aggiunge la posizione corrente alla traiettoria (passo 3).

Si itera su questa sequenza fino a quando si siano calcolati gli *n_step* punti della traiettoria da suggerire.

Passo 1: viene richiamata la funzione *SGTAlgorithm()* la quale restituisce la direzione migliore da scegliere dato un insieme di direzioni possibili costituito dall'insieme $D=\{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$ con $d_1, d_2, \dots, d_7$ angolo di virata possibili rispetto alla rotta attuale, in radianti.

Al fine di scegliere la direzione migliore, la funzione *SGTAlgorithm()* richiama a sua volta la funzione *ComputeMax()* (passo 1.1 nella colonna centrale della Figura 4) la quale per ogni direzione calcola due funzioni di utilità: la *base_selectability* e la *rejectability* (rispettivamente nei passi 1.1.1 e 1.1.2 nella colonna di destra della Figura 4). Queste funzioni rappresentano i benefici (*selectability*) ed i rischi (*rejectability*) derivanti dalla scelta di una certa rotta per il drone. L'obiettivo è quello di scegliere una traiettoria costituita da punti in ognuno dei quali la scelta della traiettoria tenda ad ottenere una alta *selectability* ed una bassa *rejectability* come descritto con la formula (passo 1.1.3):

$$d' = argmax_i(\alpha * selectability[i] - rejectability[i]\ ),\ con\ 0<i<N_d$$

La *selectability* relativa a una direzione che il drone può seguire è un indicatore di quanto tale traiettoria sia efficace per raggiungere il target. In [3] il calcolo della *selectability* si basa si due componenti: la

base_selectabily e la *parent_selectability*. La prima rappresenta la preferenza del drone stesso; la *parent_selectability* rappresenta invece le preferenze degli altri velivoli presenti nell'area con una priorità più alta del drone. La *selectability* finale viene calcolata come combinazione lineare di queste due grandezze. A differenza di quanto descritto in [3], nella presente implementazione la *selectability* coincide con la *base_selectability*.

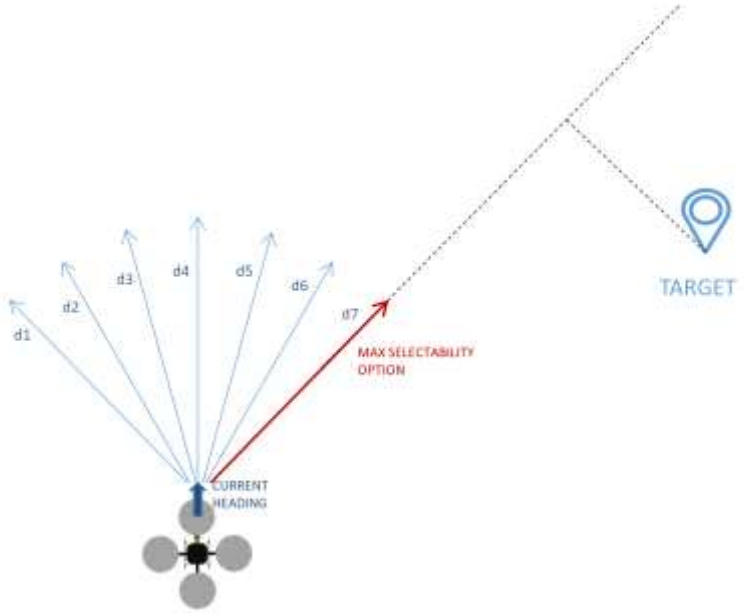

Figura 5. La scelta di rotta che massimizza la selectability

In Figura 5 sono mostrate le sette possibili scelte direzionali per il drone e quella che massimizza la *selectability* (d7, in rosso), in quanto porta il drone in una direzione più vicina al proprio target, ad esempio il prossimo *waypoint* della rotta.

La *rejectability*, al contrario, rappresenta il rischio che si corre nel seguire una certa rotta. Questo rischio può essere dovuto a possibili violazioni della separazione con la rotta di altri velivoli, con edifici o con condizioni meteo avverse. Per ognuno dei conflitti individuati l'algoritmo aggiunge un peso alla *rejectability* totale di una data scelta di manovra. Tale peso è funzione di alcuni parametri come la severità del conflitto (ad. esempio quanto vicino il drone passerà a un'altra entità) oppure la distanza nel tempo (o nello spazio, legate dalla velocità del velivolo) a cui ciò avverrà, in modo che conflitti che avvengano prima abbiano un peso maggiore.

Calcolo della rejectability nel caso di un conflitto un altro velivolo

Per ogni direzione di virata che il drone può scegliere l'algoritmo calcola il *closest point of approach* (di seguito cpa) con tutti gli altri velivoli all'interno di un certo raggio. Il cpa indica il punto in cui due velivoli saranno più vicini, nella proiezione futura della loro rotta come mostrato in Figura 6. Le rotte sono nello spazio tridimensionale e non si devono necessariamente intersecare affinché ci sia una violazione della separazione come si può vedere nel caso 3D mostrato nella parte inferiore della Figura 6. Si definiscono *horizontal separation* e *vertical separation* le distanze minime a cui i velivoli si debbano trovare sui piani orizzontale e verticale affinché non ci sia un conflitto. In caso contrario la severità del conflitto è funzione della distanza del drone 1 dal cpa, ovvero quanto sia vicino alla situazione di conflitto, e dalla effettiva misura della distanza tra i velivoli al cpa, ovvero quanto i due velivoli siano effettivamente in conflitto.

