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

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

SVILUPPO DI ALGORITMI DI AUTOLOCALIZZAZIONE SU DISPOSITIVI MOBILI

Tesi in
ELABORAZIONE DELLE IMMAGINI LM

Relatore:
Chiar.mo Prof. Alessandro Bevilacqua

Presentata da:
Ettore Esposito

Correlatori:
Prof. Alessandro Ricci
Dr. Alessandro Gherardi

Sessione I
Anno Accademico 2013–2014

Alla mia Famiglia

ABSTRACT

Negli ultimi anni, complice la rapida evoluzione degli elaboratori e dei sensori, spinta dal mercato smartphone, una tecnologia si sta sviluppando e si sta diffondendo rapidamente. Si tratta di quella relativa agli *unmanned vehicles* (UV), i veicoli senza pilota, spesso nel linguaggio comune chiamati "droni". Questi particolari veicoli sono dotati della tecnologia adatta per svolgere in relativa autonomia particolari mansioni, senza la necessità della presenza di un pilota a bordo. In questa Tesi magistrale si descrivono brevemente le diverse categorie di UV e l'attuale livello di autonomia raggiunta nello svolgimento di alcune funzioni, grazie a tecnologie quali i linguaggi ad agenti, di cui si presentano anche alcune significative applicazioni allo stato dell'arte. Per rendere più efficaci eventuali nuove funzionalità, fornendo una metodologia di sviluppo, atta ad aumentare il grado di astrazione, viene proposto un approccio architetturale a tre livelli. In particolare, viene approfondito il secondo livello, presentando l'implementazione di una funzionalità, l'autolocalizzazione spaziale, utile ad un sistema di terzo livello per arricchire la propria conoscenza dell'ambiente, al fine di raggiungere la massima autonomia nel controllo del mezzo. Questa prima esperienza ha consentito di approfondire le necessità in termini di hardware e software, al fine di poter effettuare una scelta mirata per l'ottimizzazione dei risultati ed un eventuale porting onboard, nella prospettiva di svincolare il mezzo da eventuali collegamenti con una stazione di terra, fino ad ora necessaria per eseguire le attività più complesse. Un interessante caso di studio consente di verificare la bontà del modello proposto e i risultati raggiunti nell'autolocalizzazione. In conclusione, si propongono ulteriori sviluppi che potranno fornire gli strumenti necessari alla massima espressione del potenziale che gli UV possiedono.

RINGRAZIAMENTI

Vorrei ringraziare tutte le persone che mi hanno permesso di realizzare questa Tesi di laurea magistrale, cominciando con colui che ha aiutato a tirar fuori da un insieme di componenti sconnesse una piattaforma di lavoro funzionante, il Dr. Matteo Turci del Flight Mechanics Laboratory dell'Università di Ingegneria Aerospaziale di Forlì. Ringrazio anche tutti gli altri membri del laboratorio che mi hanno dato l'opportunità di lavorare presso la loro sede, permettendomi di imparare molto anche su argomenti fino ad allora per me sconosciuti.

Ringrazio il Professor Alessandro Bevilacqua, il Dr. Alessandro Gherardi e il Professor Alessandro Ricci per avermi dato l'opportunità di svolgere un'attività in uno dei settori a cui sono più interessato, mettendomi a disposizione materiale, conoscenza e soprattutto offrendomi sempre del tempo prezioso per consultarsi e guidare il lavoro.

Ringrazio i miei amici per non aver denunciato la mia scomparsa durante questi mesi, ma che al contrario mi hanno aiutato, portandomi con la forza lontano dal monitor del mio computer a bere una birra fresca quando era proprio ora di staccare la spina.

Ed infine un caloroso ringraziamento alla mia famiglia che mi ha supportato, e sopportato, durante questi anni di università compresi gli ultimi mesi dello svolgimento di questa Tesi.

Contents

Chapter 1

Introduzione

In questi giorni una emergente categoria di veicoli tecnologici sta facendo parlare molto di sé. Si tratta degli *Unmanned Vehicles* (UV), i veicoli senza pilota. Questa tipologia di veicoli riempie le prime pagine dei giornali di cronaca e delle riviste specializzate per le possibilità ed i rischi che portano. Infatti, si tratta di veicoli dove la presenza di un operatore è sempre più superflua, acquisendo indipendenza mano a mano che la tecnologia migliora e i costi si riducono. Proprio l'autonomia di questi veicoli è un punto cardine di questa Tesi magistrale. Nel capitolo 2 verrà prima di tutto effettuata una rapida introduzione selle tipologie di UV, sulla loro suddivisione basata sull'applicazione finale e i loro impieghi allo stato attuale, con riferimento anche alle normative italiane sugli UAV, così come regolamentate dall'ENAC, categoria di maggior successo e tema centrale in questa Tesi.
Nel capitolo 3 si discuterà invece della necessità di autonomia di questi sistemi e dei compiti che possono essere portati a termine qualora si riesca ad arrivare ad un buon grado di autonomia. Si presenteranno, inoltre, alcune ricerche allo stato dell'arte dove l'uso di linguaggi ad agenti è stato proposto per tentare di modellare questi sistemi.
Il capitolo 4 introdurrà un'architettura a tre livelli con la quale strutturare un sistema autonomo, mostrando come siano necessarie infrastrutture di base al fine di semplificare il lavoro di scrittura di algoritmi autonomi. I singoli livelli verranno descritti separatamente, in particolare il secondo livello, che

si prefigge il compito di operare da mediatore fra il più basso livello macchina e il più alto livello gestionale.
L'intero capitolo 5 sarà invece dedicato a quegli algoritmi presenti in letteratura atti a consentire la autolocalizzazione online di una telecamera in un ambiente conosciuto e il mapping 3D della scena ripresa. Si descriveranno i processi matematici e le implementazioni pratiche di questi algoritmi, motivando cosi le ragioni della scelta di una particolare implementazione su tutte. Si descriveranno, infine, le modifiche apportate allo scopo di renderla compatibile con l'architettura hardware e la piattaforma software utilizzate.
Nel capitolo 6 si presenteranno e discuteranno i risultati ottenuti in questo lavoro di Tesi magistrale, in particolare la capacità di autolocalizzare fornita al UV, sulla base delle informazioni acquisite dalla telecamera. Verrà dapprima descritta la piattaforma hardware usata per lo sviluppo, introducendo brevi quanto necessarie descrizioni della sensoristica montata a bordo. Successivamente saranno analizzate le diverse librerie software utilizzate nello sviluppo di tutte le sottoparti che hanno permesso un efficiente ed efficace utilizzo del hardware a disposizione. Al termine, saranno discussi esperimenti, con particolare attenzione ad alcuni dei parametri più importanti, e i risultati ottenuti saranno presentati in veste grafica per una più semplice comprensione ed analisi.
Infine, nel capitolo 7 saranno tratte alcune conclusioni sul lavoro svolto seguite da alcune considerazioni sviluppi.

Chapter 2

I veicoli autonomi

Un veicolo autonomo, o unmanned vehicle (UV) in lingua anglosassone, è per definizione un veicolo senza persona a bordo capace di svolgere compiti più o meno complessi in autonomia. Esso può essere a pilotaggio remoto, come spesso accade, o completamente autonomo. Soprattutto è sulla completa autonomia che molta della ricerca e dello sviluppo si sta concentrando. La continua miniaturizzazione della sensoristica e la disponibilità di elaboratori sempre più potenti e parsimoniosi, con complice la forte spinta data dal boom economico degli smartphone, stanno dando a progettisti e sviluppatori la possibilità di integrare sempre più funzionalità sgravandole dagli operatori umani. I sistemi di controllo diventano sempre più precisi, facilitando il pilotaggio. L'integrazione con il GPS permette una localizzazione globale e spesso anche la possibilità di decidere percorsi a priori che il veicolo deve solo seguire. Telecamere, sia a colori che all'infrarosso, o di altro genere, montate su gimball permettono di acquisire immagini sull'ambiente circostante tutt'intorno al veicolo per una successiva analisi a terra. Termometri, barometri ed anemometri invece sono atti ad effettuare misurazioni utili alle previsioni del tempo. C'è chi pensa di montare addirittura sistemi di trasmissione dati avanzati a bordo di questi veicoli per portare tecnologie come Internet in territori ancora difficilmente raggiunti (Facebook e Google come primi pionieri, ma altri seguiranno se la sperimentazione dovesse rivelarsi proficua). E perché no, magari in futuro saranno usati per consegnarci

la corrispondenza fino a casa [1]. Solitamente questi veicoli hanno sigle differenti in base all'ambiente nel quale sono chiamati a lavorare. Abbiamo dunque unmanned ground vehicle quando operano su terra, unmanned surface/undersea vehicle quando fiumi, laghi, mari ed oceani sono i principali oggetti di esplorazione ed i famosissimi, almeno per la cronaca, unmanned aerial vehicle che operano in aria nei più disparati settori economico/militari. Esistono anche gli unmanned space vehicle, ma su questi non ci soffermeremo più di tanto in quanto si tratta di comuni satelliti che a centinaia circondano il nostro pianeta, o sono diretti verso altri pianeti, a fini sia scientifici, che commerciali, che militari. Citeremo solo qualche piccola evoluzione in questo settore.
Procediamo analizzandoli uno ad uno per chiarire le idee su questi numerosi veicoli destinati a dominare sempre più le quotidiane attività giornaliere.

2.1 Unmanned Ground Vehicle - UGV

Gli unmanned groud vehicle sono veicoli che si trovano ad operare solo su terra. Ne troviamo di tutte le dimensioni, ad esempio fan parte di questa categoria i piccoli robot teleguidati, mezzi come quelli usati per il disinnesco di bombe da parte di forze di polizia ed artificieri. Nel settore militare si stanno sviluppando anche veri e propri UV antimina, capaci di scansionare autonomamente una porzione di territorio precedentemente definito in remoto su un computer. Una volta trovata una mina poi si può tentare di rimuoverla o addirittura farla brillare insieme al veicolo se quest'ultimo è relativamente economico o costruttivamente semplice da riparare. Ultimamente si stanno realizzando robot UGV per il supporto tattico alla fanteria, ad esempio per il trasporto di attrezzatura sul campo, come il progetto BigDog [2] realizzato dalla Boston Dynamics, recentemente acquisita da Google, all'interno di un progetto DARPA. Esistono anche mezzi armati capaci di affrontare il campo di battaglia, come il Gladiator dell'esercito americano, che possono sostituire completamente la forza umana con mezzi armati capaci di affrontare il campo di battaglia come il Gladiator dell'esercito americano [3].

Figure 2.1: Il BigDog della Boston Dynamics è capace di affrontare alture e terreni sconnessi senza problemi.

Figure 2.2: Gladiator, uno degli UGV a disposizione dell'esercito americano.

Fanno parte di questa categoria anche le autonomous cars, auto a guida autonoma largamente studiate in questi anni sia dalle università che dalle aziende private. Vi è per esempio la vettura che ha percorso in modo completamente autonomo 13000 km, da Parma a Shanghai [4], realizzata dal VisLab [5] dell'Università di Parma. E' un settore dove anche grandi aziende stanno investendo molte risorse, come Google, che sembra avere uno dei progetti più maturi dopo più di 700000 miglia percorse (più di un milione di chilometri), tant'è che ha realizzato una propria autovettura priva di sterzo e pedali ed inizierà un programma sperimentale su strada con più di cento veicoli [6].

Figure 2.3: I veicoli autonomi del domani secondo Google. A breve inizierà una sperimentazione con più di cento di questi veicoli per le strade.

A seguire troviamo altre case automobilistiche [7, 8] come Mercedes [9, 10], BMW [11, 12], Audi [13], Ford [14], Nissan [15], Toyota, Lexus [16, 17], Volkswagen [18, 19], Volvo [20], ecc ... che vedono in questi veicoli un prossimo mercato, oltre alla possibilità di studiare nuovi sistemi di assistenza alla guida.
Aziende come Intel cominciano a preparasi alla nascita di questi nuovi mercati con tecnologie apposite come quelle appena presentate [21]. Ci sono ancora molti problemi legali da risolvere prima che queste automobili autonome possano cominciare ad essere commercializzate, ma sono molti i paesi che hanno cominciato a rivedere il loro codice per prepararsi al giorno in cui i produttori inseriranno a listino i primi modelli [22].

2.2 Unmanned Surface/Undersea Vehicle - USV/UUV

Quando i nostri veicoli autonomi sono chiamati ad operare sulla superficie dell'acqua vengono denominati *unmanned sea/surface vehicle*, o semplicemente USV. Sono invece detti *unmanned undersea/underwater vehicle*, UUV, se operano sotto la superficie. Questi veicoli possono essere utilizzati per abbattere molto i costi sullo studio dei mari, come lo studio delle correnti, le composizioni chimiche, le maree, gli tsunami. Sono utili anche in caso di disastri per effettuare il monitoraggio della zona, ad esempio la diffusione di una chiazza di petrolio in un incidente navale. Possono essere adoperati anche per ricognizioni marine al fine, per esempio, di effettuare stime sui danni e sulla pericolosità per l'ambiente di scavi marini, ma anche per raccogliere informazioni utili alla preparazione di un piano per l'invio di sommozzatori in presenza di relitti. Possono essere utilizzati anche per effettuare ricognizioni a seguito di incidenti aerei, come avvenuto dopo l'incidente dell'Air France Flight 447, dove è stato impiegato un AUV Abyss, un UUV realizzato dalla Hydroid [23], per effettuare ricognizioni sottomarine.

Figure 2.4: Abyss nel suo ambiente naturale.

Anche in questo caso le dimensioni possono variare molto, da piccoli radiocomandi atti a raccogliere dati scientifici autonomamente e a basso costo, a vere e proprie barche dotate di maggior autonomia, braccia meccaniche, radar

e potenti motori. (Si vedano ad esempio i veicoli realizzati dalla SeaRobotics [24]). Di recente la Rolls Royce ha annunciato di voler sviluppare navi cargo controllate da remoto [25]. Iniziativa probabilmente collegata al recente finanziamento di 3.5 milioni di Euro da parte dell'Unione Europea di un progetto chiamato *Maritime Unmanned Navigation through Intelligence in Networks* [26], il cui scopo è proprio quello di sviluppare e verificare il concept di una nave completamente autonoma, guidata cioè dai sistemi on-board installati sulla nave e controllata da un operatore remoto da una stazione di terra. Il progetto si prefigge l'obiettivo di realizzare navi prive di equipaggio con lo stesso livello di sicurezza delle attuali navi con equipaggio, ma l'assenza di quest'ultimi permetterebbe di abbassare i costi di gestione della flotta, permettendo la costruzione di navi più leggere, più economiche e capienti. Ovviamente, come per le controparti stradali, le normative attuali non permetterebbero ancora a queste navi di poter navigare, ma probabilmente per quando cominceranno le prime sperimentazioni qualcosa sarà già cambiato.

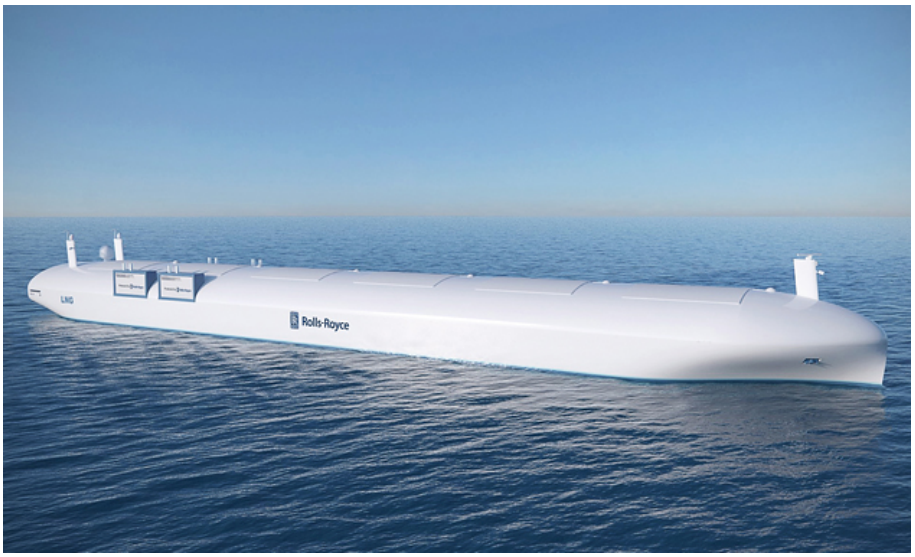


Figure 2.5: Concept della nave senza equipaggio proposta dalla Rolls Royce.

2.3 Unmanned Space Vehicle - USV

Gli *Unmanned space vehicle* chiamati anche *unmanned spacecraft*, sono probabilmente la categoria più antica di veicoli autonomi. Rientrano in questa categoria infatti tutti i satelliti che orbitano intorno alla terra, a partire quindi dal primo satellite mandato nello spazio, lo Sputnik 1 del 1957. Da allora in questo settore abbiamo fatto molti progressi. Oggi se da una parte ci sono ancora progetti di un certo spessore economico come il Boeing X-37 [27], ci sono anche progetti che cercano di realizzare satelliti relativamente economici e facili da programmare come CubeSat, che permette di predisporre una semplice piattaforma per effettuare ricerca nello spazio, a patto di trovare un passaggio [28, 29].

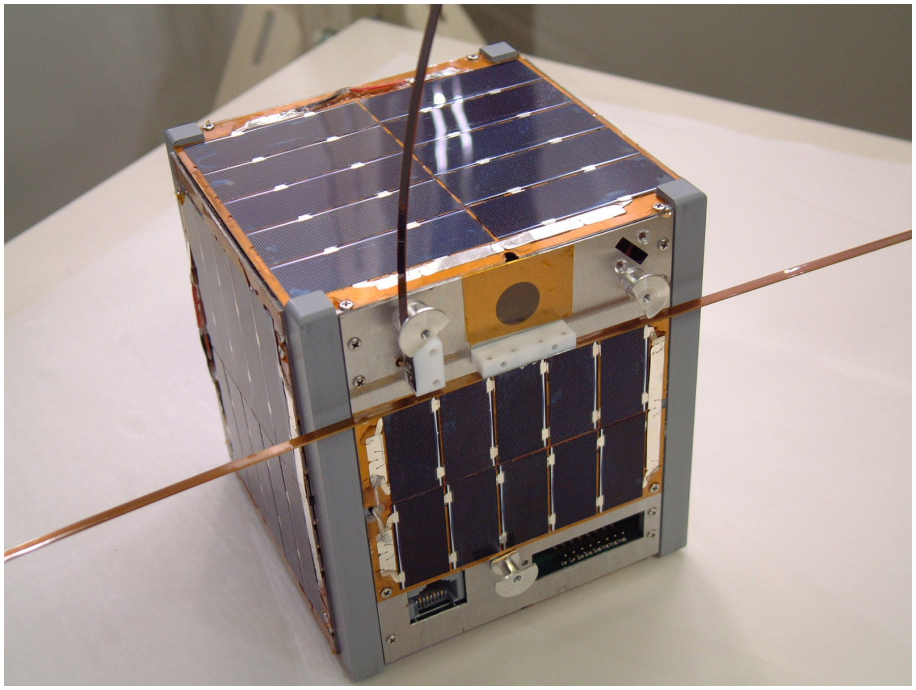


Figure 2.6: Uno dei CubeSat dell'Università di Tokyo.