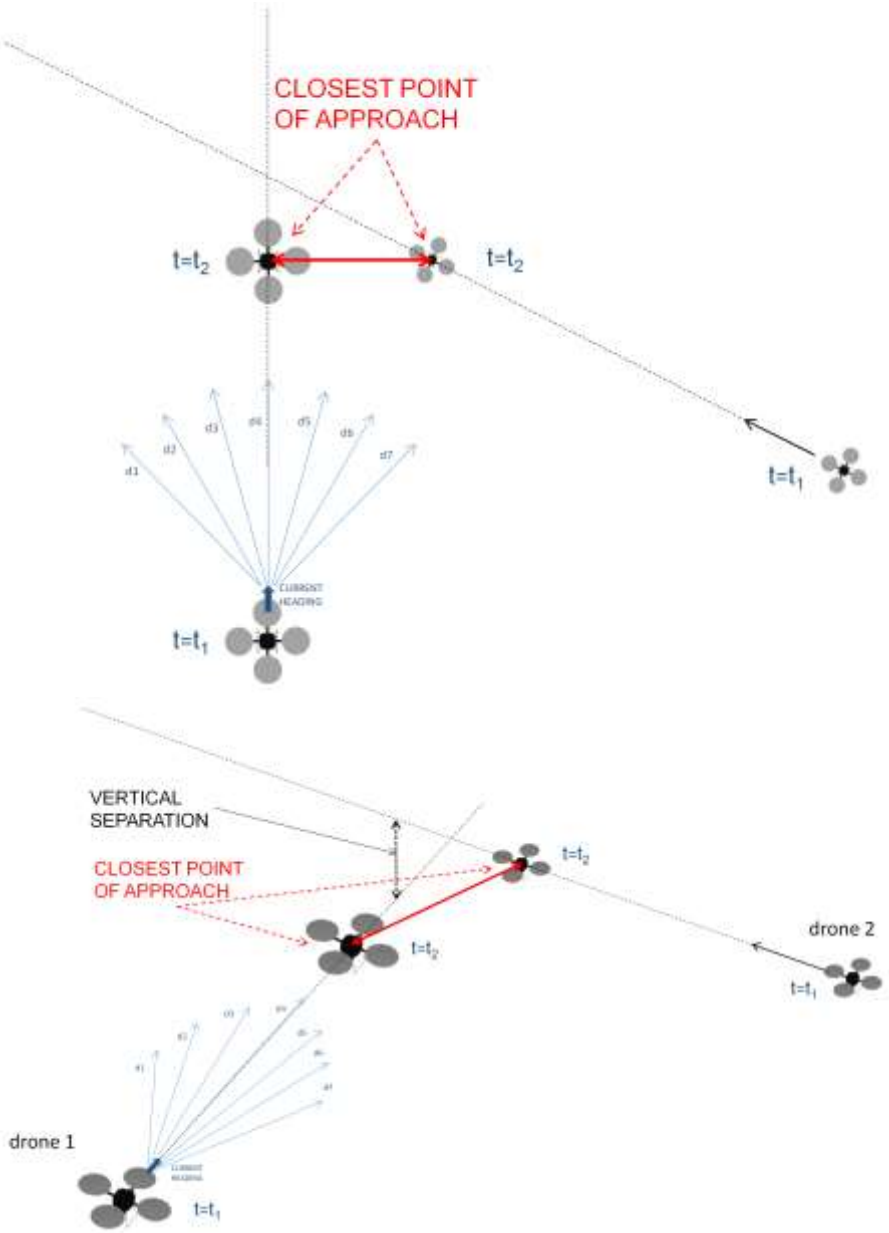

Figura 6. Il closest point of approach visto in pianta (in alto) e in 3D (in basso) nel caso di due velivoli

Calcolo della rejectability nel caso di una zona di maltempo

Il calcolo dei pesi da aggiungere alla *rejectability* di una specifica rotta direzione di volo a causa di condizioni di maltempo, avviene in maniera simile a quanto descritto in precedenza per i velivoli, si veda la Figura 7. La zona di maltempo viene rappresentata da un cilindro posizionato a una determinata latitudine e longitudine con un determinato raggio (in miglia nautiche) ed una quota (in piedi). L'algoritmo considera il centro della perturbazione come se fosse un velivolo fermo per cui il cpa, corrisponderà al punto della traiettoria proiettata del drone che si troverà alla distanza minima dal centro della perturbazione. La *horizontal_separation* in questo caso coincide con il raggio del cilindro.

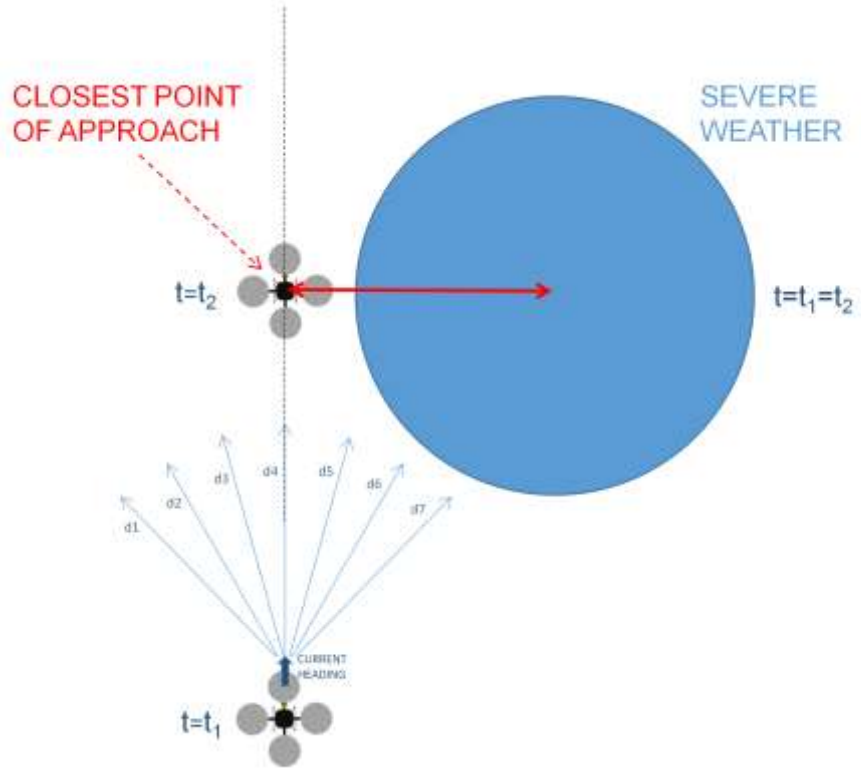

Figura 7. Il cpa nel caso del maltempo

Calcolo della rejectability nel caso di edifici

Gli edifici vengono rappresentati da una sequenza di vertici che ne definiscono il perimetro. Ogni vertice è descritto da latitudine longitudine e quota (in piedi). Per ogni possibile rotta direzione di volo l'algoritmo verifica se questa vada ad intersecare una porzione dell'edificio e calcola una lista di eventuali punti di collisione come mostrato in Figura 8. Oltre ai punti di intersezione del drone l'algoritmo considera anche i vertici del perimetro come punti di collisione.

Ognuno di questi punti viene considerato dall'algoritmo come un velivolo fermo in quella posizione e di conseguenza aumenta la *rejectability* delle rotte che passano in prossimità di essi.

Ritorniamo ora alla sequenza dei passi dell'algoritmo.

Passo 2: viene calcolata la posizione futura dei velivoli assumendo che la loro rotta e la loro velocità restino immutate ad eccezione del velivolo per cui viene fatto girare l'algoritmo il quale si assume che modifichi la sua rotta secondo quanto indicato dall'algoritmo stesso al passo 1.

Passo 3: la posizione calcolata del drone $x_d(t_i)$ con *i* che indica il passo di iterazione corrente (da *0* a *n_step*) viene aggiunta alla traiettoria da suggerire al pilota.

L'algoritmo itera i passi 1,2 e 3 fino a quando non avrà eseguito tutti gli *n_step*. La traiettoria così calcolata viene restituita dalla funzione *compute_route()* e il software che utilizza la libreria, nel caso in questione il nodo *sen_drone_node*, può utilizzarla per visualizzarla su un display di ausilio al pilota.

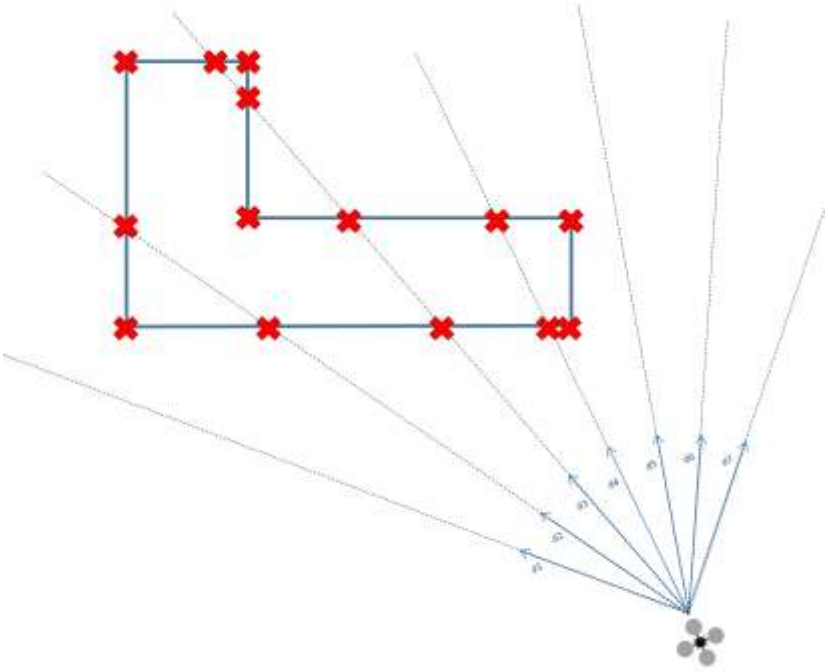

Figura 8. Il cpa nel caso di un edificio

Nel simulatore sviluppato la funzione *compute route()* viene richiamata ogni circa 3 secondi.