2.4 Unmanned Aerial Vehicle - UAV

Gli *unmanned aerial vehicle*, volgarmente detti droni, termine però che sempre più viene criticato dagli esperti del settore perché riduttivo nei confronti di ciò che queste macchine possono fare, sono quei veicoli autonomi che operano in aria. Sono utilizzati in una variegata gamma di applicazioni. Ad esempio nel settore militare, veicoli avanzati come il Predator, il Reaper o il Global Hawk, svolgono una grandissima varietà di missioni, fra cui ricognizione, spionaggio e bombardamenti. Questi veicoli vengono spesso menzionati con la sigla di UCAV, ovvero *unmanned combat aerial vehicle*, o RPA, *remotely piloted aircraft*, aeromobili a pilotaggio remoto, in quanto ad oggi ancora necessitano una stazione esterna che controlli il mezzo.

Figure 2.7: Il Boeing Phantom Ray alla sua presentazione ufficiale.

Complice la miniaturizzazione dei sensori e la riduzione dei costi, spinti dall'evoluzione del settore smartphone, anche il settore civile vede un uso sempre più massiccio di questo tipo di veicoli. L'evoluzione dei motori elettrici, diventati piccoli, efficienti e facili da utilizzare grazie all'elettronica di controllo (ESC: *Electronic speed control* [30]), ha rivoluzionato il mondo del radiomodellismo, portando vantaggi nella costruzione di piccoli UAV efficienti, spesso chiamati MAV, *Micro Aerial Vehicle*. Inoltre la continua evoluzione di queste componenti fondamentali per il controllo e l'automatizzazione permette alle applicazioni civili di avere una crescita continua. Questa tipolo-

gia di veicoli è spesso usata per attività come il monitoraggio ambientale, la sicurezza, ma anche utilizzati nel settore agricolo per massimizzare l'efficienza, in fasi come la semina e la distribuzione di fertilizzante, abbattendo i costi. Molti altri settori potenziali sono ad oggi ancora inesplorati, per limiti di natura tecnologica o burocratica. Hollywood e CNN ad esempio pressano la FAA per avere un regolamento certo e con la possibilità di usare agevolmente questi veicoli [31, 32]. Solitamente in campo civile si usano mezzi molto semplici da controllare e relativamente poco costosi come i multicopteri. I multicopteri sono infatti piccoli veicoli aerei, dotati di più eliche, dalle tre alle otto solitamente. Sono molto diffuse le configurazioni a quattro eliche per la loro semplicità e il loro costo basso, ottime per volare in indoor o sorvolare a bassa quota spazi pubblici. Sono meccanicamente più semplici di un elicottero anche se meno efficienti. In piccole dimensioni la semplicità meccanica permette di compensare appieno la minore efficienza. In futuro attività come search&reascue, in caso di incidenti in aree inaccessibili, il monitoraggio ambientale, la costruzione di mappe tridimensionali, potranno essere svolte in modo autonomo da queste macchine avanzate. Già oggi vengono utilizzati in ambito giornalistico o da parte delle forze dell'ordine per avere informazioni tattiche in tempo reale quando chiamati ad affrontare situazioni difficili. Molti centri di ricerca, come il DLR German Aerospace Center, sperimentano l'inserimento di braccia robotiche per manipolare l'ambiente circostante [33]. L'ETH ha mostrato come uno sciame di quadricopteri possa essere usato per la costruzione di strutture complesse [34], grazie ad una corretta coordinazione e suddivisione del lavoro. C'è una grande varietà di applicazioni che si possono svolgere con questi veicoli. Servono quindi avanzamenti software importanti per realizzare applicazioni complesse, serve un modello ingegneristico con il quale progettarli.

Figure 2.8: DJI Phantom Vision, un prodotto commerciale di gran successo.

Figure 2.9: Alcuni prodotti della gamma AscTec, molto apprezzati in ambito di ricerca.

2.4.1 Regolamentazione italiana ENAC per gli APR

In Italia gli UAV sono chiamati APR, Aeromobili a pilotaggio remoto. A cercare di regolamentare l'uso di questi veicoli, che, per quanto utili, possono rischiare di risultare anche molto pericolosi (ad esempio violando lo spazio aereo di un veicolo da trasporto civile o perdendo il controllo rischiando di ferire o danneggiare persone o cose a terra), ci penserà l'ENAC, Ente Nazionale per l'Aviazione Civile. Un documento pubblicato dall'ENAC e disponibile sul portale ufficiale contiene indicazioni specifiche sulle autorizzazioni necessarie per i piloti, sull'equipaggiamento, sulle certificazioni da ottenere prima del volo. L'Italia è dunque uno dei primi paesi ad adottare una normativa ufficiale sul tema, dimostrando un'ottica lungimirante per evitare di arrivare in un secondo momento impreparati e di dover gestire la situazione quando ormai troppo tardi.

Chapter 3

I sistemi autonomi

Sia nel settore civile che in quello militare la ricerca punta soprattutto alla realizzazione di veicoli che per svolgere il loro compito richiedano sempre meno l'intervento dell'operatore. Inizialmente i sistemi a bordo si occupavano semplicemente di semplificare il controllo, evolvendosi poi in sistemi capaci di svolgere funzioni avanzate come mantenere la posizione a fronte di forze di disturbo, seguire percorsi prestabiliti o acquisire i dati nel momento opportuno. Ma si cerca ancora più autonomia, sistemi che possano prendere vere e proprie decisioni basandosi sulla loro conoscenza a priori. Quale percorso è meglio seguire, quale azione è meglio effettuare. Quali degli obbiettivi prefissati ha maggior priorità e quale può essere scartato. L'intervento dell'operatore si pone pian piano su un livello sempre più astratto, la cui unica funzione è dare indicazioni, una conoscenza di base ed assistere al compimento degli obbiettivi come una figura passiva. Per ottenere questi risultati manca ancora molto lavoro da fare, ma una prima proposta è quella di usare i linguaggi ad agenti che offrono un modello concettuale molto vicino al risultato desiderato.

3.1 Linguaggi ad agenti e sistemi di veicoli autonomi

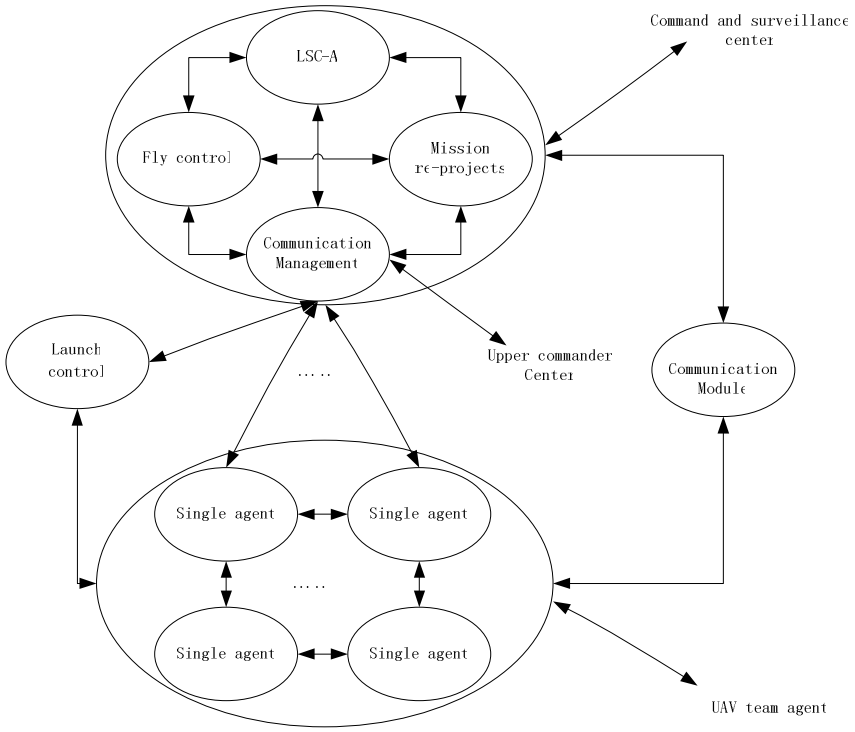
Cos'è un agente? Un agente è una entità autonoma che incapsula il controllo. Tutto ciò che può essere scambiato da un agente sono solo dati come conoscenza ed informazioni. La loro autonomia si traduce quindi in una proattività, cioè la capacità di modificare l'ambiente di propria iniziativa e non solo come reazione ad un cambiamento. Per poter far ciò gli agenti devono possedere una conoscenza dell'ambiente che li circonda, del contesto in cui si trovano ad agire. Devono cioè possedere una propria rappresentazione del mondo. Ma in un mondo non statico ma dinamico gli agenti hanno la necessità di dover percepire eventuali cambiamenti nel mondo, soprattutto per sapere quali azioni possono eseguire o conoscere se le proprie azioni hanno portato gli effetti desiderati. La componente reattiva non viene quindi meno negli agenti, che possono divenire per scelta anche puramente reattivi. Qualsiasi sia però il comportamento di un agente alla fine le sue azioni porteranno sempre ad un cambiamento intorno a se. Sia che le sue azioni vadano ad influenzare altri agenti, sia che vadano a modificare l'ambiente. L'autonomia di controllo potrebbe permettergli di avere una sorta di intelligenza che li aiuterebbe a governarsi, ma essa non è necessaria alla loro autonomia. Poiché un agente incapsula il controllo, esso può agli estremi divenire anche mobile se il suo controllo può in qualche modo essere indipendente dall'ambiente. Possedere un autonomia permette inoltre ad un agente di migliorarsi, imparando ed acquisendo nuove capacità. Possono addirittura, secondo una definizione più forte di agente, avere componenti mentali come idee, desideri, intenzioni, conoscenza [35]. Essi sono, in ultima analisi, interattivi, sociali, pro-attivi e reattivi. Sono caratterizzati da una conoscenza dell'ambiente e del contesto. Possono avere obbiettivi o compiti da completare. Possono essere intelligenti o mobili. Essi vivono in società composte da altri agenti, dette MAS, dove interagiscono tra loro tramite comunicazione e modificano l'ambiente con le loro azioni. Un sistema multi-agente è un insieme di agenti, dotati ciascuno della propria autonomia, che collaborano tramite scambio di informazioni. Infatti un sistema a singolo agente non esiste in principio.

Da un punto di vista filosofico infatti l'autonomia ha senso solo se un individuo è immerso in una società e nel caso degli agenti quindi un MAS. All'interno di esso questi agenti autonomi comunicano tramite scambio di dati, modificano il mondo tramite azioni e sono sensibili a queste variazioni tramite le percezioni. L'interazione di queste singole entità, che potrebbero avere comportamenti anche semplici, fa emergere nel sistema il comportamento desiderato. Senza che le singole entità siano per forza a conoscenza dell'obbiettivo comune del sistema.

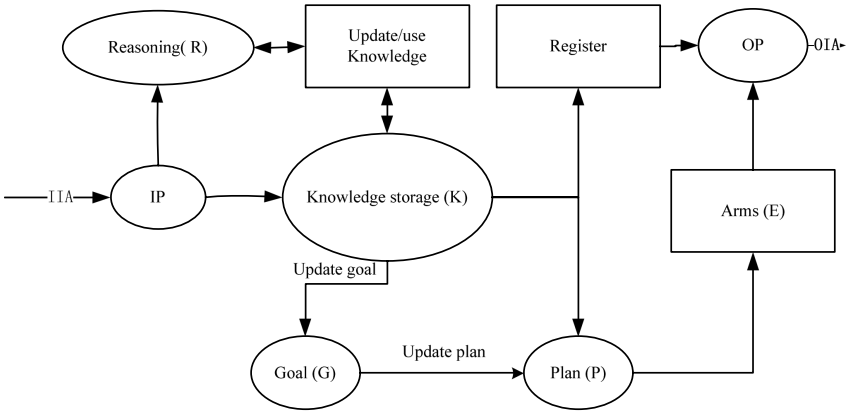
3.1.1 Esempi di applicazione di MAS ad *Unmanned Systems*

Prima di mostrare le applicazioni svolte in quest'ambito è giusto fare una precisazione. Ad oggi molti *unmanned vehicle* come detto precedentemente non sono degli *autonomous vehicle* ma possono anche essere dei semplici *remote piloted vehicle*. In questi casi il sistema svolge il cosiddetto compito di copilot, semplificando all'utilizzatore l'esecuzione di certe azioni ponendosi come intermediario. Inoltre soprattutto a livello civile quando questi veicoli sono piccoli e non dispongono di sufficiente capacità computazionale hanno a bordo una ridotta logica di controllo, il copilot per l'appunto od una sua versione più semplice, affidando poi ad una stazione a terra (GS) le funzionalità più avanzate. Ad esempio per i droni commerciali, quando si imposta una rotta da seguire la stazione a terra si occupa di tracciare un percorso fra i punti inseriti e di calcolarne tutti i parametri. In seguito invia al drone solo lo specifico vettore da seguire per giungere al punto successivo della rotta. Quindi si potranno avere sistemi multi-agente dove gli agenti saranno caricati direttamente sugli UV o sistemi multi-agente caricati sulla GS che avrà il compito poi di guidare gli UV. In [36] Magnusson, Lande'n e Doherty utilizzando il framework JADE per la costruzione di una simulazione, nella quale gli UAV effettuavano le azioni e comunicavano fra loro tramite azioni di comunicazione standardizzate, come *FIPA ACL speech acts*, hanno simulato la possibilità di realizzare un sistema multi-agente dove ogni agente giace su un UAV. Ognuno di esso ha la possibilità di pianificare le comunicazioni da

effettuare con altri agenti, le azioni da eseguire ed un piano per recuperare in caso di imprevisti come ad esempio un problema nelle comunicazioni. Questi agenti, fanno uso di tecnologie di *theorem proving* per pianificare ed eseguire le proprie azioni. Essi infatti si affidano ad una estensione delle Temporal Action Logic. Tramite un *deduction theorem prover* automatizzato genera e revisiona un piano in linea con l'ambiente circostante che è poi eseguito da un algoritmo STNU che dispacci le azioni in accordo ai vincoli temporali. Infine queste dispacci delle azioni vengono presi dalla piattaforma JADE che le esegue sulla piattaforma fisica del robot (simulato) in questo caso. Nello scenario proposto hanno dimostrato la flessibilità con cui un UAV incaricato di scannerizzare una cella in cerca di persone, a seguito di una perdita di connessione con il controllo a terra ha rapidamente riformulato il piano per appoggiarsi ad un secondo UAV che facesse da tramite per lo scambio di messaggi. In [37] si discute l'idea di come realizzare un UAVICS (*UAV Intelligence Control System*) introducendo il concetto di MAS nel suo design, che viene poi chiamato UAVTICS (*UAV Team Intelligence Control System*). In particolare provano a costruire la struttura del MAS e i gli agenti che ne faranno parte. Ad esempio il *Launch Control Agent* che si occupa delle azioni di preconfigurazione e decollo degli apparecchi, il *Command and Surveillance Center Agent* (CS-A) composto a sua volta da altri quattro agenti e che insieme supervisionano il corretto svolgimento della missione da parte del team di UAV. Il *Communication Module Agent* che deve occuparsi della comunicazione fra il UAVT Agent e il CS-A. UAVT Agent che rappresenta l'agente che si occupa del team di UAV e che è realmente composto da un insieme di *Single Unmanned Air Vehicle Agent*.



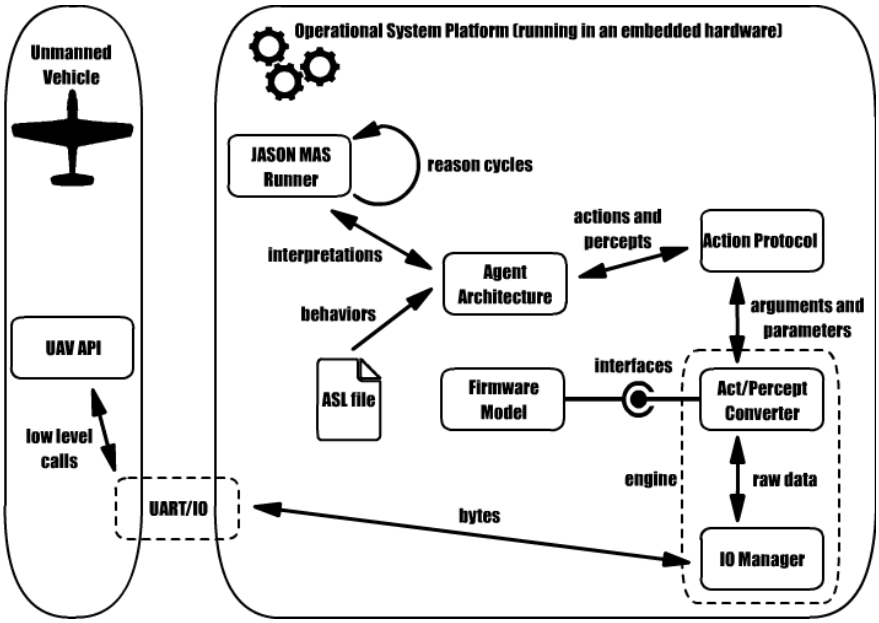
Del CS-A viene descritta l'architettura del suo *Fly-Control Agent*, pensata con l'idea principale di seguire un "approccio di integrazione uomo-macchina". Anche l'architettura del *Single Unmanned Air Vehicle Agent* viene trattata. In questo caso si è scelto di mantenere un architettura ispirata al modello BDI.