Per dare un'idea più chiara dell'output dell'algoritmo in Figura 9 è mostrata l'evoluzione temporale della traiettoria suggerita. Le tre immagini mostrano tre traiettorie suggerite dall'algoritmo mentre il drone viene pilotato. In dettaglio sono le traiettorie suggerite agli istanti $\tau = 0\ s$, $\tau = 6\ s$ e $\tau = 18\ s$, in modo da evitare un edificio che si frappone tra il drone (in basso a destra) ed il target di volo, rappresentato dalla sfera verde. L'algoritmo è stato invocato con *n_step = 30* e *delta_t = 3 s*.

In Figura 9 oltre alla traiettoria suggerita è possibile vedere la sagoma dell'edificio in grigio correttamente georeferenziato alla mappa di Mapbox [8] e la sagoma di ingombro dello stesso, rappresentata con segmenti verdi e rossi, indicanti l'area di rispetto entro cui è opportuno che il drone non voli.

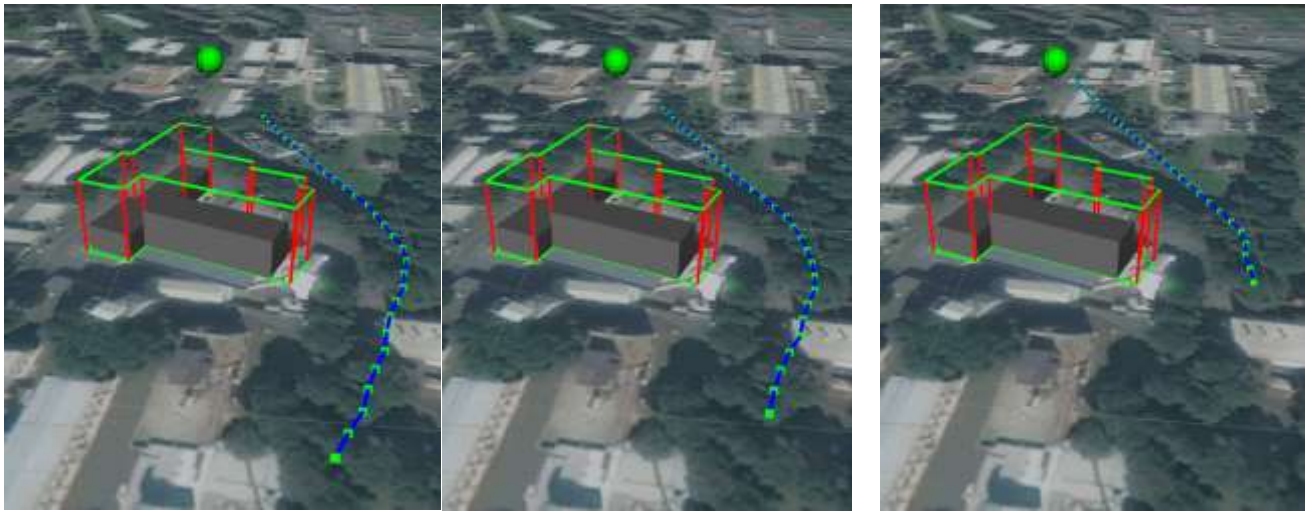

Figura 9. L'evoluzione temporale della traiettoria suggerita

2.1.4 Dettaglio dell'implementazione del simulatore

In Figura 10 è mostrata l'architettura più di dettaglio del sistema. I blocchi in colore verde costituiscono il cuore del sistema in quanto consentono di testarne le funzionalità più importanti e sono state implementate nel corso della corrente annualità. I blocchi in colore bianco sono funzionalità aggiuntive che sono state previste e tenute in conto nella scrittura del codice, ma non ancora implementate. Un caratteristica importante di questa architettura è la sua estensibilità che deriva dalla modularità intrinseca del framework ROS.

E' infatti possibile utilizzare lo stesso simulatore per valutare diversi algoritmi, a condizione che si abbia un adattatore che implementi l'interfaccia con il simulatore.

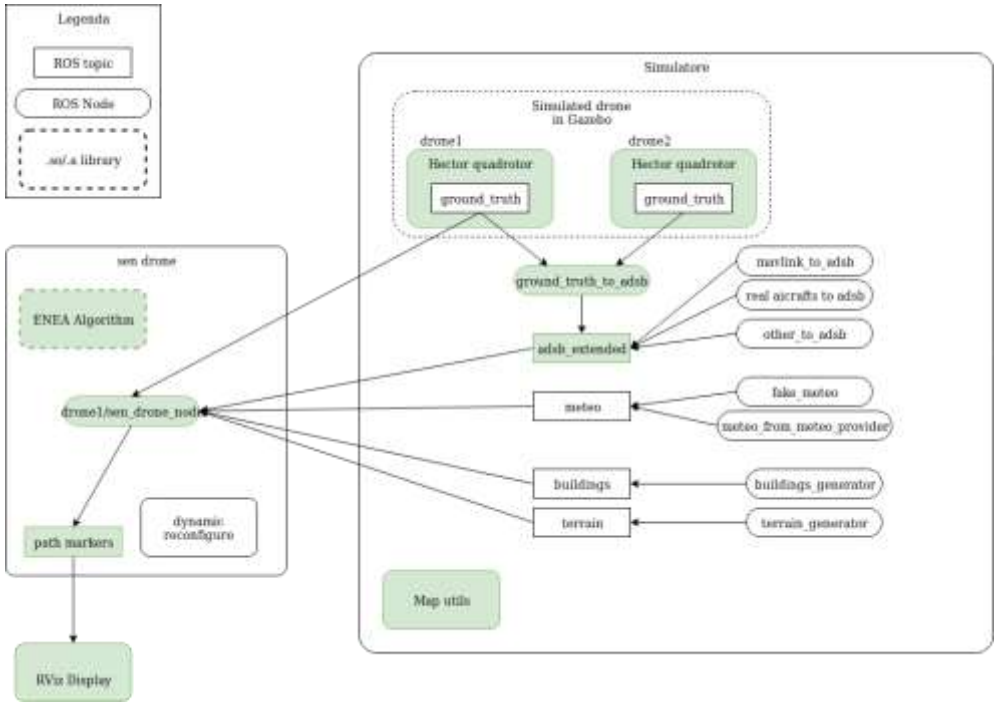

Figura 10. Architettura di dettaglio per l'algoritmo

Sulla destra è visibile il blocco del simulatore già implementato nella precedente annualità [1], allo scopo di simulare l'utilizzo di un drone per il rilevamento di dati fotogrammetrici di edifici nello spettro del visibile e nel termico. Durante la corrente annualità sono state aggiunte alcune funzionalità al simulatore con l'obiettivo di estendere la simulazione ai possibili conflitti che il drone può incontrare durante la propria attività. In particolare questi conflitti possono essere dovuti alla presenza di altre entità (come altri droni o edifici) lungo la traiettoria che il veicolo deve seguire.

Nello specifico in Figura 10 all'interno del blocco del simulatore sono presenti due RPAS; per ogni drone che si intende inserire nella simulazione è necessario eseguire il *launch* file xml riportato in Figura 11, passando dei parametri che definiscono il drone da aggiungere. Questi parametri sono:

- name: nome identificativo;
- offset: dove posizionare il drone nella simulazione;
- joypad_mapping: un file che definisce la mappatura dei comandi del joypad;
- joypad_dev: il nome del device che controlla il joypad.

```
<?xml version="1.0" encoding="UTF-8"?>
<launch>

  <env name="GAZEBO_MODEL_PATH" value="$(find sen_drone)/models:$GAZEBO_MODEL_PATH"/>
  <env name="GAZEBO_RESOURCE_PATH" value="$(find sen_drone):$GAZEBO_RESOURCE_PATH"/>

  <arg name="name" default="quadrotor"/>
  <arg name="offset" default="0.0"/>
  <arg name ="joypad_mapping" default="$(find hector_quadrotor_teleop)/launch/logitech_gamepad_nero.launch"/>
  <arg name ="joypad_dev" default="/dev/input/js0"/>

  <node pkg="tf" type="static_transform_publisher" name="world_broadcaster_$(arg name)" args="0 0 0 0 0 0 1 world $(arg name)/world 10" />
  <node pkg="sen_drone" type="ground_truth_to_adsb" name="ground_truth_to_adsb_$(arg name)" args="$(arg name)" />

  <group ns="$(arg name)">
    <include file="$(find hector_quadrotor_gazebo)/launch/spawn_quadrotor.launch" >
      <arg name="model" value="$(find hector_quadrotor_description)/urdf/quadrotor_hokuyo_utm30lx.gazebo.xacro"/>
      <arg name="world_frame" value="world"/>
      <arg name="name" value="$(arg name)"/>
      <arg name="tf_prefix" value="$(arg name)"/>
      <arg name="name_space" value="$(arg name)"/>
      <arg name="controllers" value="
          controller/attitude
          controller/velocity
          controller/position
          "/>

      <arg name="x" default="$(arg offset)"/>
      <arg name="y" default="0.0"/>
      <arg name="z" default="10.0"/>
    </include>
    <include file="$(arg joypad_mapping)">
      <arg name="joy_dev" value="$(arg joypad_dev)"/>
    </include>
  </group>
</launch>
```

Figura 11. Istruzioni nel launch file per un drone aggiuntivo

Le istruzioni ROS relative ad un dato drone sono raggruppate in un *namespace* sfruttando il *tag* 'group'. I *namespace* consentono di creare degli ambienti virtuali in ROS dove è possibile lanciare più istanze dello stesso nodo, e di conseguenza creare più *topic* con lo stesso nome senza che questi vadano in conflitto tra loro.

Con riferimento alla Figura 11: il tag *tf_prefix* consente di preporre il nome del nodo a tutte le variabili relativi a un drone; gli argomenti *x, y, z* permettono di impostare la posizione di partenza del drone che si sta aggiungendo; l'argomento *joy_dev* permette di impostare il dispositivo di input con cui controllare il drone simulato.

Per ogni drone viene avviato un nodo *ground_truth_to_adsb*. Questo nodo legge la posizione del droneX dal topic *droneX/ground_truth* e la converte in un messaggio ADSB da pubblicare sul topic */adsb_extended*. Il topic */adsb_extended* simula la presenza di un sistema di comunicazione basato su messaggi ADSB. In tale messaggio sono contenuti i campi mostrati in Figura 12.

In particolare i campi *next_wp_lat* e *next_wp_lon* contengono la latitudine e la longitudine del punto verso il quale il drone è diretto, tipicamente il prossimo waypoint. Questi campi non fanno parte dei messaggi ADSB tradizionali per cui il messaggio è chiamato ADSB_ext (extended ADSB).

L'utilizzo di un topic ROS per contenere i messaggi ADSB consente di integrare altre sorgenti di dati ADSB. Ad esempio è possibile inserire i dati ADSB provenienti dalla telemetria MAVLink [9] di un drone reale utilizzando un nodo ROS che converta questi dati nel formato ADSB_ext. Allo stesso modo è possibile integrare dati reali (da aerei in volo) provenienti da sistemi di acquisizione di dati ADSB o utilizzando servizi che forniscono dati di volo quali https://opensky-network.org o https://www.adsbexchange.com/.

```
#ADSB hex callsign
string  id
#Timestamp
string  timestamp
#Aircraft flightname
string  string_id
# lon, lat, next_wp_lat, next_wp_lon all in decimal degrees
float64 lat
float64 lon
float64 next_wp_lat
float64 next_wp_lon
#Altitude in feet
float64  altitude
# Speed in knots
float64  speed
#Heading in decimal degrees
float64  heading
# ROC in feet/min
float64  rateofclimb
```

Figura 12. Il messaggio ADSB

Ritornando alla Figura 10, il blocco sen_drone nella parte sinistra gestisce l'utilizzo dell'algoritmo *EneaAlgorithm()* nel sistema. In particolare i dati ADSB descritti precedentemente vengono letti dal nodo *sen_drone_node* che che si occupa di richiamare l'algoritmo e di fornire al nodo di display *RViz_Display* la posizione dei marker che descrivono la rotta (path) suggerita dall'algoritmo.

Sono anche presenti diversi altri possibili input (tramite topic ROS) all'algoritmo rappresentati da dati meteorologici forniti da un provider o simulati, dati relativi agli edifici, dati relativi al DEM (Digital Elevation Model, modello digitale delle altezze) del terreno. Di questi possibili input vengono al momento considerati i dati ADSB dei droni simulati, i dati degli edifici e quelli meteo, questi ultimi due non attraverso un topic ROS, ma direttamente da disco.

Una volta terminato il lavoro implementativo, il sistema è stato provato in sessioni sperimentali. Una tipica sessione è realizzata dalla realizzazione di una simulazione dove sono presenti:

- il drone per il quale si calcolano le traiettorie suggerite (pilotato tramite gamepad);
- almeno un ulteriore drone (pilotato tramite gamepad), impegnato a 'disturbare' il volo del primo;
- un edificio intorno al quale il primo drone deve volare.

In alcune sessioni sperimentali i droni con compiti di disturbo sono stati due. Queste sessioni di prova sono state svolte presso il Laboratorio di Robotica dell'ENEA e sono quindi descritte in [2].

2.2 Calibrazione di telecamere sensibili nell'infrarosso termico

Nell'ambito del sotto obiettivo d.2, nella presente annualità, si sono condotte attività volte alla ricostruzione tridimensionale di oggetti (edifici) a partire dalle immagini registrate da telecamere HD o termocamere. Per ottenere queste ricostruzioni risulta utile poter descrivere accuratamente le ottiche di ripresa delle camere. Questa descrizione si ottiene tramite un processo noto come calibrazione. Nel seguito sono descritte le attività condotte per la calibrazione di termocamere nella banda dell'infrarosso termico.

2.2.1 Modello di telecamera, distorsione e calibrazione

Una telecamera viene usualmente modellata utilizzando il cosiddetto modello pin-hole (in italiano foro di spillo, o, meglio, camera oscura [9]). Tale modello semplifica la descrizione matematica del processo utilizzando la geometria prospettica. In tale assunzione esistono due sistemi di riferimento: il sistema mondo ed il sistema della telecamera. Questo secondo riferimento è centrato nel centro ottico della stessa ed ha l'asse z orientato parallelamente all'asse ottico. Un generico punto P dello spazio tridimensionale ha coordinate $P_w \equiv [X_w, Y_w, Z_w]^T$ nel riferimento mondo e $P_c \equiv [X_c, Y_c, Z_c]^T$ in quello della telecamera. La trasformazione che lega P_w a P_c è funzione della posizione ed orientazione relativa tra i due sistemi di riferimento (ovvero i parametri estrinseci della telecamera) e può essere scritta tramite la:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \mathbf{t} \tag{1}$$

dove **R** è la matrice 3x3 di rotazione e **t** il vettore traslazione.