Inoltre viene spiegato come all'interno del team solo un agent ricopre il ruolo di leader (*lead plane agent*) e gli altri sono solo compagni (*wing plane agent*). Di questi è il leader che si occupa di comunicare con il CS-A ed ha il compito di decidere quando attaccare. Tutti gli altri agenti insieme al leader si limitano a scambiare le informazioni ottenute tramite i sensori sull'ambiente circostante e le distribuiscono in broadcast. In [38] si prova a comprendere quali siano i problemi nella realizzazione di un vero e proprio surrogato del pilota tramite agente BDI, implementato in JACK, in modo da ottenere una completa autonomia e non aver più quindi bisogno di affiancare ad un UAV una stazione a terra per il supporto alle decisioni. Questo permetterebbe di ridurre la presenza dell'uomo salvo condizioni in cui strettamente necessario, inoltre può portare in futuro ad una semplice implementazione di team di UAV completamente autonomi. Nelle simulazioni si è verificata la capacità di un team di dialogare correttamente e si sono verificati i corretti comportamenti. Tramite una prova reale si è dimostrata inoltre la capacità del veicolo di gestire il proprio volo in modo sicuro e autonomo, terminando la missione in caso di necessità. Questo sistema in futuro potrebbe prevedere la possibilità di applicazione di tecniche di volo avanzate difficilmente ottenibili tramite sistemi di controllo standard, ad esempio nel volo di un aliante che deve prendere in considerazione costi e benefici di diverse rotte in base alla conformazione del terreno. [39] introduce l'idea di un framework, che viene chiamato UAVAS, per descrivere il comportamento intelligente degli UAV tramite agenti specificati con Jason/AgentSpeak. Questo framework verrà esteso per creare una analogia in cui un agente Jason viene visto come un umano seduto all'interno del cockpit di un classico veicolo pilotato. Nell'astrazione creata questo modello viene realizzato:

***UAVAS* Model**		**Manned Aerial Vehicle**
UAV Agents	models	Aerial Vehicle Tripulation
AgentSpeak .ASL File	models	Crew Intelligence and Flight Plans
Action Protocol	models	Aerial Vehicle's Commands
Agent's Environment	models	Pilot Cabin
UAV	models	Manned Aerial Vehicle

Il modello concettuale di questo framework può essere riassunto dalla seguente figura:

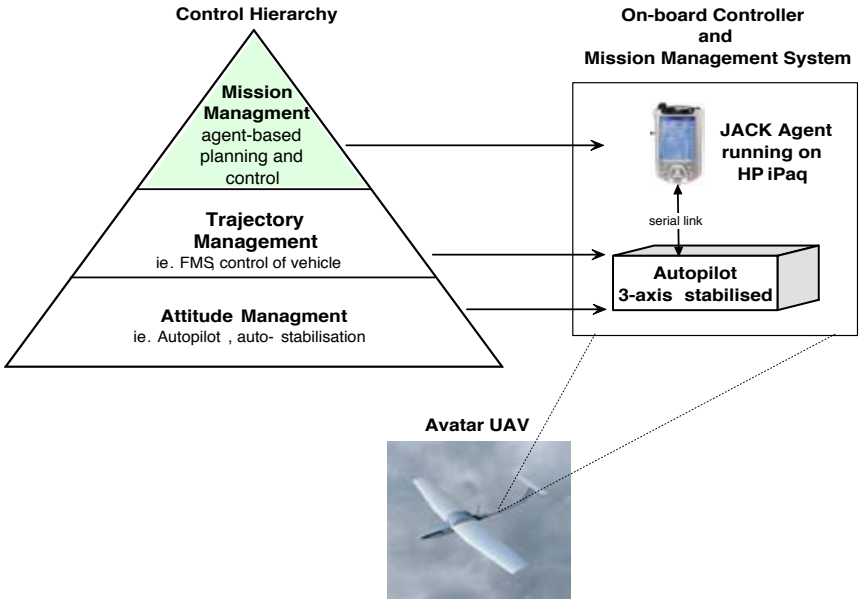


Il firmware dell'UAV è direttamente connesso all'agent tramite un collegamento asincrono detto UART (*Universal Asynchronous Receiver/transmitter*). Una serie di classi intermedie trasforma le azioni dell'agente in valori interpretabili dall'UAV. Un semplice decollo, simulato all'interno del simulatore RC-Sim, può essere descritto con il semplice codice:

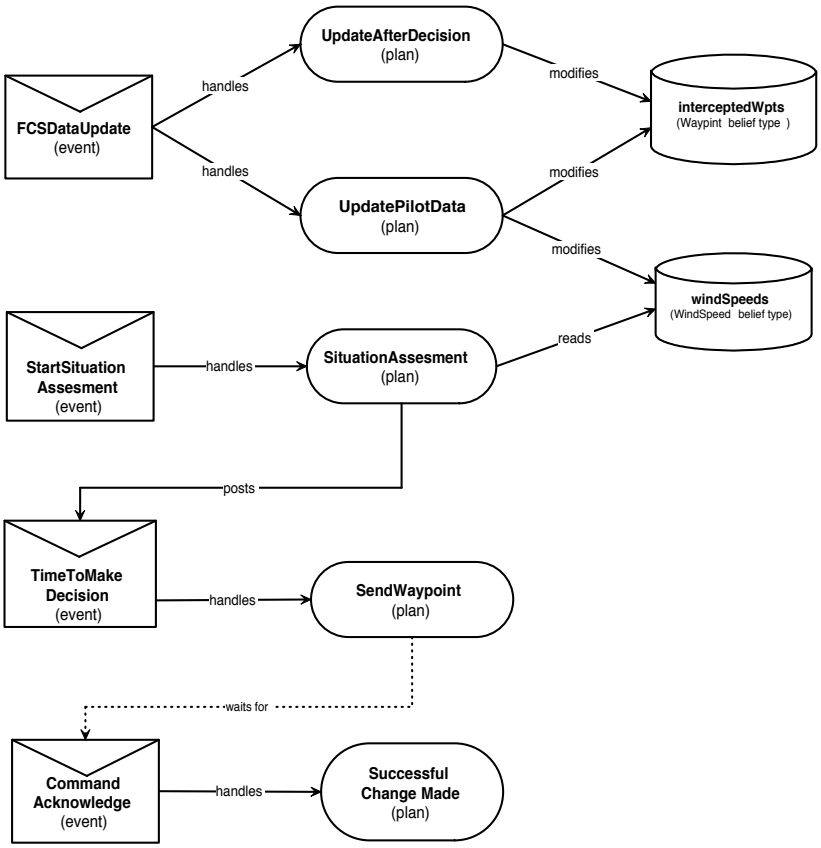
```
+!takeOff :
   status(ok) <-
   collective(8);
```

```
.wait(10000);
collective(4);
!doTheMission.
```

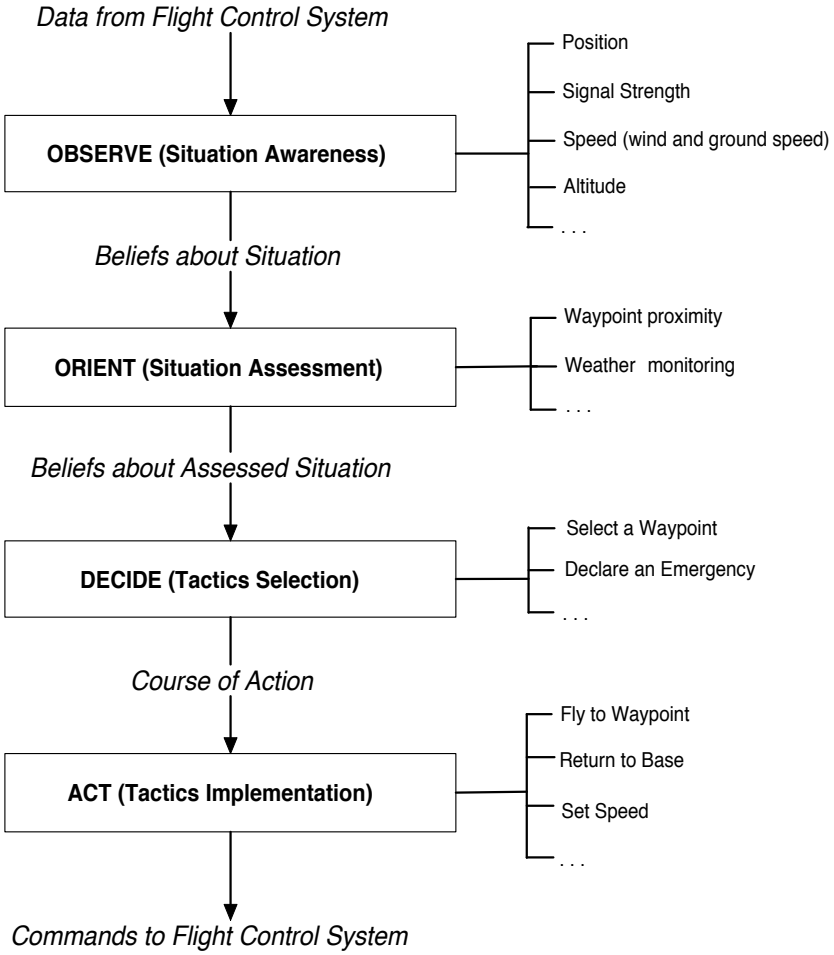
In [40] si mettono a confronto due approcci diversi con il quale applicare tecnologie *agent-based* per il controllo di un sistema di UAV, entrambe implementate in JACK [41]. In uno si applica un approccio standard in cui si aggiunge un livello sul sistema di controllo di volo, che girerà su una piattaforma separata e dialogherà con l'autopilot tramite un interfaccia fornita all'agente.



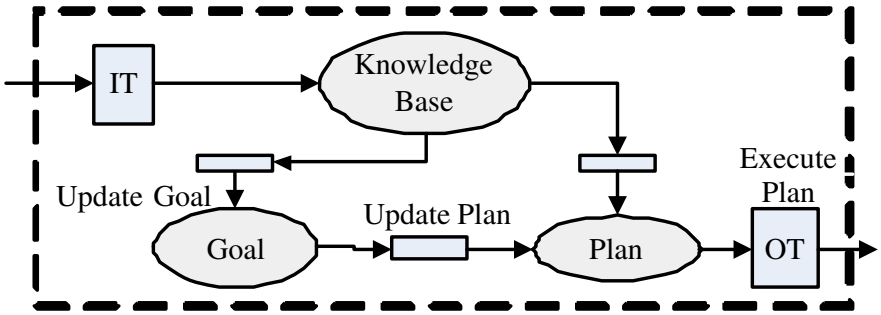
In questa implementazione l'agente ha una struttura semplice e minimalista che si rifà ad una struttura *feed-back control loop*. In tal modo si ottiene una implementazione robusta ed efficiente anche se non abbastanza flessibile da poter gestire più compiti a parte quelli che richiedo una singola decisione o azione da parte dell'agente.



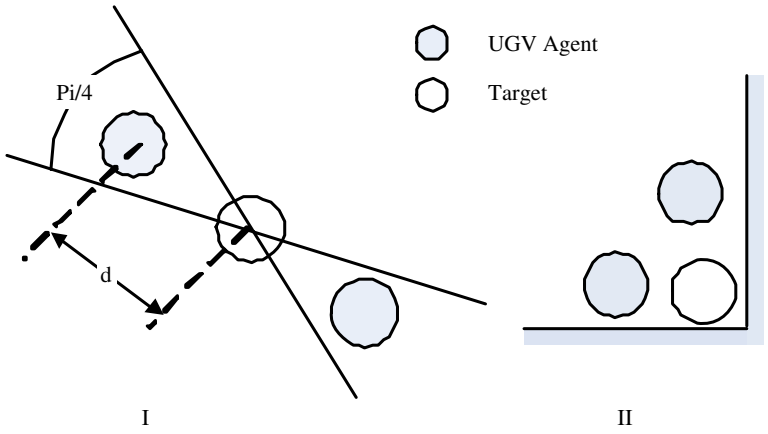
L'altra implementazione invece realizza un controllo autonomo ispirato ad un modello decisionale umano (nello specifico basato su OODA, *observe-orient-decide-act*). Questa modellazione permette di avere un sistema scalabile ed estendibile. Basta infatti modificare le giuste fasi aggiungendo le funzionalità richieste senza dover stravolgere la struttura. Inoltre questa modellazione permette un effettiva collaborazione in caso si abbiano team di UAV. Ad esempio il modulo ORIENT di un agente può rappresentare un ingresso per il modulo OBSERVE di tutti gli altri.



In [42] si prende invece in esame l'uso di *Unmanned Groud Vehicle* simulati in un caso di *pursuit-evasion war game*. In questa simulazione due agenti collaborano per la cattura di un nemico. Viene realizzato per il caso un modello di sistema multiagente basato su una rete di Petri ad oggetti. Anche in questo caso si tratta di agenti BDI e tramite l'uso delle reti di Petri ad oggetti è possibile fare un analisi formale del modello così da poter verificare le proprietà di questo. Ad esempio il modello astratto di un singolo agente UGV è descritto dalla seguente rete di Petri.



La comunicazione fra agenti è alla base per la coordinazione e la cooperazione e per sentire i cambiamenti ambientali. Gli agenti per comunicare hanno a disposizione un *supervisor* che coordina l'azione di ricerca del nemico. Un nemico si considera catturato nelle seguenti condizioni:

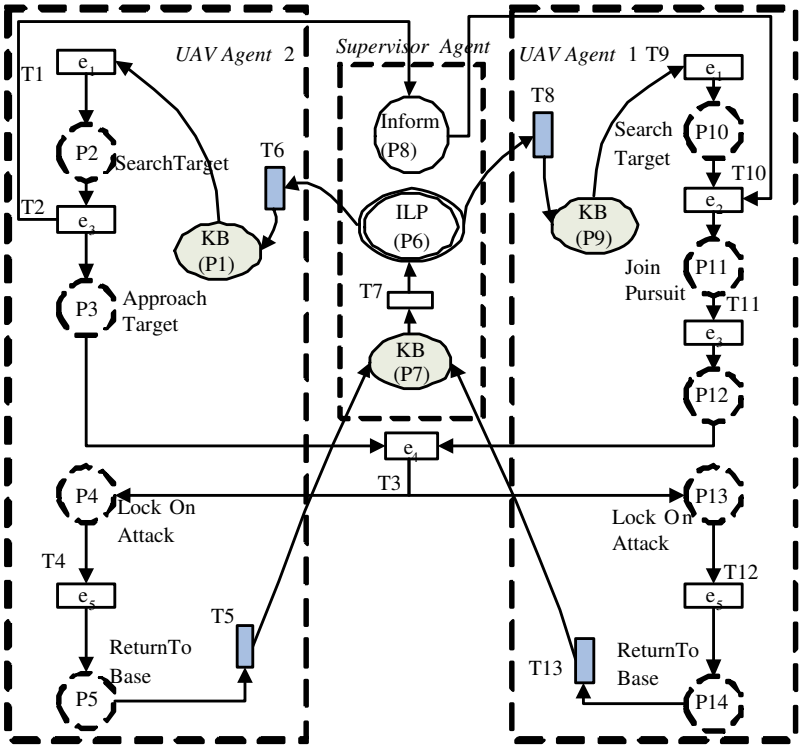


Gli agenti hanno a disposizione cinque azioni discrete che caratterizzano le seguenti azioni che possono effettuare per completare la missione e sono:

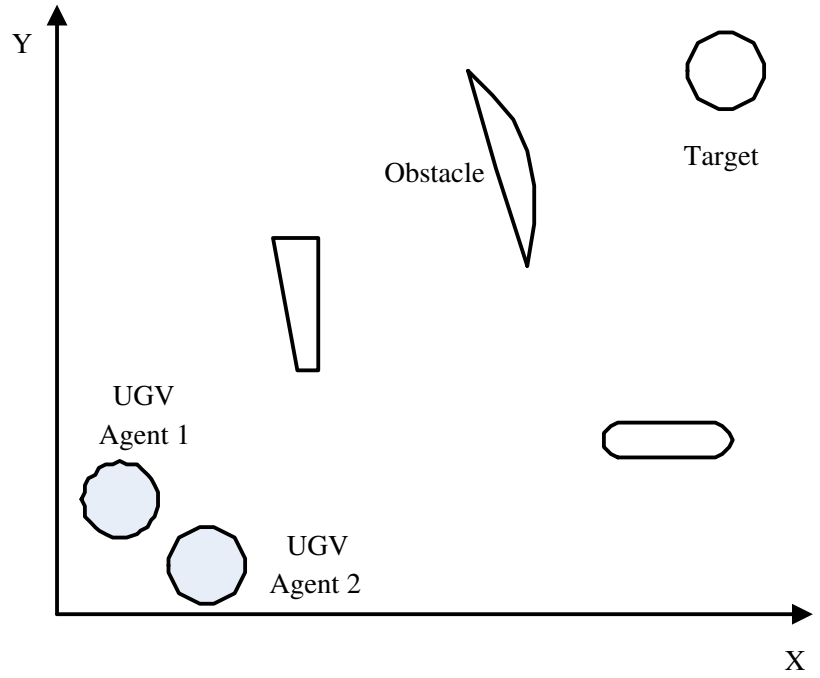
- e1: ricerca
- e2: obbiettivo trovato da un membro del team
- e3: obbiettivo in vista
- e4: obbiettivo abbastanza vicino per un attacco

- e5: ritorno alla base

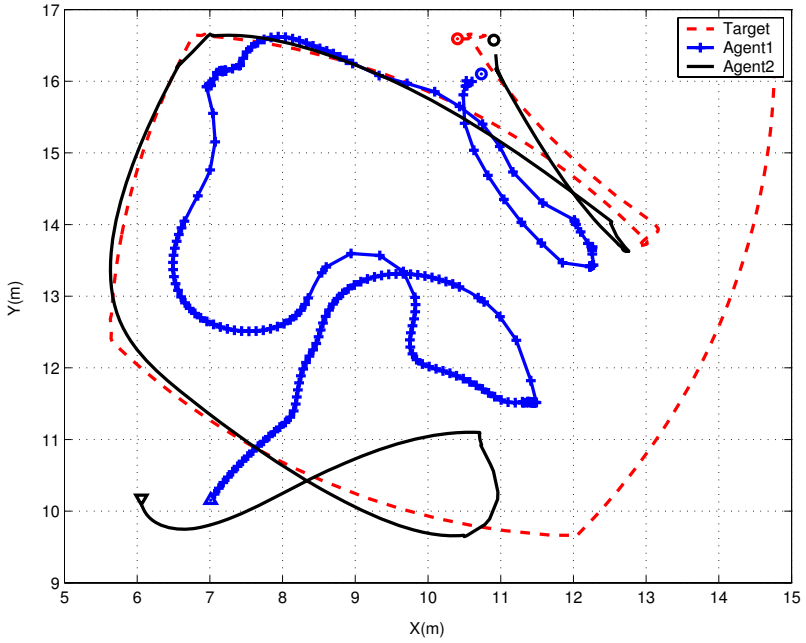
Date queste azioni il modello in cui l'agente due è il primo a prendere contatto con il nemico ed informa l'agente uno è quello che segue.



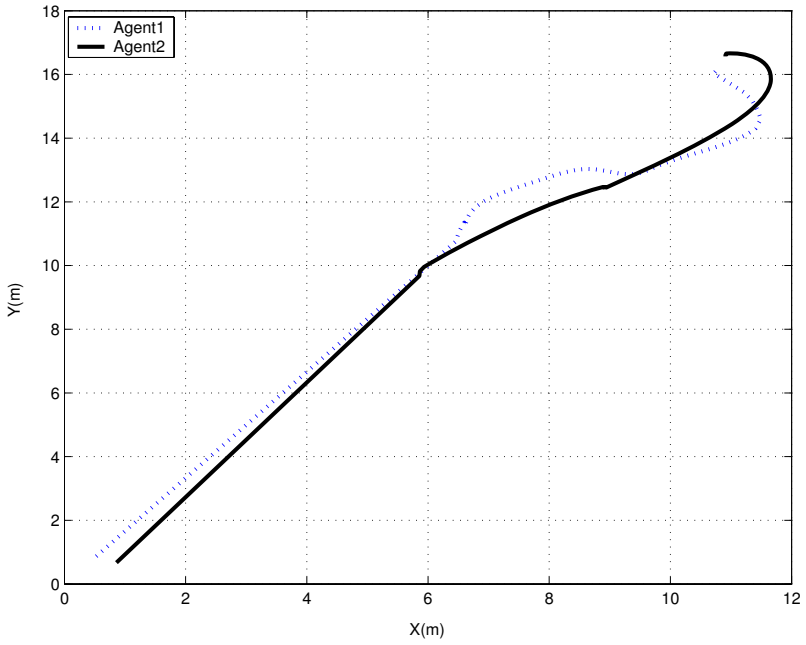
Tramite l'utilizzo di INA, Integrated Net Analyzer, le proprietà del modello sono state automaticamente analizzate. Il modello è *boundness* e *live*, inoltre il numero di stati raggiungibili è quarantaquattro. Nella simulazione a partire dalle peggiori condizioni iniziali.



Il sistema è riuscito a completare il suo obbiettivo e tornare poi alla base.