Il punto sarà poi proiettato dal sistema di lenti della telecamera nel punto immagine $p \equiv [u_u, v_u]$ sul piano focale della telecamera tramite una trasformazione che può essere espressa sia con la:

$$\begin{bmatrix} u_u \\ v_u \end{bmatrix} = \frac{f}{Z_c} \begin{bmatrix} X_c \\ Y_c \end{bmatrix} \tag{2}$$

dove f è la lunghezza focale, che, usando la notazione in coordinate omogenee che permette la linearizzazione della relazione, con la:

$$\begin{bmatrix} u_u \\ v_u \\ 1 \end{bmatrix} \propto \begin{bmatrix} \lambda u_u \\ \lambda v_u \\ \lambda \end{bmatrix} = \mathbf{F} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3}$$

dove λ è un fattore di scala ed **F** è la matrice trasformazione tra le coordinate mondo e le coordinate immagine. Nelle formule (2) e (3) è stato usato il pedice *u* ad indicare che questa proiezione prevede la condizione ideale di assenza di distorsione, ovvero lente ideale. Usualmente non è questo il caso ed è necessario modellare in un qualche modo la distorsione introdotta dalle lenti reali. Solitamente essa è modellata con uno sviluppo in serie fino ad un certo ordine nelle componenti radiale e tangenziale. Di

queste, la componente più significativa è usualmente quella radiale, a causa della quale l'errore di riproiezione su un pixel dell'immagine diviene funzione della sua distanza dal centro.

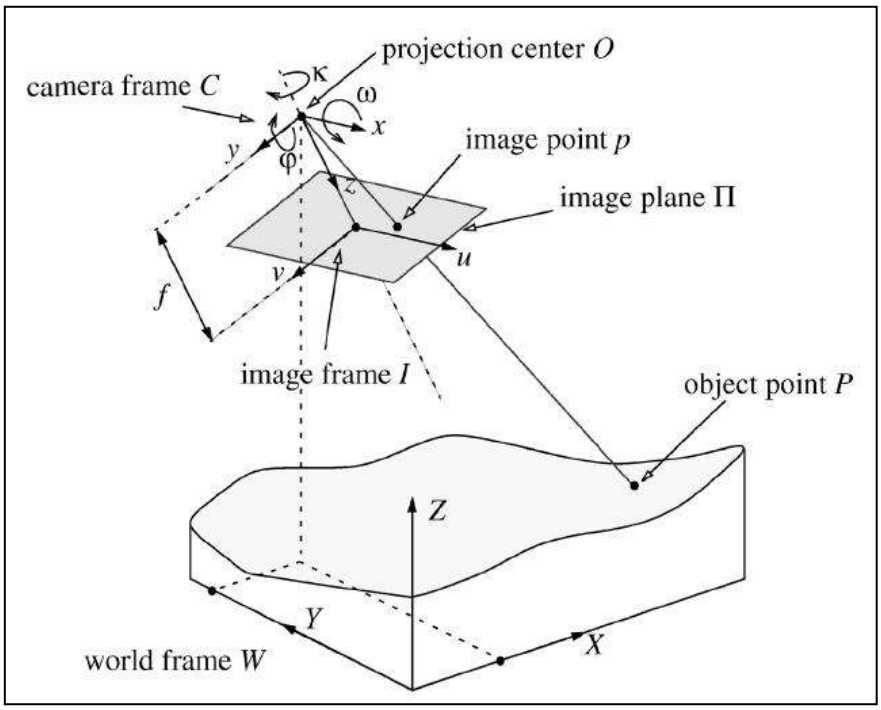

Figura 13. Riproiezione da coordinate mondo a coordinate camera

Prima di entrare in dettaglio è necessario fare una considerazione. L'uso diretto della equazione (3) per la calibrazione di un sistema ottico, prevede il calcolo degli elementi della matrice **F** a partire da coppie di misure di punti nel riferimento mondo e nel piano focale della telecamera. Questa procedura determina quello che viene usualmente chiamato il modello implicito della telecamera.

Per quanto detto sopra ed utilizzando la (1) e la (2) è possibile esprimere l'equazione (3) in una composizione di matrici in grado di rendere il modello della telecamera più chiaro fisicamente. Ovvero:

$$\begin{bmatrix} \lambda u_u \\ \lambda v_u \\ \lambda \end{bmatrix} = \mathbf{K}[\mathbf{Rt}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4}$$

dove **K** è la matrice dei parametri intrinseci della telecamera, **R** è la matrice rotazione e **t** il vettore traslazione così scritte:

$$\mathbf{K} = \begin{bmatrix} f\alpha & \gamma & c_x & 0 \\ 0 & f\beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad [\mathbf{Rt}] = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{5}$$

I parametri della matrice **K** variano lievemente al variare del modello di calibrazione impiegato, ma, comunque, contengono la lunghezza focale della lente *f*, il centro ottico (*cx, cy*) e la forma del pixel del rivelatore (*γ*).

Utilizzando questo secondo modo di esprimere il modello di una telecamera (ovvero l'equazione (4)) è possibile dunque separare i parametri estrinseci (rotazione e traslazione della telecamera rispetto al sistema di riferimento mondo) da quelli intrinseci (parametri interni del sistema ottico). I primi variano con il posizionamento spaziale della telecamera, mentre i secondi sono tipici di una data telecamera. In questo caso si parla di un modello esplicito della telecamera.

2.2.2 Modelli di calibrazione per telecamere

Come precedentemente anticipato, per ottenere una buona calibrazione di una telecamera è necessario tenere in considerazione anche l'errore introdotto dalla non idealità del sistema di lenti. La modellazione viene usualmente compiuta tramite uno sviluppo in serie fino ad un certo ordine, diverso al variare del modello, che tiene in considerazioni termini radiali e, per taluni modelli, componenti tangenziali. Di solito i termini radiali sono quelli più importanti in quanto le lenti utilizzate sono normalmente solidi di rotazione, ma il considerare anche i termini tangenziali può migliorare la modellazione complessiva del sistema.

Esistono diversi modelli in letteratura, ad esempio il modello di Tsai [10], di Heikkila [11] e di Zhang [12], che possono utilizzare dei pattern di calibrazione planari o tridimensionali. Nel seguito si è utilizzato un pattern di tipo planare (scacchiera) utilizzando un modello di Zhang.

2.2.3 Procedura di calibrazione

La calibrazione di una telecamera viene compiuta utilizzando generalmente un pattern appositamente studiato, di cui si conoscano le caratteristiche geometriche, che occupino la maggior parte dell'angolo di campo della telecamera. Il cuore dell'operazione di calibrazione è la risoluzione di un sistema di equazioni sovradeterminato rappresentato dall'equazione (4), tenendo in considerazione anche il modello di distorsione. Per ovviare alla non linearità introdotta dai termini radiali e tangenziali della distorsione si utilizzano tecniche di minimizzazione non lineare per rendere l'errore di riproiezione il minore possibile; al termine di questa operazione si otterranno così i valori per i parametri intrinseci ed estrinseci ed i coefficienti di distorsione.

Riassumendo, il procedimento per passare dalle coordinate mondo a quelle reali in pixel nel piano retina (considerando la distorsione) si concretizza nei seguenti passaggi:

1) passaggio dal sistema di riferimento del mondo a quello della camera:

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \begin{vmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{vmatrix} \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

ovvero:

$$P_c = R \cdot P_w + T$$

2) trasformazione prospettica ideale dei punti del sistema camera nei punti immagine del piano retina della camera (coordinate ideali non distorte):

$$\begin{cases} x = -f \cdot \dfrac{X}{Z} \\ y = -f \cdot \dfrac{Y}{Z} \end{cases}$$

3) introduzione della distorsione (passaggio da coordinate ideali a reali):

$$\begin{cases} x_d = x + \partial_x(x, y) \\ y_d = y + \partial_y(x, y) \end{cases}$$

4) passaggio da coordinate immagine reali a coordinate immagine in pixel:

$$\begin{cases} u = x_d \cdot k_u + u_0 \\ v = y_d \cdot k_v + v_0 \end{cases}$$

Da questa sintesi si può immediatamente notare come i parametri da calibrare siano quindi: i 6 estrinseci del moto rigido tra sistema mondo e quello camera, e quelli intrinseci ovvero la lunghezza focale *f*, le coordinate del punto principale u_0 e v_0 e i vari parametri del modello di distorsione utilizzato.

In Figura 14 sono mostrate tipiche immagini utilizzate per la calibrazione di una telecamera nel visibile. Il pattern utilizzato è una scacchiera in modo da poter facilmente estrarre gli angoli con semplici operazioni di elaborazione immagini quali gradienti ed estrazioni di bordi.

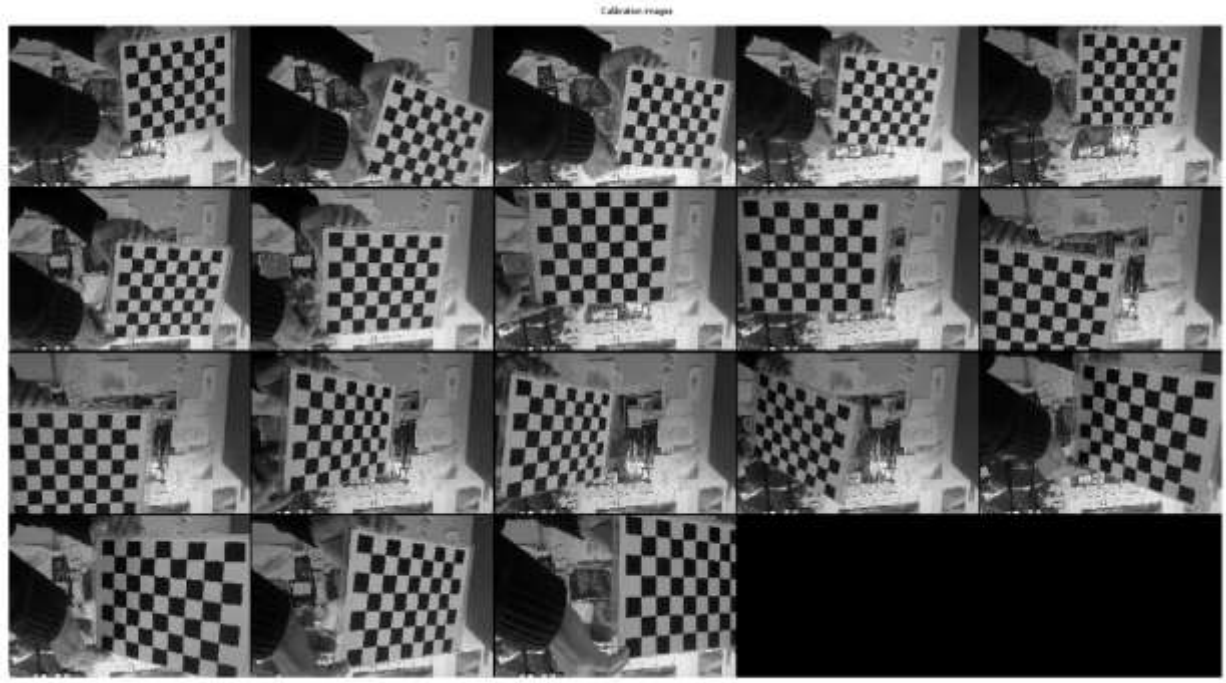

Figura 14. Immagini utilizzate per la calibrazione della telecamera nella banda visibile

Calibrazione di una telecamera nell'infrarosso termico

La calibrazione di una telecamera nell'infrarosso termico presenta una difficoltà ben maggiore rispetto a quella nella banda del visibile. Il problema principale è mostrato in Figura 15: nella figura si vede a sinistra una immagine del pattern di calibrazione ripresa nel visibile, è evidente come siano facilmente estraibili le posizioni degli angoli degli scacchi, necessari alla calibrazione in modo automatico. Al contrario nell'immagine a destra è mostrato il medesimo pattern ma ripreso da una termocamera; è evidente come il pattern sia poco visibile e come i vertici dei quadrati della scacchiera siano più difficili da determinare con precisione.

Figura 15. Il medesimo pattern di calibrazione nella banda visibile e nell'infrarosso termico

Ciò è dovuto al fatto che le differenze di temperature tra i bianchi ed i neri della scacchiera sono minime. E' necessario trovare un qualche trucco per aumentare la dinamica dell'immagine permettendo così la misura automatica degli angoli degli scacchi oppure cambiare strategia realizzando un pattern di calibrazione più adatto alle necessità.

Si è deciso di perseguire la seconda strategia disegnando un pattern realizzato in legno con una serie di fori equispaziati all'interno dei quali è stato posto un filo di rame. Il filo, tagliato a filo del pannello da un lato, dall'altro è stato disposto in modo da poter essere facilmente riscaldato ad esempio con un termoventilatore o una superficie radiante (termosifone). Una volta posti in contatto con la sorgente di calore, i fili di rame si riscaldano al contrario della struttura di legno; rimossa la sorgente è ora possibile utilizzare il pannello come pattern per la termo camera. In Figura 16 è mostrato il pattern in legno dal lato anteriore (quello che sarà ripreso dalla termocamera) e posteriore e l'immagine termica del pannello così realizzato. Il nastro adesivo metallico visibile nel retro del pattern serve a distribuire in modo quanto più uniforme possibile il calore ad i fili di rame.

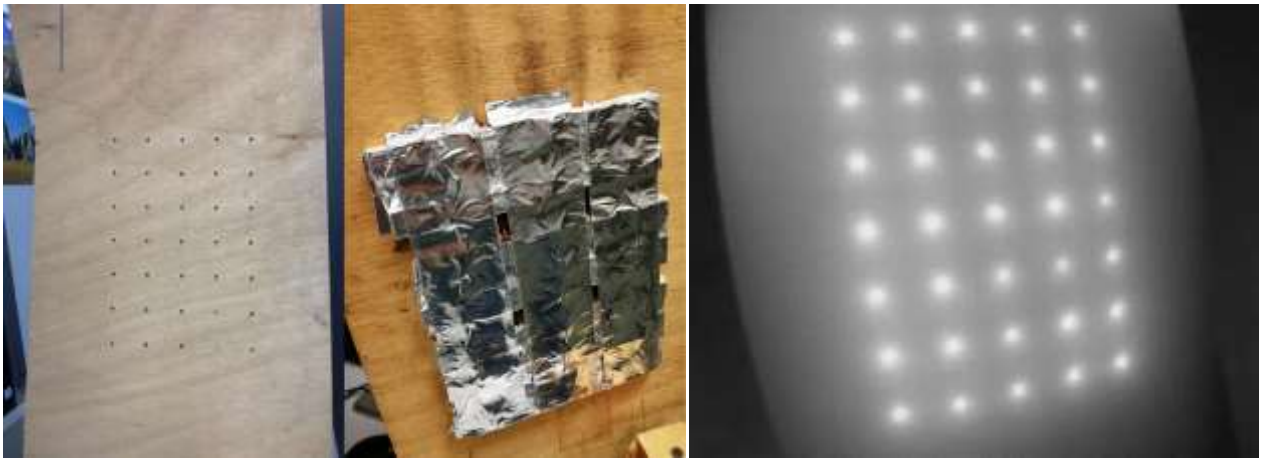

Figura 16. Il pattern termico realizzato (fronte e retro) e l'immagine termica del pattern

In questo modo i punti di calibrazione risultano sufficientemente evidenti nel segnale termico. Per automatizzare l'estrazione della posizione dei punti del pattern è stata realizzata una procedura di elaborazione dell'immagine sotto GNU Octave: un ambiente software per l'analisi numerica sotto S.O. Linux, ma ampiamente compatibile con il più noto MATLAB.