(a) Attack the target



(b) Return to Base

Chapter 4

Architettura a livelli

Nella ricerca continua per l'autonomia totale degli *Unmanned Vehicle*, bisogna porre attenzione a non dimenticarsi di astrarre correttamente i concetti, altrimenti si corre il rischio di ammassare insieme funzionalità troppo differenti fra loro, rendendo difficile l'evoluzione e la manutenzione del codice. Mantenere in un unico concetto idee diverse, come il controllo di un motore o la pianificazione del prossimo checkpoint da raggiungere, può portare allo sviluppo di un sistema molto complesso. Si corre inoltre il rischio di ritrovarsi a implementare più volte e, probabilmente in modalità differenti, lo stesso tipo di azioni o azioni molto simili fra loro. Per tale motivo in questa Tesi proponiamo una suddivisione a livelli dei compiti, che può essere usata come linea guida nello sviluppo di un sistema autonomo. Inoltre un'architettura a livelli permette di dividere i compiti fra diversi specialisti, che non solo possono quindi specializzarsi prescindendo dalle specifiche del sistema, ma possono coordinarsi nel lavoro in termini di interfacce e comportamenti attesi, parallelizzando il lavoro e riducendo quindi tempi e costi di sviluppo. In particolare il livello alto, il terzo livello, può cominciare a lavorare su algoritmi per lo sviluppo di comportamenti autonomi ipotizzando quali funzionalità ha "necessità" di riceve dai livelli più bassi, guidando quindi lo sviluppo in modo *goal oriented*. In attesa che queste funzionalità vengano implementate, gli sviluppatori possono comunque cominciare a studiare e verificare i risultati dei propri algoritmi tramite simulatori, riproducendo i comportamenti at-

tesi in modo fittizio. Ogni livello può dialogare con tutti i livelli sottostanti, ma un'implementazione ben sviluppata dovrebbe sollecitare lo sviluppatore a non scendere oltre il livello sottostante, perché in questo sono presenti tutte le funzionalità di cui può aver bisogno.

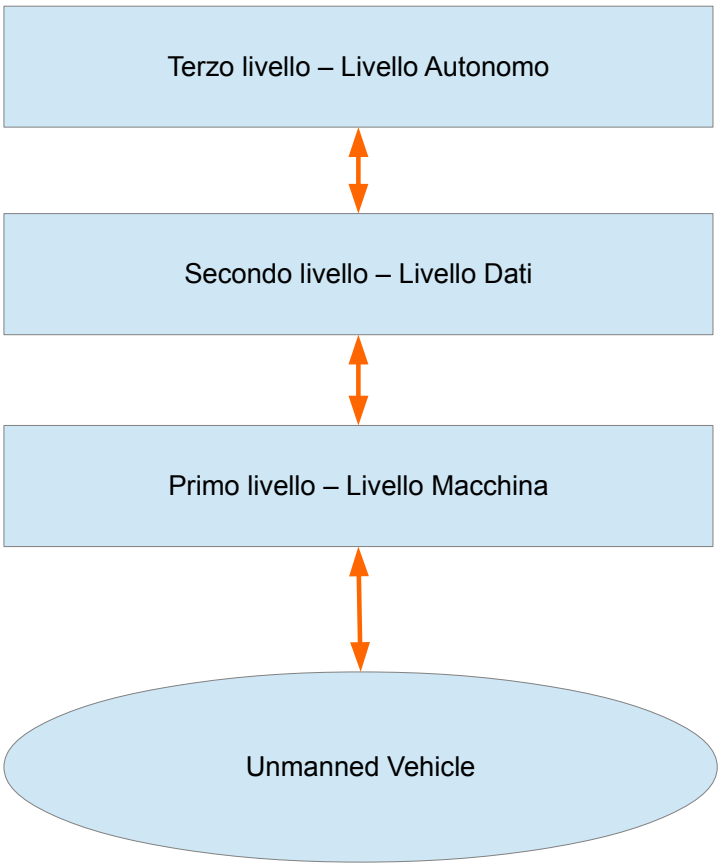


Figure 4.1: Rappresentazione grafica di una architettura a livelli

4.1 Primo livello - Livello Macchina, interazione con il mezzo

Il primo livello è il livello più basso, qui è dove il codice è molto più condizionato dal hardware su cui si troverà a funzionare. Bisogna implementare tutto il codice di controllo del mezzo affinché questo possa essere utilizzato correttamente. Deve presentare all'utente una semplice interfaccia per interagire, dalla quale impartire i comandi base di azionamento degli attuatori e semplificare la lettura dei suoi sensori di bordo. Dettagli come tensioni operative, frequenze di clock, tempi di attesa, devono essere nascosti all'utilizzatore che deve poter vedere la periferica come un dispositivo generale di una certa categoria di componenti, quindi con un determinato comportamento atteso. Nel mondo dell'openSource vi sono piattaforme molto avanzate che potrebbero tranquillamente adempiere ai compiti previsti da questo livello, con la necessità forse di qualche piccola modifica. Per i MAV, *Micro Air Vehicle*, in particolare due piattaforme openSource spiccano su tutte le altre, [43] e OpenPilot [44]. Ve ne sono però tante altre come consultabile dalla pratica tabella realizzata su oddcopter.com [45]. Queste piattaforme offrono tutto il necessario, per realizzare economici UAV artigianali. La recente integrazione di queste unità con embedded PC permetterà di sviluppare le interfacce di basso livello in modo via via più agevole, aprendo le porte a molto del software già realizzato per le soluzioni desktop.

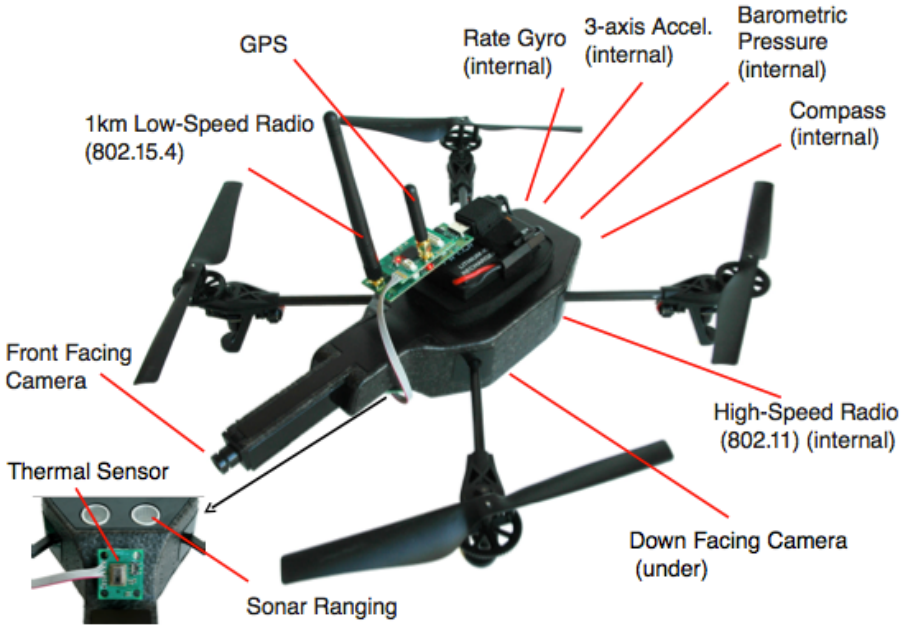


Figure 4.2: Esempio di alcune componenti di basso livello a cui si accede in questo livello.

4.2 Secondo livello - Livello Dati, elaborazione dati

Il secondo livello ha il compito di trattare i dati forniti dal primo livello per consegnare al terzo livello, o all'utente, informazioni utili per prendere decisioni. Inoltre offre un'interfaccia per eseguire azioni di alto livello, svincolando dalla necessità di conoscere come effettuare quelle date azioni su differenti tipologie di veicoli, aerei piuttosto che marittimi. Molto oggi è già disponibile, come la navigazione per waypoint, l'esecuzione di decollo e atterraggio e il *return to home* in caso di difficoltà. Questo ha già aperto le porte del mondo degli *Unmanned Vehicle* ad una comunità molto più vasta, fatta non solo di professionisti e ricercatori, ma anche di amatori e nuove leve. Anche i piloti più inesperti possono affidarsi al supporto di queste funzionalità per svolgere in sicurezza le azioni più complesse. Ad oggi infatti quasi tutte

le unità di controllo che offrono questo genere di assistenza di alto livello vengono chiamate anche copilot, proprio ad evidenziare l'aiuto che danno al pilota nel controllo del mezzo. La già citata piattaforma ArduPilot offre uno dei più vasti campionari di funzionalità, molte per UAV, ma alcune disponibili anche per UGV,USV,UUV. Tra i supporti implementati quelli di maggior rilievo risultano: *Gyro Stabilization*, che aiuta a mantenere il mezzo in assetto stabile, *Self Leveling*, che mantiene il multicoptero in posizione orizzontale quando non soggetto a comandi di pitch o roll, "Care Free", per pilotare il multicoptero come se puntasse nella direzione originale anche se l'orientamento è cambiato, *Altitude Hold*, per mantenere in modo autonomo una predeterminata quota dal suolo, *Position Hold*, per stazionare in una predeterminata posizione, *Return Home*, per riportare il mezzo autonomamente alla posizione di decollo e *Waypoint Navigation*, che permette di definire un piano di volo che il mezzo autonomamente seguirà.

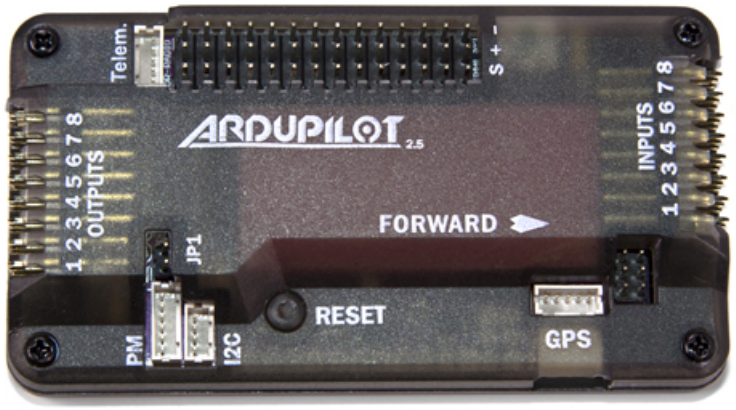


Figure 4.3: L'ultimo prodotto della 3DRobotics, la APM 2.6, che integra a bordo il software di controllo ArduPilot

4.3 Terzo livello - Livello Autonomo, formulazione strategie autonome

Il terzo livello è quello che partendo dalle informazioni messe a disposizione dal secondo livello può, tramite algoritmi di intelligenza artificiale, effettuare compiti di *task resolution* o *problem solving*. Fanno parte di questa categoria di problemi ad esempio la scelta del percorso migliore fra diverse opzioni disponibili, calcolato tenendo in considerazione correnti d'aria o eventuali imprevisti avvenuti durante il corso della missione, la gestione di risorse limitate e condivise fra i diversi mezzi, come ad esempio un limitato numero di batterie che i mezzi possono automaticamente sostituire, la creazione di piani di lavoro per riuscire a portare a termine tutti i task nel miglior modo possibile, come il porre il mattone dopo aver steso il calcestruzzo nella costruzione di un edificio al fine di evitarne il crollo e quindi il conseguente fallimento dell'intero piano. Questi sono solo esempi della tipologia di problemi che questo livello ha l'obbiettivo di trattare e risolvere. Sono compiti dove i dettagli sul mezzo e il suo controllo sono superflui, dove si desiderano interazioni avanzate come i comandi "dirigiti verso una posizione P", "abbassati", "afferra l'elemento E" (senza neanche dovere descrivere dove l'elemento E possa trovarsi o com'è fatto), ecc.. messe a disposizione dal secondo livello o addirittura dal terzo livello stesso, creando azioni complesse come composizione di azioni già di per sè elaborate. Lo sviluppatore dovrebbe in questo livello essere libero e non costretto di seguire la particolare tecnologia implementativa, o un determinato paradigma, più adatto al risultato finale da raggiungere. Ad esempio come mostrato in precedenza comportamenti autonomi possono essere facili da modellare con sistemi multi-agente, soprattutto se azioni complesse possono considerarsi come conoscenza di base. Molte più applicazioni diventano possibili quando una moltitudine di veicoli può essere utilizzata fidandosi della sua capacità di coordinarsi e autonomamente portare a termine l'insieme di compiti assegnato alla squadra.

Chapter 5

Autolocalizzazione

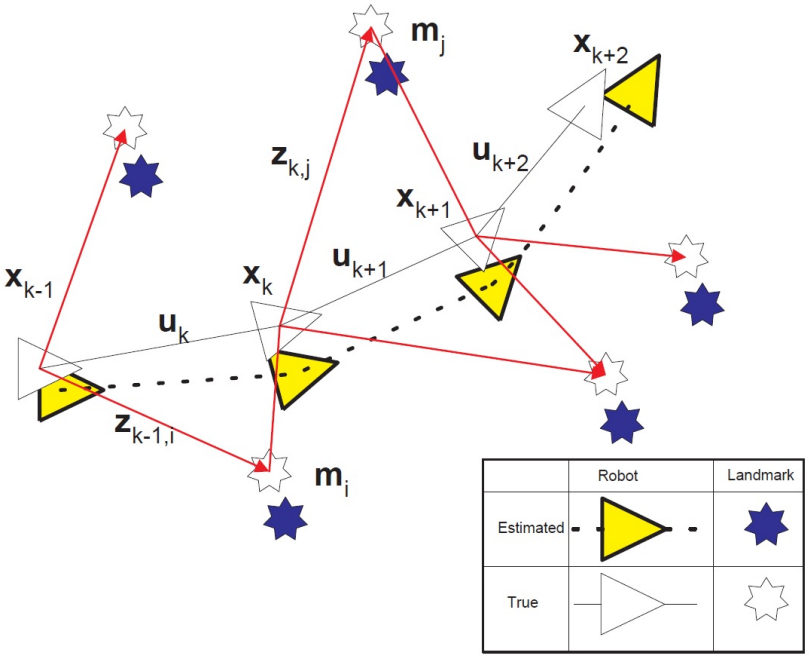
In questo capitolo analizzeremo i principali algoritmi di autolocalizzazione, partendo dal più generico SLAM per passare poi ai Visual SLAM, basati su un'autolocalizzazione visiva. In seguito presenteremo nel dettaglio il PTAM, l'algoritmo scelto per l'implementazione di questa Tesi.

5.1 SLAM

Il termine SLAM sta per *Simultaneous Localizzation and Mapping* ed indica l'insieme di tecniche atte a permettere a robot e veicoli autonomi la creazione di una mappa di un ambiente sconosciuto o l'aggiornamento di una in possesso e nello stesso tempo tenere traccia della propria posizione all'interno di quell'ambiente. I due aspetti del problema, la localizzazione e il mapping, possono essere mappati su due processi separati, e la continua iterazione permette, grazie ai risultati ottenuti da uno, di migliorare i risultati ottenuti dall'altro al passo successivo. Il mapping deve essere in grado di correlare le informazioni che arrivano dai suoi sensori con le informazioni già in suo possesso e costruire un modello consistente dell'ambiente. La localizzazione cerca invece di posizionarci all'interno di questa mappa e possibilmente di darci anche informazioni sull'assetto. I due problemi, come intuibile, sono strettamente correlati e le soluzioni possono migliorare in modo incrementale solo tramite un'interazione costante. I primi riferimenti storici

a questo tipo di problema risalgono al 1986 quando esso è stato presentato all'IEEE Robotics and Automation Conference di San Francisco. Già negli anni successivi una serie di paper fondamentali [46, 47] ha stabilito la base statistica per descrivere le relazioni fra i landmark e l'elaborazione delle incertezze geometriche. Un elemento chiave di questi lavori è stato evidenziare che deve esserci un alto grado di correlazione fra le stime delle posizioni dei diversi landmark nella mappa. La correlazione tende poi a crescere mano a mano che successive osservazioni vengono effettuate. Il paper [48] mostra come un robot che si muove in un ambiente sconosciuto, effettuando osservazioni dei landmark relative alla sua posizione, ottiene una stima delle posizioni di questi landmark che sono necessariamente tutte correlate a causa dell'errore comune nella valutazione della posizione del veicolo. Una soluzione consistente del problema di localizzazione e mapping richiede uno stato composto dalla posa del veicolo e dalle posizioni dei landmark, da aggiornare ad ogni osservazione. Questo richiede di mantenere un vettore di stato, con dimensione pari al numero di landmark della mappa, che richiede un costo computazionale di aggiornamento pari al quadrato del numero dei landmark. Questo lavoro non ha però guardato alla convergenza delle proprietà della mappa in situazioni stazionarie. All'epoca, è stato assunto che l'errore stimato non sarebbe mai confluito, ma anzi avrebbe esibito un comportamento casuale nella sua crescita illimitata. Dato il costo computazionale elevato e senza conoscere il comportamento convergente della mappa, la ricerca gradualmente si è concentrata solo su approssimazioni del problema finché non si è giunti ad uno stop total. Eventuali lavori si concentravano sul problema della localizzazione o del mapping separatamente. Quando è stato compreso che la combinazione della localizzazione con il mapping, formulati come un singolo problema di stima, convergeva la ricerca ha trovato nuova linfa. É stato riscontrato che la correlazione fra i landmark, che molti ricercatori avevano cercato di minimizzare, era il punto critico del problema, e che al contrario più questa correlazione cresceva, migliore era la soluzione. Ufficialmente l'acronimo SLAM è stato coniato nel paper [49] presentato nel 1995 all'International Symposium on Robotics Research, dove si descrive anche la struttura del problema e la convergenza dei risultati. Da qui in poi i lavori si

sono concentrati molto sull'incrementare l'efficienza computazionale e risolvere i problemi legati all'associazione dei dati e al "loop closure", riconoscere percorsi chiusi. Nel 1999 all'International Symposium on Robotics Research (ISRR'99) si è svolto un importante meeting, dove si è tenuta la prima sessione dedicata allo SLAM. Nel 2000 il IEEE ICRA Workshop on SLAM ha attratto quindici ricercatori, che si sono concentrati su problemi come la complessità computazionale, l'associazione dei dati e le sfide implementative. Nel 2002 all'ICRA i ricercatori erano già divenuti 150, ed avevano portato tanti sviluppi interessanti ed applicazioni. Ancora oggi l'interesse verso gli algoritmi SLAM vede una crescita esponenziale, probabilmente dovuta anche alla diffusione di robot e mezzi autonomi a costi sempre più accessibili e dotati di sensori sempre più avanzati.
Se da un punto di vista teorico il problema dello SLAM si può dire risolto, non mancano le complicazioni da un punto di vista implementativo. Alla complessità aggiunta dal movimento del veicolo vi è infatti da sommare anche l'incertezza dei dati generati dai nostri sensori che non hanno una accuratezza assoluta e possono essere anche imprecisi. Lo stesso ambiente inoltre può generare errori o può avere variazioni dinamiche. Tutte queste variabili possono incrementare l'errore e l'inaccuratezza del nostro mapping e della nostra localizzazione. Ci sono varie tecniche fortunatamente che possono essere messe in campo per compensare l'errore, come riconoscere delle caratteristiche già osservate e rimodellare la mappa per far si che due caratteristiche diventino di fatto una. Tecniche statistiche sono usate in SLAM, come il filtro di Kalman o filtri particellari. Permettono di avere una stima sulla probabilità a posteriori della posa e dei parametri della mappa. Immaginiamo di avere un veicolo autonomo capace di muoversi per l'ambiente e tramite un sensore leggere la posizione dei landmark.