Una volta acquisita la serie di immagini termiche del pattern, l'elaborazione delle stesse è stata compiuta utilizzando i seguenti passi:

- smoothing: utilizzando un filtro gaussiano per ridurre il rumore a piccola scala;
- riduzione dei livelli di grigi a 2 (bianco/nero) applicando una soglia che lasciasse circa il 95% dell'immagine come sfondo e quindi il 5% in primo piano;
- il calcolo dei centroidi delle regioni così individuate ed il loro salvataggio su disco.

In Figura 17 è mostrato il secondo passo della procedura, che mostra le regioni trovate sull'immagine termica relative ai 7 x 5 punti caldi.

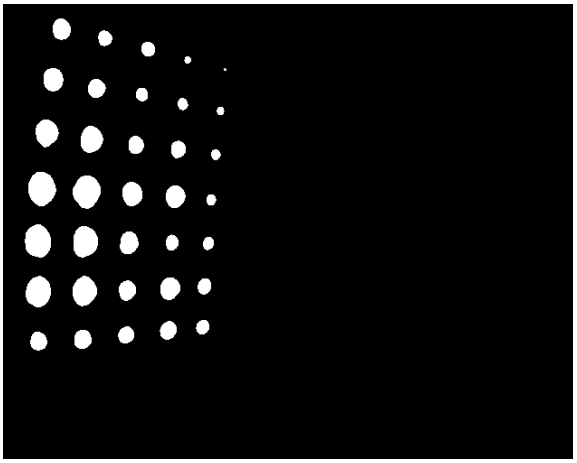

Figura 17. Le regioni calde trovate nell'immagine termica

Una volta calcolate le posizioni dei punti caldi del pattern termico, si è poi proceduto alla calibrazione utilizzando la procedura sopra esposta.

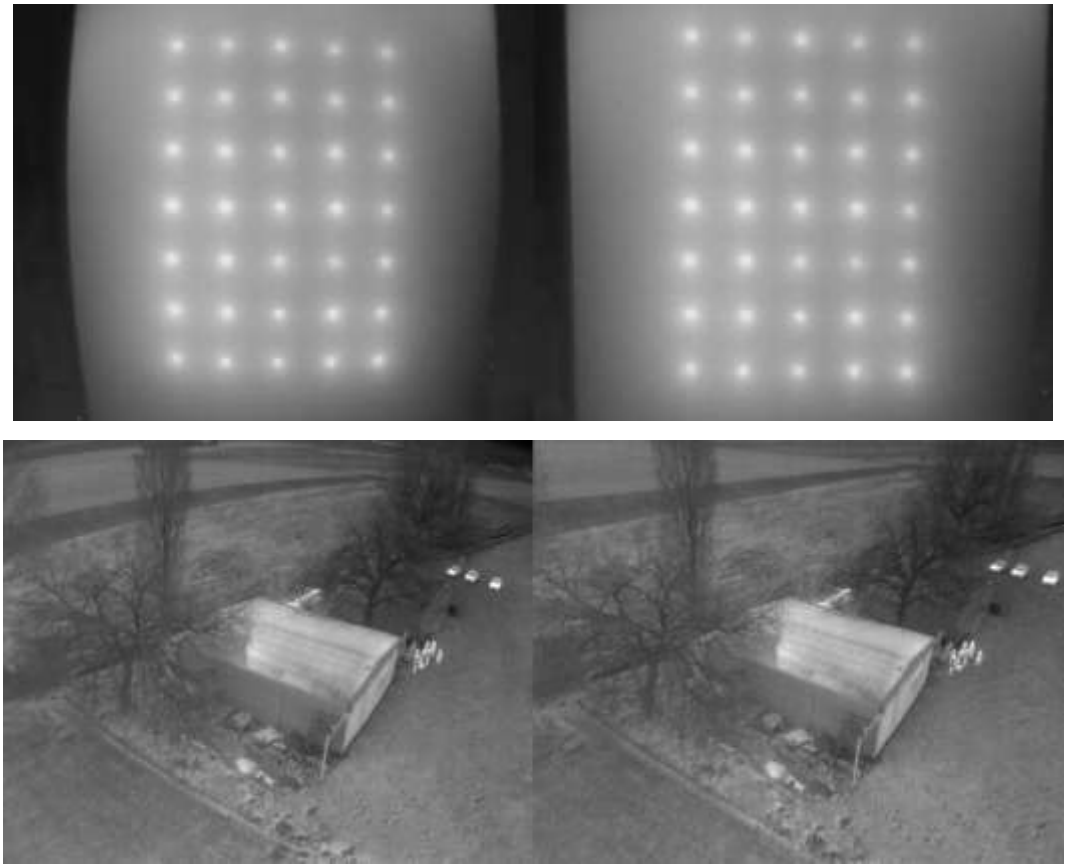

Figura 18. Immagini prima (sinistra) e dopo (destra) la correzione di calibrazione

In Figura 18 è mostrato l'effetto della correzione sul pattern di calibrazione stesso e su una immagine ripresa durante un volo di test al di sopra di un'aviosuperficie presso Terni.

E' chiaramente visibile l'effetto di 'raddrizzamento' delle linee soprattutto in prossimità dei bordi delle immagini.

3 Conclusioni

Questo documento ha descritto le attività svolte ed i risultati conseguiti dal Dipartimento di Ingegneria Elettronica dell'Università di Roma "Tor Vergata", nell'ambito dell'Accordo di Collaborazione con ENEA volto allo studio e sviluppo di algoritmi per l'ausilio al pilota e l'elaborazione dei dati sensoriali di bordo.

Gli ambiti di attività si sono articolati in aspetti teorici e di specifica, in aspetti di implementazione e programmazione e in test funzionali ed operativi.

Gli aspetti teorici e di specifica sono stati indirizzati allo studio di un algoritmo di separazione tra aeromobili sviluppato nel Laboratorio di Robotica dell'ENEA per studiarne la portabilità in un diverso ambito, quello dell'ausilio ad un pilota di RPAS. Una volta chiarita la parte teorica ci si è rivolti alla parte implementativa dove sono stati realizzati i moduli software per rendere operativo l'algoritmo all'interno del simulatore robotico, realizzato nel corso del PAR 2016, disegnato per l'addestramento dei piloti di drone. L'implementazione è stata condotta nell'ambito del *framework* ROS (Robot Operating System) entro cui è già operante il simulatore.

Ultimata l'implementazione del software si è passati ad una fase di test durante la quale si sono verificate le rispondenze con le specifiche. I test sono stati condotti in collaborazione con il Laboratorio di Robotica dell'ENEA e hanno previsto una serie di voli simulati di due o più droni contemporaneamente, fino ad un massimo di tre, intorno ad un edificio dello Smart Village del Centro Ricerche della Casaccia. La configurazione di test prevedeva il volo pilotato di due dei droni in modo completamente privo di vincoli, mentre il pilota del terzo drone riceveva suggerimenti dall'algoritmo allo scopo di non mettersi in situazioni di pericolo con l'edificio o con gli altri droni.