x_k rappresenta la posizione e direzione del veicolo, u_k il vettore applicato al tempo k-1 per guidare il veicolo allo stato x_k. m_i è la posizione reale del landmark i-esimo ed infine z_{ik} rappresenta l'osservazione effettuata dal veicolo sulla posizione del landmark i-esimo al tempo k. Com'è possibile vedere gran parte dell'errore nel posizionamento è dovuto all'errore nella lettura delle posizioni dei landmark, probabilmente a causa di una incertezza costante nel sensore. Affidandosi a questa sola informazione si tenderebbe a divergere. Nel Probabilistic SLAM, ad ogni istante k ci viene richiesto di calcolare la distribuzione di probabilità della seguente:

$$P(\mathbf{x}_k, \mathbf{m} \mid \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$$

Questa distribuzione di probabilità descrive la densità di probabilità a posteriori delle posizioni dei landmark e dello stato del veicolo all'istante k, date le osservazioni precedenti, i comandi impartiti e lo stato iniziale. Una soluzione iterativa è ottenibile data la distribuzione per un istante temporale k-1, un controllo u_k ed una osservazione z_k computando tramite il teorema

di Bayes. Per effettuare questa computazione è richiesto però la definizione di uno modello di movimento e un modello di osservazione che descrivano gli effetti di un controllo sul veicolo e l'osservazione dei landmark. Questi sono descritti come segue:

$$P(\mathbf{z}_k \mid \mathbf{x}_k, \mathbf{m}).$$

Figure 5.1: Observation model

$$P(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_k)$$

Figure 5.2: Motion model

L'algoritmo SLAM è implementato tramite una sequenza ricorsiva a due passaggi con predizione della posizione e correzione della misura, date da:

Time-update

$$\begin{aligned} &P(\mathbf{x}_k, \mathbf{m} \mid \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0) \\ &= \int P(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_k) \\ &\times P(\mathbf{x}_{k-1}, \mathbf{m} \mid \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}, \mathbf{x}_0) \mathrm{d}\mathbf{x}_{k-1} \end{aligned}$$

Measurement Update

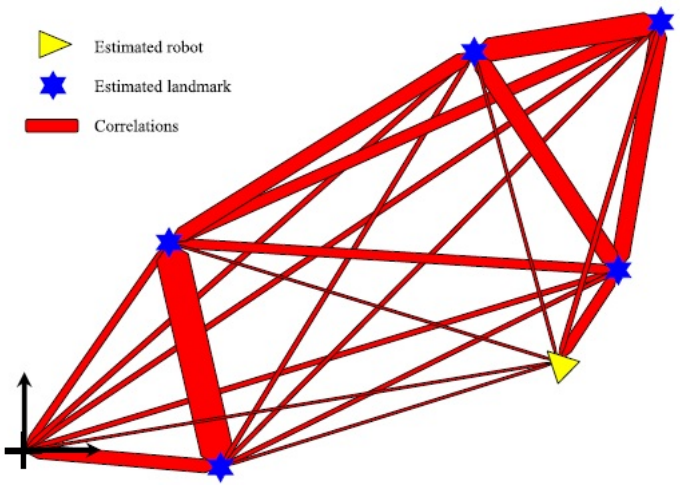
$$\begin{aligned} &P(\mathbf{x}_k, \mathbf{m} \mid \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \\ &= \frac{P(\mathbf{z}_k \mid \mathbf{x}_k, \mathbf{m}) P(\mathbf{x}_k, \mathbf{m} \mid \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{P(\mathbf{z}_k \mid \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \end{aligned}$$

Vale la pena notare che per realizzare la mappa bisogna ora calcolare la densità condizionale $P(\mathbf{m} \mid \mathbf{X}_{0:k}, \mathbf{Z}_{0:k}, \mathbf{U}_{0:k})$. Questo assume che la posizione del veicolo x_k sia conosciuta (o perlomeno deterministica) in tutti gli istanti temporali k, soggetti alla conoscenza della posizione iniziale. Una mappa m è costruita fondendo le osservazioni da diverse posizioni. Viceversa la localizzazione richiede di calcolare la distribuzione di probabilità $P(\mathbf{x}_k \mid \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{m})$ e questo assume la conoscenza esatta della posizione dei landmark. L' "Observation model" mostra evidentemente che la probabilità

a posteriori dei landmark e dello stato del veicolo non può essere calcolata tramite la suddivisione dei problemi.

$$P(\mathbf{x}_k, \mathbf{m} \mid \mathbf{z}_k) \neq P(\mathbf{x}_k \mid \mathbf{z}_k) P(\mathbf{m} \mid \mathbf{z}_k)$$

Il problema SLAM ha però una struttura più complessa di quella ipotizzabile dalle sole equazioni. Riferendosi alla figura infatti si può notare come l'errore tra la posizione esatta del landmark e quella stimata è comune fra i landmark ed è dovuta alla sorgente. Questo implica che gli errori nelle posizioni stimate dei landmark sono molto correlati. Praticamente questo significa che la posizione fra due landmark può essere conosciuta con grande precisione, anche se la posizione del singolo è piuttosto incerta. Inoltre la correlazione fra le stime sui landmark aumenta in modo monotono ad ogni osservazione, aumentando di fatto la conoscenza sulle loro posizioni relative. Mano a mano che il veicolo effettua osservazioni sul mondo arricchisce la sua conoscenza dello stesso e rafforza le correlazioni fra i punti, creandone di nuove fintanto che nuovi landmark vengono aggiunti fino a raggiungere una vera e propria rete immaginabile come la figura sottostante.



Ad ogni osservazione lo spessore del collegamento aumenta, l'effetto maggiore si ha sui vicini, soprattutto se già molto correlati, e diminuisce se ci si

allontana dal vicinato. Durante la costruzione della mappa anche la localizzazione del veicolo migliora e teoricamente il limite è la capacità di autolocalizzarsi in modo accurato su una mappa costruita come se fosse stata data come conoscenza pregressa. Resta da definire quale implementazione usare come "observation model" e come "motion model" per ottenere un efficiente e consistente calcolo delle distribuzioni a priori e posteriori della "Time Update" e della "Measurament Update". Le più importanti sono EKF-SLAM e FastSLAM, anche se ovviamente non sono le uniche. EKF-SLAM prevede l'utilizzo di un modello stato-spazio con l'aggiunta di rumore Gaussiano, portando all'utilizzo di un filtro di Kalman esteso per risolvere il problema SLAM. FastSlam invece descrive il "motion model" come un insieme di campioni di una più generale distribuzione di probabilità non Gaussiana. Questo porta all'uso di un filtro particellare di Rao-Blackwellised per risolvere il problema. Non ci soffermeremo però su questi dettagli implementativi in quanto non è lo scopo di questa Tesi.

Bundle adjustment

Il bundle adjustment è un algoritmo che permette di stimare, da un insieme di immagini, una ricostruzione proiettiva o affine [50]. Consideriamo che da ogni immagine estraiamo dei punti x_j^i. Ciò che vogliamo ottenere è la matrice P^i che trasforma i punti dello spazio X_j in punti immagine. Purtroppo le misurazioni di questi punti sono affette da rumore e la relazione $x_j^i = P^i X_j$ non è completamente soddisfatta. Vogliamo quindi stimare solo la matrice di proiezione $\acute{P}^i$ che minimizza la distanza fra i punti riproiettati e quelli effettivamente misurati. Per effettuare questa stima bisogna quindi minimizzare l'errore di riproiezione tramite un processo iterativo.

$$\min \sum d(\acute{P}^i \acute{X}_j, x_j^i)^2$$

Questo in pratica è il bundle adjustment. Si tratta di correggere un fascio di raggi fra il centro della camera ed un insieme di punti tridimensionali. Questo algoritmo ha il pregio di fornire anche valori di covarianza per ogni

misurazione. Il problema è che richiede una buona inizializzazione e rischia di diventare un problema di minimizzazione estremamente pesante.

5.1.1 Vision-based SLAM

Il V-SLAM è una categoria di implementazioni dello SLAM dove il sensore per monitorare l'ambiente circostante è la camera. Da questa se ne cerca di ricavare informazioni sull'ambiente circostante. Si possono utilizzare una o più camere e mettere in campo diverse tecniche per il riconoscimento della posizione dei landmark dell'ambiente. Da tecniche SfM, *structure from motion*, che cercano di calcolarla analizzando le trasformazioni avvenute all'ambiente a seguito di un movimento, fino all'utilizzo di sistemi di fotocamera stereo preventivamente calibrati. Recentemente si usano anche camere RGBD che aggiungono ai tre parametri RGB per ogni pixel, che rappresentano il colore (Red, Green, Blue), un quarto parametro D (Depth) a rappresentare la profondità del pixel. In [51] viene presentata una implementazione dell'allora (2005) novello algoritmo. In questo paper si utilizza la telecamera per creare, e tracciare landmark nell'ambiente e se ne dimostra la solidità a variazioni ambientali, come un cambio di luce, e il ridotto costo implementativo. Inoltre i dati ottenuti dalla(e) telecamera(e) sono fusi con il sistema odometrico del veicolo, ed è stata realizzata una struttura a due livelli, un front-end e un back-end.

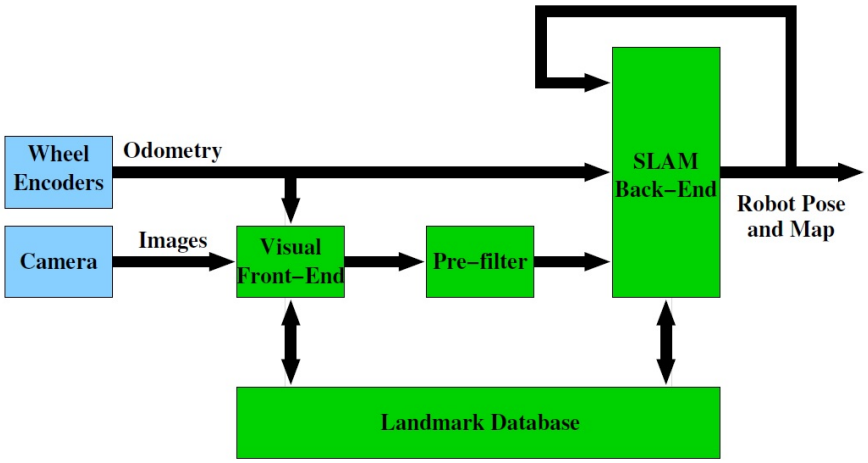


Figure 5.3: Architettura proposta da Karlsson & Co. nel loro paper sul vSLAM.

Nel 2007 Davison & Co. presentano nel loro paper [52] una implementazione di V-SLAM dove la telecamera viene usata per generare una mappa tridimensionale probabilistica in real-time tramite un flusso video di 30 fps. Il tutto senza sfruttare architetture multi-core e con tempi medi di 19ms per elaborare un frame su un Intel Pentium M a 1.6 GHz.

5.2 PTAM

Il PTAM, *Parallel Tracking and Mapping*, è un algoritmo SLAM basto su keyframe, presentato la prima volta in [53]. E' stato uno dei primi algoritmi di Monocular SLAM ad effettuare un tracking della camera senza nessun mapping disponibile come prerequisito. Può effettuare il tracking di una camera in un ambiente completamente sconosciuto, dove non vi è conoscenza degli oggetti che lo riempiono, e nel frattempo crearsi internamente una mappa di quest'ultimo. Si prefigge lo scopo di riuscire a creare una mappa in modo rapido, preciso ed accurato, aggiornandola mano a mano che nuove scene vengono presentate. Per adempiere al compito si è prefissato il vincolo che le ambientazioni fosserò però statiche e piccole. La metodica applicata si può suddividere nei seguenti punti:

- suddividere il tracking e il mapping in due processi paralleli e separati, così che il tracking non sia più probabilisticamente correlato al mapping e possa usare qualsiasi metodo robusto si voglia
- realizzare il mapping tramite l'acquisizione di keyframe, che non devono più essere la totalità dei frame acquisiti ma solo frame importanti, scene significative, che vengono processate con tecniche batch come il bundle adjustment, grazie alla non necessità di rispettare stretti vincoli temporali dovuti al tracking
- la mappa è inizializzata da un'alta densità di punti, acqusiti da una coppia di immagini stereo
- nuovi punti sono aggiunti con una ricerca epipolare
- si cerca di inserire nella mappa migliaia di questi punti

Descriveremo i due processi, tracking e mapping, separatamente e alcune considerazioni generali sull'algortimo tratte dalla nostra esperienza.

Tracking

Il tracking in PTAM è effettuato su tutti i frame acquisiti dalla camera, e si basa su una procedura a due stadi, dando come prerequisito l'esistenza di una mappa dell'ambiente in memoria (generata dall'inizializzazione stereo). Il primo stadio consiste nell'effettuare una prima considerazione sul cambio della posa della camera utilizzando un modello di decadimento della velocità simile al modello di velocità con costanti alfa-beta. Sulla base di questo cambiamento i punti della mappa sono riproiettati sull'immagine. Un piccolo numero di grossolane caratteristiche viene poi cercato, con un raffinamento al subpixel, sul livello più alto della piramide generata dall'immagine. Piramide composta da quattro livelli dove ognuno ha risoluzione dimezzata rispetto al precedente livello. Sulla base di questa ricerca la posa è aggiornata. In seguito un più alto numero di punti è riproiettato e cercato nell'immagine più dettagliata, utilizzando un raffinamento subpixel stavolta solo per un sottoinsieme di questi punti. Alla fine da queste due ricerche la posa viene ricalcolata

tenendo in considerazione tutte le corrispondenze. Il modello utilizzato per la proiezione dei punti è il FOV-model presentato da Devernay e Faugeras [54]. É un sistema di calibrazione molto efficiente e prestazionale, se paragonato ad altri modelli, che permette di correggere le distorsioni dell'ottica con un buon risultato basandosi su soli cinque parametri. Due per indicare la lunghezza focale, due per il punto principale ed uno ad indicare la distorsione radiale. L'aggiornamento della posa viene calcolando iterativamente cercando di minimizzare una robusta funzione obbiettivo della riproiezione dell'errore:

$$\boldsymbol{\mu}' = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \sum_{j \in S} \operatorname{Obj}\left(\frac{|e_j|}{\sigma_j}, \sigma_T\right)$$

Dove μ rappresenta un vettore di sei elementi, utilizzato come parametri della funzione esponenziale $\exp(\mu)$ che sostituisce la matrice M 4x4 nel calcolo della nuova matrice E_{cw} (usata per la trasformazione dei punti da coordinate immagini a coordinate camera).

$$E'_{\mathcal{CW}} = M E_{\mathcal{CW}} = \exp(\boldsymbol{\mu}) E_{\mathcal{CW}}$$

e_j rappresenta il vettore di errore di riproiezione ed è calcolato tramite:

$$\boldsymbol{e_j} = \begin{pmatrix} \hat{u}_j \\ \hat{v}_j \end{pmatrix} - \operatorname{CamProj}\left(\exp\left(\boldsymbol{\mu}\right) E_{\mathcal{CW}} \boldsymbol{p_j}\right).$$

Infine Obj(.,σ_T) è la funzione obbiettivo bipeso con σ_T che rappresenta una robusta stima della valore della derivazione standard della distribuzione derivato da tutti i residui. CamProj() è la funzione che proietta i punti dalla camera all'immagine e può essere definita come segue: (u_i , v_i)^T = CamProj ($E_{cw} p_i w$). Come detto sopra si utilizza una funzione di proiezione per camere pin-hole con il supporto di lenti che esibiscono una distorsione radiale a barile. La funzione diviene:

$$\text{CamProj}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} + \begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix} \frac{\mathrm{r}'}{\mathrm{r}} \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}$$

$$r = \sqrt{\frac{x^2 + y^2}{z^2}}$$

$$r' = \frac{1}{\omega} \arctan(2r \tan \frac{\omega}{2})$$

(f_u , f_v) sono i parametri della lunghezza focale, (u_0, v_0) quelli del punto principale e ω è la distorsione radiale. La qualità del tracking è determinata da due soglie, calcolate con il rapporto fra le caratteristiche correttamente osservate e le totali. Dopo essere scesi sotto la prima soglia il tracking viene considerato scadente e non vengono generati keyframe da consegnare al mapping perché considerati poco affidabili, anche se il tracking continua. Se il valore calcolato continua a scendere e va oltre una seconda soglia allora il tracking si considera perso e bisogna procedere ad una fase di recupero. Nell'implementazione concreta dell'algoritmo per evitare *jitter* il tracking a due stadi viene temporaneamente disabilitato qualora si considerasse la camera quasi stazionaria. Disattivando la modalità più grossolana e lasciando attiva solo quella basata sull'immagine a piena risoluzione si risparmia una parte di calcoli portando vantaggi alla stabilità del sistema.

Mapping

Il mapping può essere suddiviso in due fasi, una è l'inizializzazione che viene effettuata tramite una coppia di immagini stereo e l'altra riguarda l'aggiornamento della mappa durante il normale utilizzo. L'intera struttura può essere descritta dalla figura sottostante:

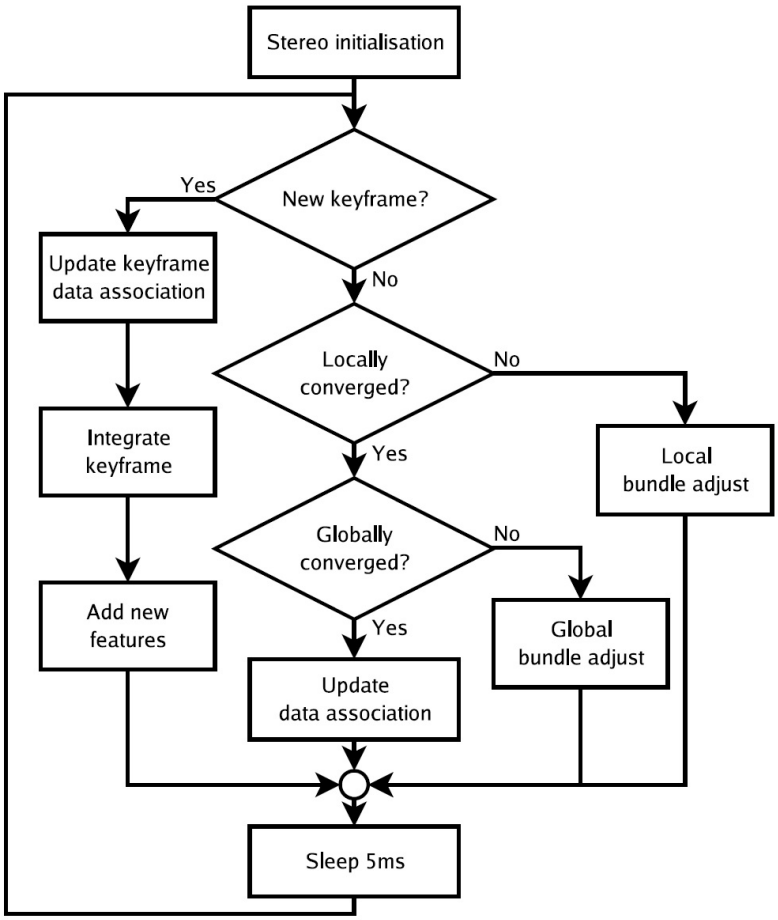


Figure 5.4: Struttura schematica del mapping del PTAM.

Per l'inizializzazione stereo si richiede la collaborazione dell'utente. Dopo la pressione di un comando sulla tastiera inizia la fase di acquisizione e la prima immagine acquisita viene salvata come keyframe. Vengono disegnate a monitor 1000 linee sulla base di caratteristiche rivelate tramite l'algoritmo FAST per aiutare l'utente. Esso deve lentamente traslare la camera, e legger-

mente ruotarla. Infine una seconda pressione permette di salvare un secondo keyframe e terminare l'operazione. Dalla traslazione delle caratteristiche tracciate l'algoritmo stereo a cinque punti e RANSAC possono stimare la matrice essenziale e triangolare la mappa base. Un bundle adjustment [55] viene infine effettuato per completare l'operazione. Dopo l'inizializzazione inizia la vera e propria fase di arricchimento della mappa. Nuovi keyframe vengono aggiunti in memoria a patto di rispettare dei vincoli inseriti per evitare una sovrabbondanza di informazioni ridondanti. Il tracking deve essere di buona qualità, il nuovo keyframe non può essere acquisito prima di altri venti frame rispetto all'ultimo keyframe acquisito e la distanza dal punto più vicino già presente in mappa deve super una soglia prestabilita. Se tutte queste condizioni sono rispettate un nuovo keyframe è aggiunto in memoria. Ovviamente questo keyframe porta con sé le informazioni già calcolate dal tracker quindi il mapper cerca quelle caratteristiche sfuggite effettuando la riproiezione dei punti ed aggiornando le misurazioni. Il tracker ha anche già trovato i punti significativi nell'immagine grazie all'algoritmo FAST, in tutti i livelli della piramide di immagini. In fase di mapping viene quindi eseguita una soppressione dei non massimi tramite l'algoritmo di Shi-Tomasi [56]. Aggiungere nuovi punti alla mappa richiede però informazioni non ricavabili tramite un solo keyframe. Queste informazioni vengono calcolate eseguendo una triangolazione con il keyframe più vicino disponibile in memoria. Le corrispondenze fra i due keyframe sono stabilite usando una ricerca epipolare fra le caratteristiche dei due keyframe allo stesso livello della piramide di immagini. Per ottimizzare la ricerca ed evitare che stalli in una ricerca infinita vengono usate delle euristiche: si fanno ipotesi iniziali sulla possibile profondità in cui cercare il nuovo punto candidato, sulla base della distribuzione della profondità dei punti in memoria nel nuovo keyframe. Se un match viene trovato il punto candidato può divenire a tutti gli effetti un nuovo punto della mappa. Quando non vi sono nuovi keyframe da aggiungere, si raffinano le informazioni in proprio possesso effettuando operazioni di bundle adjustment fra i keyframe in memoria, dove si minimizza una robusta funzione obiettivo.

$$\{\{\mu_2..\mu_N\},\{p'_1..p'_M\}\} = \underset{\{\{\mu\},\{p\}\}}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j \in S_i} \mathrm{Obj}\left(\frac{|e_{ji}|}{\sigma_{ji}}, \sigma_T\right)$$

Poiché si tratta di un'operazione che non scala bene con il numero di keyframe in memoria (O(N^3)), si è deciso di suddividerla in due operazioni, una locale ed una globale. La differenza principale è che l'operazione di bundle adjustment locale avviene solo sugli ultimi sette keyframe acquisiti. Quando tutte le operazioni locali sono concluse si procede poi ad un bundle adjustment globale. In ogni caso, che si stia calcolando il bundle adjustment locale o globale, all'arrivo di un nuovo keyframe il lavoro viene interroto per poter computare i nuovi punti da aggiungere alla mappa. Invece se tutte le operazioni di bundle adjustment sono portate a termine, avendo tempo libero a disposizione il mapper comincia a rielaborare i vecchi keyframe alla ricerca di nuove caratteristiche da mappare e aggiungere. Come attività meno prioritaria vi è invece il ricalcolo delle caratteristiche dei punti considerati outlier, perché giacevano nella zona a peso zero dello stimatore di Tukey. Se comunque, nella fase di ricalcolo, dovessero essere riconfermati come outlier verrebbero definitivamente esclusi, in caso contrario verranno aggiunti alla mappa.

Considerazioni personali

Abbiamo compilato ed eseguito l'algoritmo sotto ambiente Windows, ed è stato utilizzato con una videocamera non in grado di generare un flusso video con un frame rate superiore o uguale a 30 fps. Questo è uno dei requisiti minimi consigliati dagli sviluppatori. Ciò nonostante grazie ad uno specifico fix per Windows, proposto anche agli sviluppatori principali sulla pagina GitHub del progetto, e con la calibrazione dei parametri al fine di sfruttare la maggior potenza offerta dalla macchina da noi usata rispetto a quella usata in [53] siamo riusciti a migliorarne leggermente le prestazioni, anche se distanti da quelle viste nei video allegati. Un grande miglioramento si è invece avuto dalla sostituzione degli stimatori del bundle adjustment e

del tracker da Tukey a Cauchy. Una diversa implementazione di PTAM è disponibile openSource, si tratta di una implementazione dall'Autonoumus System Lab dell'ETH e disponibile sulla pagina GitHub al seguente indirizzo: https://github.com/ethz-asl/ethzasl_ptam. Questa versione è stata modificata, per essere compatibile con ROS, ed avere prestazioni migliori su macchine più deboli. Viene dichiarata anche una maggior robustezza in outdoor [57, 58, 59]. Questa è ottenuta dall'integrazione dei dati ottenuti dal PTAM con quelli ottenuti dall'IMU. Il nostro obbiettivo è però diverso, vogliamo rafforzare le informazioni ottenute dall'autolocalizzazione visiva fino a renderle affidabili quanto, se non di più, quelle ottenute dall'IMU.

Chapter 6

Caso di studio

In questa Tesi ci siamo prefissi l'obbiettivo di preparare un infrastruttura hardware/software per la sperimentazione di algoritmi di autolocalizzazione su UAV, il PTAM è l'algoritmo scelto nello specifico per questa Tesi. L'obbiettivo era di sperimentare la robustezza e l'accuratezza di questo algoritmo in outdoor, per capirne le debolezze e guidare uno sviluppo futuro. Questa Tesi vuol far infatti da apri pista a nuove tesi atte a sviluppare funzionalità avanzate su Unmanned Vehicle. Inizialmente si è ipotizzato di dotarsi di un MAV su cui effettuare il lavoro. É stata quinti eseguita una ricerca di mercato per conoscere i costi di questi veicoli, le loro possibilità di utilizzo ed eventuali sviluppi futuri. Ciò che è stato subito evidente è che vi è un mercato stratificato a livelli. Al livello più basso troviamo piattaforme con prezzi variabili dai 30 ai 500 euro. Anche se chiamati "droni" spesso si tratta di semplici radiocomandi, dove solo i modelli più costosi offrono qualche funzionalità già integrata. Lasciano però difficilmente spazio alla modifica. Fa eccezione il Parrot della AR.Drone, che dispone di un pratico SDK che permette di interagire con il veicolo tramite WIFI ed ha a disposizione una telecamera HD 720p con obbiettivo grandangolare capace di acquisire video a 30 FPS codificati in H264, una telecamera VGA da 60 FPS puntata verso il suolo, giroscopio, accelerometro, magnetometro, barometro e sensori ad ultrasuoni, processore ARM 32bit ad 1 GHz, 1GB di RAM e sistema operativo Linux 2.6.32. Oltre questa fascia vi è poi una fascia più professionale, dove

vi sono i primi dispositivi pensati per usi più avanzati come riprese amatoriali. Questi dispositivi possono facilmente superare il migliaio di euro ma offrono una gamma di funzionalità completa che li rende pertanto dei veri e propri UAV. Ci sono molti produttori in questo fascia di mercato, i quali usano piattaforme sia aperte che non, a volte fra i prodotti a listino della stessa azienda. É bene porre attenzione quindi oltre alle caratteristiche del mezzo anche al Flight Controller installato nel caso si voglia poter accedere anche al suo codice. Il più delle volte questo non è però necessario grazie alle interfacce verso l'esterno già implementate o ai software forniti in bundle che fungono da middleware fra il proprio codice e la piattaforma. Inoltre vi è sempre la possibilità di installare in modo artigianale embedded PC a bordo per arricchire la gamma di funzionalità archiviabili. Infine vi sono i prodotti professionali più avanzati che partono da qualche migliaio di euro fino ad arrivare a decine se non anche centinaia di migliaia di euro. Si tratta di dispositivi estremamente avanzati, dotati di telai in fibra di carbonio e CFRP, che gli conferiscono grande resistenza e basso peso. Con eliche perfettamente calibrate per ridurre le vibrazioni ed aumentare l'autonomia. Questi infatti sono capaci di restare in volo per tempi prolungati, a volte fino a mezz'ora nelle giuste condizioni. Anche i Flight Controller sono molto più avanzati, dando a questi mezzi la capacità di rimane stabili anche in condizioni di forte vento, e di eseguire manovre avanzate con semplici comandi. Vengono usati solitamente per riprese aree professionali o altri scopi avanzati. I multirotori per ragioni tecniche tendono a non essere molto grandi, questi mezzi infatti peccano di efficienza se paragonati agli elicotteri. Difficilmente si va oltre soluzioni a dodici eliche (anche se recentemente è stato presentato e-Volo VC200, un veicolo per il trasporto umano realizzato tramite l'utilizzo di 18 eliche [60]), perché la semplicità meccanica a questo punto non ripaga la perdita di efficienza. Anche l'autonomia è per ora fortemente limitata dalla capacità delle batterie. L'utilizzo di più motori è un problema per questi veicoli, perché richiedono molta energia per essere azionati. Una piattaforma che poteva rispondere ai nostri requisiti è realizzata dalla tedesca AscTec, ma il costo superava facilmente i 7000 euro. Inoltre non avendo l'esperienza per pilotare uno di questi mezzi, né tanto meno le conoscenze tecniche adeguate

per intervenire in caso di guasti, abbiamo optato per richiedere la collaborazione della vicina facoltà di Ingegneria Aerospaziale di Forlì, che vanta uno staff preparato ed esperto oltre che un vasto campionario di mezzi. La facoltà di Ingegneria Aerospaziale di Forlì ha messo a nostra disposizione un UAV molto avanzato, una piattaforma DJI S800 Evo. Su questa piattaforma è stata aggiunta una struttura artigianale atta a sostenere un nuovo hardware, aggiunto per rispondere ai requisiti minimi necessari per effettuare le prime sperimentazioni. Da queste è stato possibile trarre considerazioni sulle difficoltà tecniche affrontabili e sulle soluzioni percorribili per potenziare questo primo lavoro. Questa può quindi essere considerata anche come una base di partenza per chiunque voglia realizzarsi un UAV in casa partendo da una comune piattaforma multirotore (o perché no, anche altre piattaforme) disponibile in commercio o autonomamente realizzata. Su questa piattaforma abbiamo riadattato molto dell'hardware a nostra disposizione, fra cui l'embedded PC, la telecamera e l'antenna WiFi per la trasmissione dei dati. L'algoritmo scelto come detto è il PTAM. Questo perché il codice distribuito in openSource rappresenta un buon punto di inizio per i lavori futuri. Ci ha permesso infatti di tralasciare per il momento gli aspetti più puramente implementativi e di concentrarci sulla modellazione del problema.

6.1 Piattaforma UAV

La piattaforma UAV utilizzata, come anticipato è un DJI S800 EVO, realizzato dalla DJI Innovations. E' un esacoptero, cioè un multicoptero con sei eliche, a coppia di due controrotanti. Queste sono disposte in configurazione esagonale. La DJI Innovations è un azienda del settore UAS (unmanned aerial systems) con base a Shenzen, Cina, che produce piattaforme commerciali sia per amatori che per esperti, come specificamente evidenziato dal loro portale web. Il DJI S800 EVO è in particolare una piattaforma per esperti, essa infatti è costituita da un frame molto semplice, leggero e resistente realizzato in CFRP (*Carbon Fiber Renforced Plastic*). Le eliche dei motori sono ripiegabili, e sono della misura 15 x 5.2 pollici. Esse sono nell'ancora più pregiata fibra di carbonio così da essere molto leggere, pesano solo 13g l'una,

e resistenti. Inoltre queste eliche sono bilanciate dallo stesso produttore per ridurre al massimo le vibrazioni ed incrementare l'efficienza. I sei motori sono dei *brushless outrunner* con statore di dimensioni 41 x 14 mm, un KV di 400rpm/V. Hanno una potenza di 500W e pesano ognuno 158g.

Figure 6.1: Il DJI S800 Evo è uno dei più avanzati prodotti presenti nel listino professionisti della DJI con un costo del solo frame, privo quindi di Flight Controllers o gimball per la camera, che si avvicina ai 3000 euro.

6.1.1 Dati inerziali

I dati inerziali utilizzati per il paragone con quelli ottenuti dall'algoritmo di autolocalizzazione provengono, non dal controller di volo che pilota il mezzo, ma da una seconda piattaforma realizzata dalla facoltà di Ingegneria Aerospaziale di Forlì su base Arduino Due. Arduino Due è una piattaforma di prototipizzazione rapida dotata di processore ARM a 32bit con 84MHz di frequenza di clock, 96KB di SRAM e 512 KB di Flash per il codice. E' dotata di 54 pin digitali di I/O, di cui 12 capaci di rilasciare anche un segnale PWM in uscita, 12 pin analogici di ingresso e due di uscita, dotati di DAC. Su questa piattaforma sono stati montati un sensore inerziale a dieci gradi di libertà, accelerometro, giroscopio, magnetometro, barometro ed una IMU con gli stessi gradi di libertà. É connessa poi ad una unità GPS esterna il cui ricevitore è montato su una torre rialzata, per prevenire disturbi elettromagnetici da parte della scheda o dell'embedded PC e per massimizzare la ricezione del segnale satellitare.

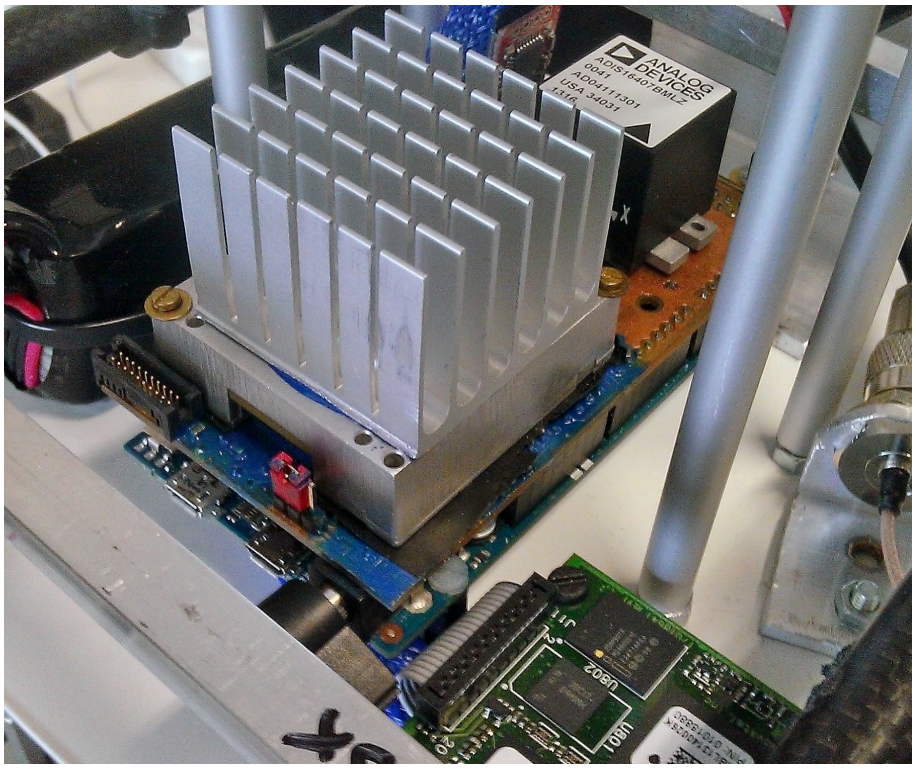


Figure 6.2: La piattaforma sviluppata su Arduino dalla facoltà di Ingegneria di Forlì completa e montata sul frame. In primo piano si può notare il grande dissipatore che mantiene stabile la temperatura della MEMS IMU (Inertial Measurement Units). Dietro di essa si nota invece un MEMS Inertial Sensors a 10DOF (Degrees of Freedom), 3 per l'accelerometro, 3 per il giroscopio, 3 per il magnetometro ed uno per il barometro.

Figure 6.3: Nell'immagini soprastanti è presente il modulo che si occupa del collegamento della gestione del sistema GPS (Global Positioning System) e la sua antenna.

6.1.2 Embedded PC

Come embedded PC è stata usata una soluzione già disponibile che è stata poi modificata per lo scopo. Si tratta del Optris PI Lightweight, una soluzione pensata per effettuare riprese ad infrarossi su UAV al fine di misurare le temperature a distanza. Questa soluzione è composta da un embedded PC e una telecamera termica. Il PC è una macchina dotata di processore Intel Atom Z530 ad 1.6 GHz. 512 MB di RAM DD2 a 533MHz e 2 GB di memoria SSD. Per collegarsi a questa unità vi sono 2 porte USB, una porta Mini-USB ed una connesione Ethernet. Sopra questo PC gira una copia autorizzata di Microsoft Windows XP Professional, leggermente modificato per ridurre l'impatto prestazionale. Le poche modifiche effettuate a questa unità riguardano il comparto software. Tutta la suite di programmi preinstallati per la gestione della telecamera termica sono stati disattivati, mentre sono stati installati quei software necessari all'utilizzo delle periferiche connesse. Quindi i driver di Arduino Due, i driver per il dongle Wifi USB della NetGear e il software Matrix Vision mvIMPACT per la gestione della telecamera. Per interagire con il PC basta connetterlo ad una rete locale, o tramite Ethernet o tramite Wifi, e connettersi al servizio di desktop remoto offerto dall'applicazione UltraVNC Server. Tramite quest'ultimo si possono anche trasferire i file comodamente con poche azioni. Si è evitato quindi di installare un IDE su questo PC e tutto il software è stato realizzato su una seconda macchina. Basta poi trasferire l'eseguibile e le DLL connesse alle librerie usate per poterlo lanciare da remoto. Ciò ha causato qualche imprevisto poiché, a volte, librerie dichiarate compatibili a Windows XP richiedevano come prerequisito la presenza di altre librerie, non installabili a causa della criticità di memoria a disposizione. Altre volte le librerie semplicemente non riuscivano a girare correttamente a causa della potenza elaborativa (questo si vedrà soprattutto nel capitolo dedicato alla telecamera, quando, per ridurre la banda necessaria, delle librerie di compressione sono state messe in campo). Erano necessari più tentativi quindi prima di raggiungere il risultato finale.