Va qui sottolineato che il SW realizzato per l'implementazione dell'algoritmo di *routing* è stato concepito in modo da poter essere facilmente scalato verso l'alto per poter considerare altri dati provenienti dall'area di volo. Infatti al momento esso considera unicamente l'interazione tra il drone e gli edifici, il meteo ed eventuali altri droni, ma è già prevista la possibilità di prendere in considerazione la presenza di aeromobili di qualunque tipo purchè segnalanti la propria posizione, velocità e direzione ed anche il modello geometrico del terreno.

Possibili sviluppi quindi in questo campo sono rappresentati dall'allargamento a tali dati dell'algoritmo e, soprattutto, ad un suo test in campo, al di fuori della simulazione cui è attualmente limitato.

Un secondo filone di attività si è diretto poi verso la calibrazione ottica di telecamere termiche. La necessità di una tale attività deriva dal fatto che una parte consistente delle attività del PAR 2017 sono volte alla realizzazione di modelli 3D sia nella banda del visibile che in quella dell'infrarosso termico. Per ottenere modelli più accurati è utile conoscere i parametri ottici delle telecamere utilizzate.

E' quindi stata realizzata una procedura per la modellazione dei parametri di una telecamera termica, creando pattern 'termici' e applicando alcune elaborazioni alle immagini riprese per un efficace calcolo dei parametri ottici.

4 Riferimenti bibliografici

[1] S. Taraglio, L. Blasi, G. Cupertino, C. Moriconi, V. Nanni, S. De Vito, F. Formisano, G. Zanini, F. Russo, M. Villani e L. Vitali, «Sviluppo di un sistema di monitoraggio aereo per lo smart district,» 2017. RdS/PAR2016/021.

[2] S. Taraglio, L. Blasi, G. Cupertino, C. Moriconi, V. Nanni, S. De Vito e F. Formisano, «Sistema di monitoraggio aereo per lo Smart District: elaborazione dati, ausilio al pilota, sviluppo sensori,» 2017. RdS/PAR2017/062.

[3] D. Taurino, S. Taraglio, A. Tedeschi, A. Pasquini e V. Nanni, «Satisficing Game Theory for enhancing autonomy in unmanned aerial vehicles,» *International Journal of Artificial Intelligence,* vol. 7, n. A11, pp. 316 - 328, 2011.

[4] «Teoria dei Giochi - Wikipedia,» [Online]. Available: https://it.wikipedia.org/wiki/Teoria_dei_giochi. [Consultato il giorno 28 9 2018].

[5] «Satisficing - Wikipedia,» [Online]. Available: https://en.wikipedia.org/wiki/Satisficing. [Consultato il giorno 28 9 2018].

[6] «ADS-B - Wikipedia,» [Online]. Available: https://it.wikipedia.org/wiki/ADS-B. [Consultato il giorno 28 9 2018].

[7] «ROS/Introduction - ROS Wiki,» [Online]. Available: http://wiki.ros.org/ROS/Introduction. [Consultato il giorno 28 9 2018].

[8] «Mapbox,» [Online]. Available: http://www.mapbox.com. [Consultato il giorno 28 9 2018].

[9] «Wikipedia - MAVLink,» [Online]. Available: https://en.wikipedia.org/wiki/MAVLink. [Consultato il giorno 28 9 2018].

[10] O. Faugeras, Three-dimensional computer vision: a geometric viewpoint, MIT Press, 1993.

[11] R. Tsai, «An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision,» in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Miami Beach, FL, 1987.

[12] J. Heikkilä, «Geometric Camera Calibration Using Circular Control Points,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, n. 10, pp. 1066 - 1077, 2000.

[13] Z. Zhang, «A flexible new technique for camera calibration,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, n. 11, pp. 1330 - 1334, 2000.

5 Indice delle figure

6 Appendice: Presentazione del Gruppo di Lavoro del Dipartimento di Ingegneria Elettronica

Il gruppo di ricerca in Reti e Comunicazione è uno dei gruppi del Dipartimento di Ingegneria Elettronica dell'Università di Roma "Tor Vergata". Il gruppo è stato fondato nel 1980 con l'obiettivo di condurre ricerche nei differenti campi delle tecnologie elettroniche e della comunicazione. Esso si occupa di elettronica ad alta frequenza, elettronica digitale, comunicazioni radio ed ottiche, reti, sensori e microsistemi, optoelettronica, misure elettroniche. I relativi campi di applicazione per queste tecnologie sono lo spazio, la medicina, i sistemi di difesa, le applicazioni industriali e i servizi di telecomunicazione.
Il gruppo è composto da 4 professori di ruolo, 3 professori associati, 9 ricercatori e più di 20 tra studenti di dottorato e post doc.

Le competenze del gruppo gravitano nelle seguenti aree:

- Reti wireless: WLAN, Device Wireless Programmabili, Reti Delay Tolerant;
- Comunicazioni Wireless: link MultiGbit/s, Ultra Wide Band a Onde Millimetriche;
- Comunicazioni Ottiche e "Radio su fibra": Sistemi di comunicazione su fibra ottica, Reti Ottiche (PON, MAN, Transport Network), Sistemi Ibridi Ottici Fibra-Radio, Sistemi Ottici Wireless, Signal Processing Ottico;
- Trustworthy Internet: Monitoring della Rete, Cybersecurity, Tecnologie Orientate alla Privacy;
- Architetture di Internet del Futuro: Information Centric Networking, Software Defined Networking, Network Functions Virtualization;
- Sistemi Radar: Rivelazione ed Analisi di Segnali Radar, Interference Analysis, Noise Radar, Multifunction Phased Array Radar (MPAR), High Resolution Radar;
- NavCom Devices: Location-awareness, Service Integration, NavCom Device Prototype;
- Tecnologie per Comunicazione in Banda EHF: Satellite Communications, Adaptive Modulation and Coding, Power Control, Optimization of DVB-S2 Standard, Communications in Q/V band.
- Protocolli e Reti Satellitari: TCP/IP, MAC, DAMA, DVB RCS, Cross-Layer Methodologies, Cloud Computing, Capacity Optimization.

Le attività del gruppo beneficiano dei seguenti laboratori:

- Laboratorio Reti;
- Laboratorio Comunicazioni Satellitari;
- Laboratorio Radar and Navigazione (RadarLab);
- Laboratorio HASCON (Harware and Algorithms for Systems of COmmunication and Navigation);
- Laboratorio Optoelettronico (Optolab).

I membri del gruppo hanno coordinato diversi progetti comunitari per un valore totale di 35 milioni di euro, ottenendo finanziamenti per 6.5 milioni di euro.

I progetti di ricerca internazionali e nazionali sono stati con:

- la Comunita Europea (EU);
- la European Space Agency (ESA);
- il Ministero dell'Istruzione dell'Università e della (MIUR);
- il Consiglio Nazionale delle Ricerche (CNR).

Il gruppo ha stabilito contratti di ricerca/consulenza con:
Alenia Spazio, Cisco, DoCoMo, EADS, Ericsson, Siemens, NEC, Space Software Italia/Elsag Datamat, Telecom Italia, Telespazio, PointerCom, Bull, Acotel, ENEA, ISCOM-Ministero dello Sviluppo Economico, Virtualabs.

Il gruppo ha contribuito a gruppi internazionali di standardizzazione:
ETSI: European Telecommunications Standards Institute, IETF: Internet Engineering Task Force, OASIS: Organization for the Advancement of Structured Information Standards, WWRF: Wireless World Research Forum.