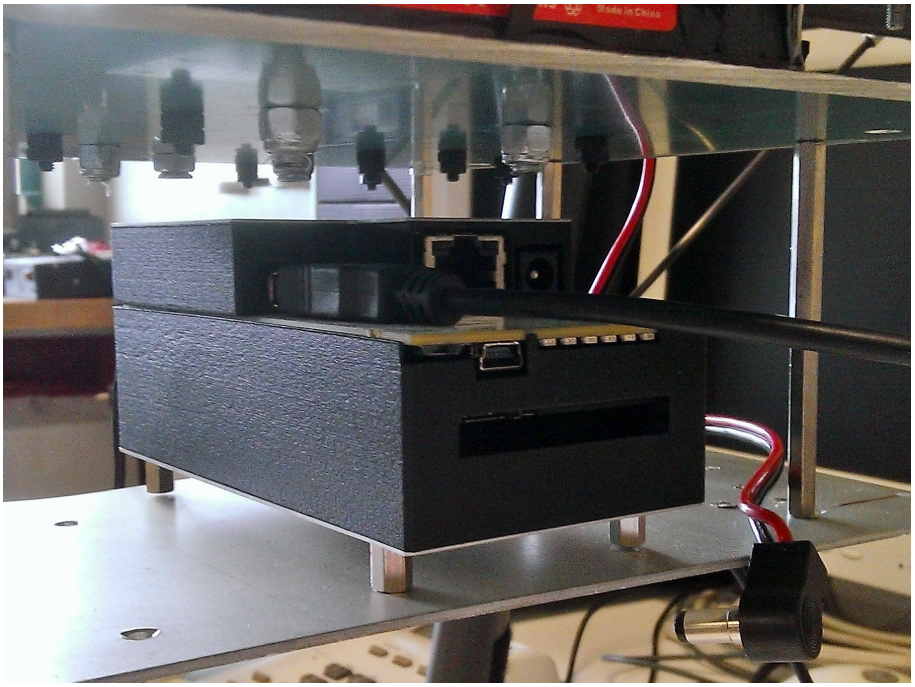


Figure 6.4: L'embedded PC della Optris installato nella sua sede

6.1.3 Trasmissione Dati

Il DJI S800 Evo monta una radiotrasmittente per comunicare con il radiocomando, ma non è adatto alla trasmissione dei dati a noi necessari. Per applicazioni FPV la facoltà di Forlì utilizza un modulo apposito per la trasmissione di video analogico. Ma anche questo non poteva essere utilizzato per diverse ragioni. Sia l'embedded PC che la telecamera non dispongono di una uscita video analogica, avrebbero richiesto quindi l'installazione di ulteriori dispositivi, non a nostra disposizione. Inoltre restava la necessità di inviare i dati dell'unità inerziale correlati ai frame immagine acquisiti dalla telecamera. La scelta è stata quella di installare una antenna WiFi sull'embedded PC tramite dongle USB. Si è utilizzata una NetGear N150 già a disposizione. Un adattatore di fascia bassa ma che supporta il protocollo 802.11 b/g/n. Sul computer remoto invece, trattandosi di un notebook e quindi già dotato di antenna WiFi integrata, non è stato installato altro hardware.

Creando connessioni ad hoc la banda risulta però limitata a soli 11MBits/sec. Non sufficienti a tenere un flusso video/dati costante. Utilizzando un software openSource chiamato VirtualRouter è stato possibile superare agevolmente questo problema. Questo infatti trasforma il proprio computer in un router,

che appare così a tutti gli altri dispositivi come un'infrastruttura. In questo modo è stato possibile avere una connessione fra l'embedded PC e il notebook di 65/72 MBits/sec. Stranamente pur provando a settare manualmente molti parametri, sia sull'hardware installato sull'embedded PC, sia sul notebook, non si è riusciti ad ottenere connessioni superiori a questa velocità anche se l'adattatore USB può raggiungere una velocità teorica di 150MBits/sec e l'antenna installata nel notebook supporta i protocolli 802.11 b/g/n/ac. In futuro, verranno effettuate ulteriori prove, anche con hardware differente per cercare di scoprire se si trattasse di un problema di natura fisica o meno. Da un punto di vista software la trasmissione è uno degli aspetti che ha richiesto il maggior lavoro. Per la natura dell'applicazione la scelta è stata di affidarsi all'uso del protocollo UDP per la trasmissione dei dati. Di fatto è una connessione punto-punto fra due dispositivi, ma che muovendosi possono essere soggetti a perdita di dati. Una trasmissione TCP pur risultando per sua natura più affidabile, rischia di divenire troppo inefficiente per via delle politiche di gestione della congestione già implementate. La perdita dati non è trascurabile, per un buon funzionamento degli algoritmi di autolocalizzazione le immagini non devono essere compromesse, ed è d'aiuto se persino il flusso di immagini non è compromesso. Purtroppo come evidenziato da test effettuati con irperf, il tasso di errore sulla trasmissione dati via WIFI risulta piuttosto alto. Un errore quasi del 7% con i due apparecchi uno a fianco all'altro e ben del 13% con gli apparecchi poco più distanti. Le prime prove hanno poi confermato questi valori, mostrando a monitor casuali ma frequenti immagini molto compromesse quando la dimensione dei pacchetti era settata al massimo dei 65KB disponibili, o la totalità delle immagini con delle leggere compromissioni quando si usavano pacchetti dati più piccoli, con dimensioni fra i 512 Byte e il KByte così come consigliato da alcuni esperti. Bisognava introdurre quindi un algoritmo di reinvio dati a fronte di perdite. Ma questa necessità richiedeva una profonda rivisitazione dell'architettura software realizzata, con il rischio di passare più tempo nell'implementazione di un dettaglio tecnico che nella risoluzione del problema. Si è dato un occhio alla disponibilità di protocolli di livello applicativo nel mondo openSource alla ricerca di un'implementazione concreta

del Reliable-UDP, un UDP arricchito cioè di quelle funzionalità di affidabilità tipiche del TCP. UDT, UDP-based Data Transfer è stata la scelta finale. Si tratta di un'implementazione openSource a cura del Dr. Yunhong Gu e del Dr. Robert L. Grossman. Questa libreria permette di avere connessioni affidabili UDP, gestite come fossero connessioni TCP, svincolando dal problema di dover gestire i vari destinatari manualmente, anche se in questa fase di sviluppo del progetto non è ancora necessario. Inoltre si può scegliere di far apparire la trasmissione come fosse uno stream oltre che come uno scambio di pacchetti, ma questa soluzione non è stata usata. Molto utile però la possibilità, realizzabile grazie ad un semplice flag, di attivare anche la consegna ordinata dei datagrammi, rimuovendo la preoccupazione di dovere gestire i pacchetti ricevuti dall'utente finale. Questa libreria è appositamente pensata per cercare di massimizzare l'utilizzo di bandwith disponibile, infatti è usata in molte applicazioni di supercomputing. Questo da una certa garanzia sulla possibilità di sfruttare a pieno il nostro collegamento WIFI. Infatti i numeri non sono a nostro favore, la telecamera genera un flusso di dati variabile fra 35 MB/sec e i 48 MB/sec se usata a pieno potenziale. Considerando l'alto tasso di errore della rete e una certa difficoltà ad usare più di 40MB/sec, questo dato rischia di diventare molto problematico. É per questo è stato deciso di mettere in campo una compressione dei dati video per ovviare al problema. Ma ne discuteremo nel dettaglio nei sotto paragrafi dedicati nel capitolo della telecamera.

Figure 6.5: Il ricevitore wifi Netgear N150 collegato ad un hub Thrust. Tramite l'utilizzo del hub è stato possibile ampliare il numero di porte USB a disposizione e al contempo posizionare esternamente all'infrastruttura metallica l'antenna WiFi per una miglior ricezione del segnale.

6.1.4 Telecamera

La telecamera utilizzata è una Matrix Vision Bluefox USB2.0 121 G/C. Si tratta di una telecamera industriale con sensore CCD a risoluzione di 1024 x 768 pixels. La telecamera offre un'interfaccia USB 2.0 e tramite i suoi driver e il suo SDK chiamato MV IMPACT è possibile comandarla e configurare tutti i suoi parametri. Inoltre dispone anche di una porta D-Sub a 9 pin attraverso la quale può essere connessa a device esterni per pilotare l'acquisizione delle immagini o per fare in modo che sia la telecamera a pilotare questi device. Proprio questa seconda opzione è risultata comoda ai fine della Tesi. Utilizzando il trigger out della telecamera connesso ad un trigger in sulla piattaforma Arduino, si potevano acquisire i dati di assetto in correlazione ai frame della telecamera. Essa infatti è la componente più lenta del sistema, con un frame rate massimo dichiarato di 39 fps ma che in condizioni reali si assesta fra gli 11 e i 18 fps a piena risoluzione. Questo ha permesso di pilotare l'IMU e il GPS con tranquillità. Tramite un supporto metallico è stata montata in posizione verticale, con l'obbiettivo puntato verso il basso. Attualmente questa scelta è preferibile per la sperimentazione sugli algoritmi di autolocalizzazione. Una volta raggiunta la piena maturità del sistema si potrà ipotizzare un montaggio su gimball per permettere una visione libera dell'ambiente sottostante il multirotore. L'acquisizione di immagini può essere effettuata dal codice grazie all'accesso al buffer immagine che risulta disponibile sempre tramite le API messe a disposizione dal SDK. Il formato immagine di destinazione è configurabile fra varie opzioni a disposizione fra cui citiamo le due da noi utilizzate, RGB888 e Mono8. RGB888 permette di avere i dati impacchettati in 24bit per pixel, 8 per ogni componente colore (rosso, verde, blu). Mono8 invece restituisce un frame immagine in scala di grigi dove la profondità di un pixel è definita da 8 bit. Tutti i formati disponibili sono però non compressi. Si hanno quindi immagini da 787 KB l'una quando si effettuano acquisizioni in scala di grigi, che divengono circa 2,4 MB se prese a colori. Un flusso di immagini di questo peso è ingestibile per la connessione WIFI installata sul nostro UAV. Infatti con un frame teoricamente variabile fra 15 e 20 frame per secondo, la banda richiesta

solo per il flusso video ammonta ad un valore variabile fra 36 e 48 MB/sec, cioé 288-384 Mbit/sec. Molto maggiore della nostra portata teorica di 72 Mbit/sec. La soluzione è stata comprimere queste immagini al fine di ridurre lo streaming. Infatti senza nessuna compressione video il flusso dati fin dalle prime fasi mostrava un forte ritardo. Rapidamente accumulava ben 30 secondi di ritardo che poi cominciavano inesorabilmente a salire mano a mano che la trasmissione continuava. Saturato il buffer lato mittente il frame rate del video crollava ad un frame ogni trenta o più secondi. Di seguito sono presentate tre possibili opzioni di compressione implementate nel codice. Questi sono stati inseriti in ordine inverso a quello presentato, questo perché nelle prime fasi implementative a causa di limiti prestazionali dell'embedded PC, in mancanza di ottimizzazioni del codice, o semplicemente per problemi di compatibilità software, è stata necessaria la ricerca di soluzioni alternative mentre una delle scelte veniva temporaneamente accantonata per trovare una soluzione.

Figure 6.6: La telecamera Matrix Vision BlueFox nella sua installazione sulla piattaforma. In futuro l'installazione di dampener ball permetterà di ridurre ulteriormente le poche vibrazioni ancora presenti. Le eliche ben calibrate di fabbrica aiutano ma le vibrazioni sono sempre presenti in questo tipo di veicoli.

Snappy

Snappy [61] è una libreria openSource per la compressione/decompressione dei dati. L'obbiettivo di questa libreria non è raggiungere alti livelli di compressione dati ma avere una buona velocità a fronte di un guadagno ragionevole. In ordine cronologico è stata una delle ultime scelte ad essere implementata, ma è stato necessario aggiungerla per le grandi problematiche che le altre opzioni presentavano. L'installazione di Snappy in Windows è particolarmente semplice, ancor di più se si utilizza Visual Studio come IDE di sviluppo. In quest'ultimo caso basta una istruzione al packet manager per ritrovarsi Snappy perfettamente integrato nel progetto. Le API per il suo utilizzo sono ridotte ai minimi termini. Le chiamate fondamentali sono solamente due:

- snappy_compress
- snappy_uncompress

L'utilizzo di questo algoritmo di compressione non causa nessun significativo rallentamento al software, ma permette di dimezzare la dimensione dei dati di una immagine in scala di grigi, mentre per le immagini a colori la riduzione varia fra da un 30 al 40%. Utilizzando questo algoritmo di compressione il ritardo della trasmissione si è ridotto a soli 11 secondi. Scegliendo di abbassare la risoluzione ad un ragionevole valore di 640x480 si sono potuti cominciare a fare i primi esperimenti di comunicazione remota per periodi di tempo medio-lunghi. Ma si sentiva la necessità di guadagnare ancora qualcosina perché in movimento il flusso video tendeva facilmente a bloccarsi, probabilmente per un innalzamento del rate di perdita dei pacchetti UDP.

Opencv - PNG

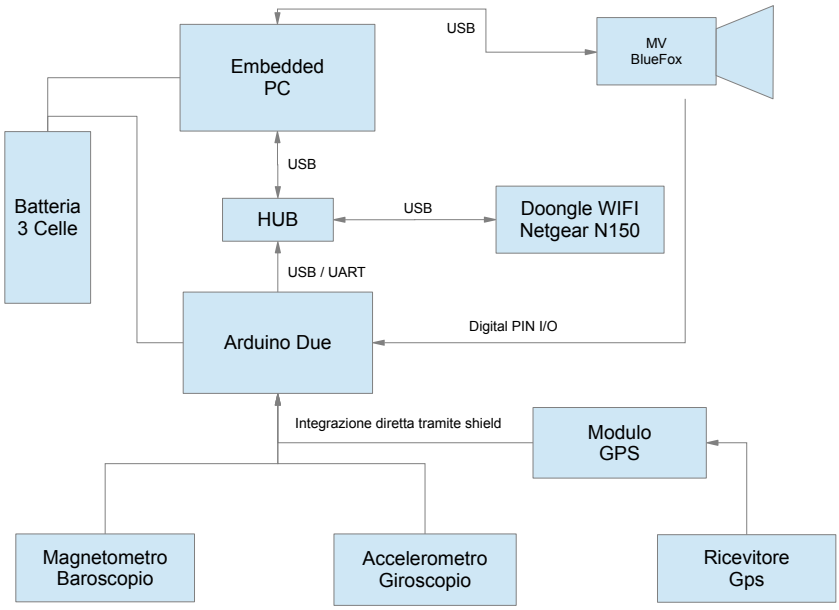
OpenCV [62] è una libreria openSource per la computer vision e il machine learning, utilizzabile grazie alla licenza BSD che vanta più di 47mila persone coinvolte nel suo sviluppo e grandi compagnie come Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota. Utilizzare OpenCV per effettuare la compressione di immagini è una delle prime soluzioni ipotizzate, complice la personale formazione precedente acquisita su questa libreria. Problemi di compatibilità delle librerie DLL hanno però rimandato l'implementazione di questa funzionalità. Infatti fino alla versione 2.4.8 di OpenCV, le DLL per Windows XP erano compromesse. Esse richiedevano una chiamata di sistema a 64 bit inserita sui sistemi operativi Windows solo a partire da Vista. Questo rendeva praticamente inutilizzabile la funzione di encoding delle immagini su Windows XP. Fortunatamente a breve vi è stato il rilascio della versione 2.4.9 che ha risolto questo problema. L'uso di un encoding di immagini ha permesso di guadagnare molto in termini di banda utilizzata. Infatti la riduzione di memoria sia in scala di grigi che a colori si assesta solitamente su un 75%. Inoltre essendo PNG un formato lossless l'immagine che ne si riottiene a destinazione è perfettamente identica all'originale. Il flusso dati con l'uso di questa compressione è diventato costante con ritardi che nei casi peggiori arrivano nell'ordine dei 2-3 secondi. Durante l'implementazione di questo sistema di compressione era stata iniziata l'implementazione anche della compressione descritta nel prossimo paragrafo, ma si è poi deciso di accantonarla a seguito degli ottimi risultati ottenuti. Con l'arrivo della stagione calda, le componenti di fascia bassa come l'antenna WIFI privi di un flusso d'aria adatto a raffreddarli hanno visto in alcune occasioni di molto ridotte le proprie prestazioni costringendo a riprendere lo sviluppo dell'ultimo algoritmo di compressione.

FFMpeg - H264

FFMpeg [63] è un framework per l'elaborazione di media video. Permette di codificarli e decodificarli in quasi tutti i codec disponibili. Il suo utilizzo non è semplice e la documentazione risulta a volte carente sull'aspetto del codice, ma fra i forum si trova una buona quantità di materiale e di esperti pronti ad aiutare. Tramite questo framework si convertono le immagini ottenute dalla camera in un flusso video codificato con il codec H264. Le opzioni di compressione sono state settate per cercare una compressione meno spinta possibile e nel tentativo di ridurre la latenza (preset: veryfast, tune: zerolatency). Inizialmente si è tentato di ottenere un flusso lossless con l'opzione qp: 0 passata fra i parametri di configurazione, ma questo porta la latenza ad aumentare molto. Si introduce infatti un ritardo superiore ai venti secondi, a causa del debole processore disponibile sulla piattaforma embedded. La scelta al momento è stata quella di lasciare questo parametro di default ed avere un "gop" nullo per avere solo i-frames, sufficienti a ricreare l'immagine, senza p-frames o b-frames. Questo perché ogni qual volta un nuovo destinatario si sottoscrive al UAV ha bisogno di un i-frame per iniziare il video e se questo non è disponibile la libreria si blocca. Attualmente l'implementazione di questa funzionalità ha incontrato difficoltà che si cercherà di risolvere in seguito. Così facendo, grazie ad un intelligente utilizzo della compressione lossy la riduzione si assesta a valori sempre superiori al 90% con qualsiasi formato immagine. Questo garantisce un video praticamente costante e quasi in real-time. L'implementazione di questo algoritmo potrà tornare sicuramente molto più comoda in caso di sostituzione della camera con modelli a risoluzione maggiore o con frame rate superiori.

6.2 Architettura Hardware

La struttura hardware del UAV può essere descritta sommariamente dalla figura seguente:



Tutti i dispositivi hardware presenti sulla struttura traggono la loro energia da una batteria a tre celle al litio. Delle prove a regime hanno mostrato che questa soluzione permette una autonomia variabile fra i 10 e 15 minuti. Non ancora tantissimi, ma si tratta di un compromesso scelto per non aumentare troppo il peso con batterie a più celle sapendo che si tratta di una configurazione di prova per effettuare le prime sperimentazioni. Inoltre l'architettura Intel Atom se pur parsimoniosa nei consumi non è capace di battere le Architetture ARM. A pieno carico l'efficienza complessiva del sistema è battuta anche dalla sua cugina più esosa, l'Intel i3. Questo quindi ci fa sperare che un cambio di hardware possa giovare al sistema già con questa tipologia di batteria, incrementando la potenza computazionale e la durata in un colpo solo. Si è comunque ipotizzata la possibilità di inserire un circuito di controllo alla batteria per poi effettuare una lettura da Arduino. Questi dati verrebbero poi trasmessi a terra all'operatore, e se il livello di carica

scendesse troppo, si potrebbe far scattare anche un allarme sonoro per avvisare il personale a terra dell'esaurimento della batteria. E' un'informazione molto importante perché batterie troppo scariche posso far nascere principi di incendio con possibili gravi risvolti per la sicurezza del mezzo e del personale operante lo stesso. La piattaforma inerziale Arduino e il ricevitore WiFi della NetGear sono connessi all'embedde PC tramite un HUB della Thrust, connesso alla porta USB della sezione posteriore. La telecamera è invece connessa direttamente alla sezione frontale sempre tramite connessione USB. Un evento curioso è stato scoprire che la piattaforma Arduino ha evidenziato comportamenti strani con l'embedded PC. Il collegamento viene riconosciuto solo dalla porta posteriore del PC e, soprattutto, se il dispositivo viene connesso a PC spento preclude un regolare avvio di quest'ultimo. La procedura corretta d'avvio consiste quindi nell'accendere l'embedded PC, attendere la sua connessione automatica alla Ground Station e solo allora connettere il dispositivo Arduino. Il bootloader del PC Optris, come impostazione di fabbrica, cerca prima un dispositivo di archiviazione USB e solo in seguito avvia la sua partizione interna, questo per semplificare il ripristino del dispositivo tramite la chiave USB inclusa nella confezione. Arduino probabilmente veniva scambiato per una pendrive interrompendo quindi il regolare avvio dal SSD interno. Anche se questa è solo una teoria, perché l'assenza di un output video ci ha impedito di verificare. Potrebbe anche non trattarsi di questo dato che era possibile illuminare tutti i led del PC stesso semplicemente connettendolo tramite USB ad un Arduino alimentato, pur scollegando fisicamente l'alimentazione del PC. Questo insieme al problema di riconoscimento delle porte restano tuttora inconvenienti sui quali continuiamo ad investigare. Il collegamento tra la telecamera BlueFox e la piattaforma Arduino, per trasportare il trigger di acquisizione, è realizzato tramite una connessione fisica fra i pin 7 e 2 della porta D-Sub della telecamera e due piedini digitali della piattaforma Arduino, programmati via software per la ricezione di un trigger. Arduino offre la possibilità di alimentare a 3,3V queste porte e di attivare tramite software una resistenza da 20KΩ sgravandoci dal compito di implementare un circuito elettrico di alimentazione. Non c'è molto da dire sulla GroundStation. Consiste in un notebook Apple MacBook Pro Retina

15́ dotato di processore Intel Core i7, quad-core 2.3GHz con Turbo Boost fino a 3.5GHz. 16GB 1600MHz DDR3L SDRAM, 512GB PCIe-based Flash Storage ed antenna WiFI 802.11 b/g/n/ac. L'autonomia media di questa macchina supera le 8 ore con il sistema operativo originale, ma scende a 4 ore in Windows 7. Permette comunque di avere un tempo di utilizzo vicino alle due ore quando usata a pieno carico e con la luminosità del monitor impostata al massimo, visibile anche in giornate di pieno sole. Si consiglia però l'utilizzo in futuro di apparecchi professionali pensati per il volo in FPV (first person view), dotati di schermo antiriflesso e maggiore autonomia delle batterie.

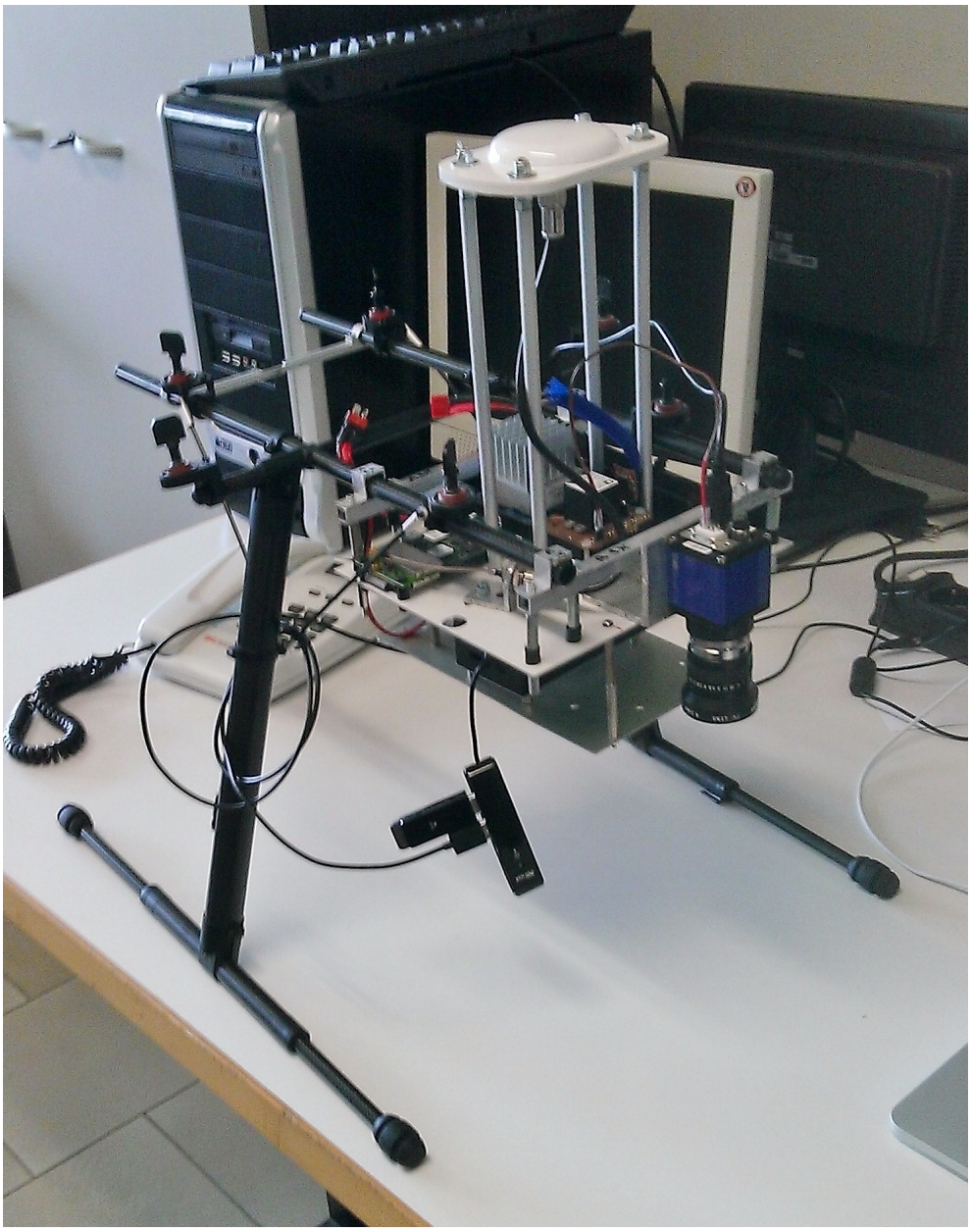
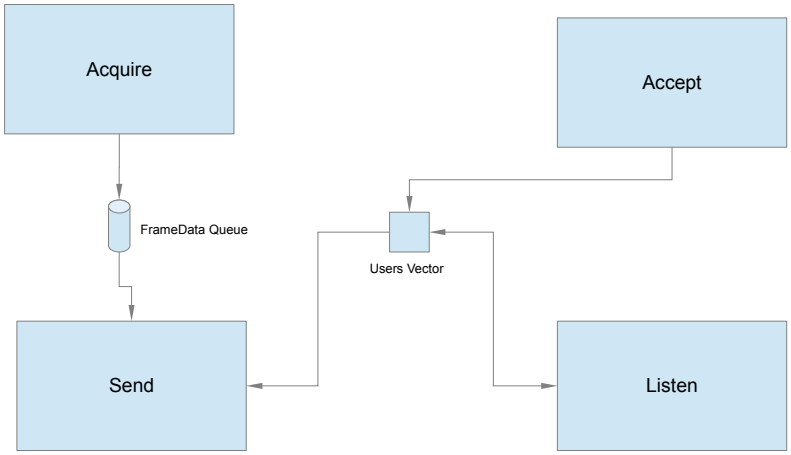


Figure 6.7: Nell'immagine si può vedere come appare la piattaforma una volta montata. Questa si connette poi al frame con i motori tramite sei semplici clip.

6.3 Architettura Software

Entrambi i software sono stati realizzati con un occhio particolare al parallelismo, implementando quindi un'architettura multi-thread. Infatti anche se l'attuale embedded PC utilizzato monta un processore single-core, un Intel Atom come descritto precedentemente, che non risulta essere un'architettura hardware specificatamente pensata per il multi-threading, questo approccio ci garantisce che qualora dovessimo sostituire l'hardware utilizzato con del nuovo più performante riusciremmo subito a sfruttarne a pieno il potenziale. La struttura core del software caricato sul MAV può essere riassunta dalla seguente figura:



Il lavoro principale è svolto da quattro thread: Acquire, Send, Accept e Listen. Il thread Acquire condivide con il thread Send una "concurrent bounded queue", una lista concorrente limitata. Inoltre Send condivide con Accept e Listen anche un "concurrent vector", un vettore concorrente.
Il thread acquire ha il compito di acquisire le informazioni, correlarle ed impacchettarle per la trasmissione. Ciclicamente interroga quindi i due dispositivi principali per raccogliere i dati da loro accumulati e poi trattarli. Il

suo comportamento a livello di pseudo codice è così descrivibile:

```
while(!conditionToStop){
   FrameData = Arduino->acquireData()
   FrameData->Image = Bluefox->acquireImage()
   putInQueu(FrameData)
}
```

L'acquisizione dei dati da parte di Arduino è un'operazione bloccante, sbloccata solo dall'acquisizione di un'immagine da parte della BlueFox. Il trigger out della MatrixVision Bluefox è un parametro pilotabile da codice, ed è stato implementato in modo tale che quando una richiesta di acquisizione è stata completata il trigger generi un segnale di discesa che fa attivare la piattaforma Arduino. Lo pseudo codice viene quindi così modificato:

```
Bluefox->requestImage()
Bluefox->signalUp()
while(!conditionToStop){
   Bluefox->waitRequest()
   Bluefox->signalDown()
   FrameData = Arduino->acquireData()
   FrameData->Image = Bluefox->acquireImage()
   putInQueu(FrameData)
   Bluefox->requestImage()
   Bluefox->signalUp()
}
```

Ora abbiamo la certezza che la piattaforma Arduino raccoglierà i dati e li renderà disponibili in un tempo prossimo, quasi istantaneo, all'acquisizione dell'immagine. Una richiesta di acquisizione potrebbe però consegnare una risposta negativa, non è infatti assicurato che ogni richiesta di acquisizione immagine si completi con successo. Per non creare del delay rispetto alla raccolta dei dati inerziali si è scelto di valutare la richiesta solo nell'istante successivo alla ricezione dei dati da Arduino. Se la richiesta immagine dovesse

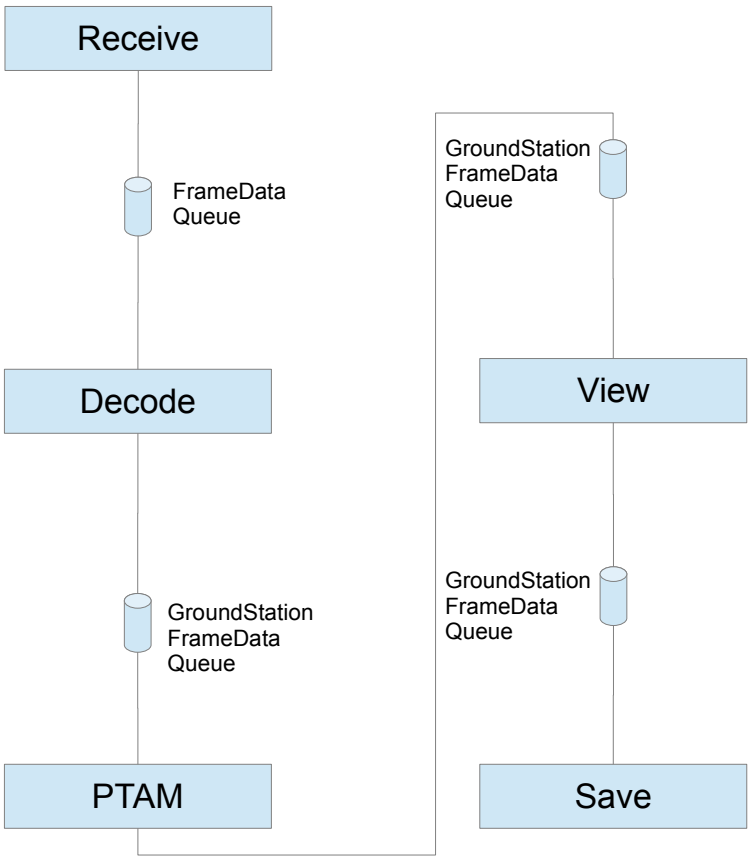
presentare un esito negativo basterebbe scartare i dati e proseguire. La frequenza della telecamera è decisamente inferiore alla frequenza massima a cui può acquisire l'IMU su Arduino, questo quindi ci da ampio margine e ci permette di poter avere qualche lettura spuria. Nella pratica però questo genere di situazione non si è verificato neanche una volta. I frame immagine consegnati dalla BlueFox sono sempre risultati corretti. La connessione USB di Arduino con il PC simula la connessione seriale UART. Il sistema operativo Windows XP mette a disposizione una libreria per effettuare trasmissioni di dati seriali. Ma a differenza della libreria introdotta da Windows 7 in avanti, risulta ancora una libreria di basso livello. Per nascondere questi dettagli tecnici all'interno del core del programma è stata preparata una classe seriale che inizializza il collegamento, per la piattaforma specifica a 115200baud, 8 bit per byte, un bit di stop e nessun bit di parità. Inoltre in questa classe sono state sviluppate comode funzionalità come la verifica della connessione, la lettura bloccante di linee di testo e, in previsione futura, la scrittura di quest'ultima. Una seconda classe a far da decoratrice a questa si occupa invece di ricevere la stringa dei dati, suddividerla e preparare un elemento di tipo FrameData che altro non è che una struttura dati da noi pensata per passare agevolmente i dati fra i vari processi, occupandosi anche di de allocarsi automaticamente quando il suo contenuto non è più utilizzato da nessuno. La stringa dati consegnata da Arduino al nostro embedded PC è strutturata come segue:

Timestamp è un parametro interno ad Arduino ed è stato utilizzato in fase di debug, rappresenta il tick del suo clock interno. Delay invece indica il tempo in microsecondi passato fra il fronte di discesa e la preparazione dei dati. Questo perché se per qualche ragione questo tempo dovesse risultare

troppo elevato (dai 100 millisecondi in su) si opterebbe per scartare i dati in quanto non ben sincronizzati con l'immagine. Roll, pitch e yaw sono gli angoli di assetto misurati dal magnetometro. Questi variano sulla base del nord magnetico, non dipendono quindi dall'inizializzazione del sistema ma dalla sua posizione nello spazio. Infine gli ultimi parametri sono informazioni ottenute dal GPS. Quality è un flag che può variare da 0 a 2 ed indica l'affidabilità dei dati visualizzati. Il parametro successivo porta il conteggio del numero di satelliti connessi. Per un volo in sicurezza questo parametro è consigliato superiore o uguale a sette. Gli ultimi tre rappresentano le ben note informazioni di latitudine, longitudine ed assetto. Questi parametri sono trasferiti come stringa di caratteri, ognuno di essi è separato dal successivo dal carattere speciale "|" e la stringa termina grazie al terminatore \n. Come detto questa stringa viene suddivisa e viene inizializzata, con i valori letti, una struttura dati a noi più congeniale. A questo punto come anticipato procediamo con il verificare se la risposta alla richiesta di cattura immagine è andata a buon fine. Se cosi dovesse essere, passiamo il puntatore al buffer immagine ad una unità di preparazione del frame immagine per spedirlo. Sulla base dei parametri d'avvio verrà selezionata una modalità di compressione fra le quattro implementate: nessuna, Snappy, PNG tramite OpenCV o H264 tramite ffMpeg. I parametri di compressione sono settati all'avvio sulla base dei parametri scelti di risoluzione e numero dei canali. Difficoltà tecniche legate ad ffMpeg impediscono purtroppo per ora di usufruire di un cambio di risoluzione dinamico così come implementato nelle prime release. Non sono mancate le difficoltà tecniche per far sì che il tutto funzionasse correttamente. OpenCV ha avuto problemi con le librerie DLL rilasciate nella versione 2.4.8, mentre ffMpeg ha sempre dimostrato una richiesta di risorse tale da non riuscire a giustificare il suo utilizzo fino alle ultime fasi quando varie ottimizzazioni hanno permesso al tutto di funzionare in armonia, se pur con il settaggio di compressione meno spinto. Una volta ottenuta l'immagine compressa viene inserita nella struttura Frame e il tutto viene inserito nella coda. La coda, concorrenziale e limitata è stata implementata anch'essa a mano. Nella libreria Windows, infatti, la coda concorrente è implementata solo da Windows 8, così come molte altre astrazioni riguardanti il

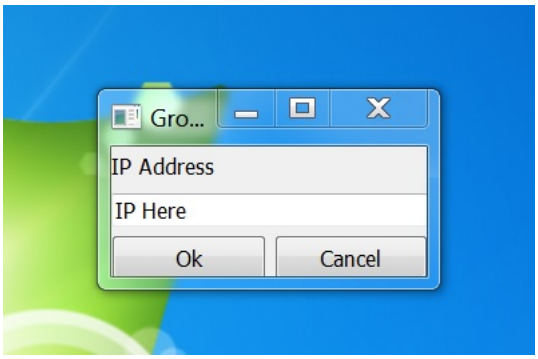
parallelismo. La limitazione al numero di elementi è arrivata in una seguente fase. Durante le prove si è verificato infatti che qualora il segnale dovesse indebolirsi troppo e la trasmissione dovesse cominciare a rallentare allora si correrebbe il rischio di improvvisi shutdown dell'applicazione. Alla fine si è scoperto che la causa era data dalla congiunzione di poca RAM e file di paging disattivato di default. Con soli 512 MB di RAM a disposizione, di cui più della metà occupata dal sistema operativo, era facile che con il crescere di immagini in coda, in attesa di essere inviate, si saturasse rapidamente la memoria RAM comportando un blocco totale. Avere questo limite sulla coda ha giovato enormemente la stabilità del sistema.
I dati inseriti in coda vengono prelevati dal thred Sender, che non deve far altro che spedirli a tutti gli utenti registrati tramite la libreria UDT. Gli utenti sono disponibili in un vettore di UDTSOCKET, che rappresentano le sessioni, in modo molto similare alle sessioni TCP. I dati di ogni singolo Frame sono inviati tramite due datagrammi. Il primo specifica al destinatario che il prossimo datagramma che sta per ricevere è di tipo dato, questo perché il sistema sull'UAV può anche inviare altre informazioni al sistema a terra, come la segnalazione che il sistema è in fase di shutdown. Il primo datagramma porta con sé anche le informazioni ottenute da Arduino, più un parametro che indica la dimensione del immagine allegata. Il secondo datagramma che viene inviato trasporta proprio i dati di questa immagine. Accept e Listen intervengono più di rado e non impattano quasi per nulla sulle prestazioni del sistema. Il primo Accept si occupa di accettare nuove connessioni da parte di nuovi utenti, creare le connessioni ed aggiornare il vettore in modo tale che anche Send e Listen siano al corrente dei nuovi utenti collegati. Queste nuove conenssioni si creano inviando un messaggio di SUBSCRIBE al MAV e attendendo poi la ricezione dei dati. Listen invece ascolta in ingresso la socket del MAV in attesa di comandi specifici. Ad esempio un utente può segnalare l'intenzione di abbandonare la sessione con un messaggio di UNSUBSCRIBE. Si può richiedere anche lo shutdown del servizio tramite messaggio STOP. Tutti i thread e le risorse sono create da un thread principale, in caso di necessità quindi per aggiungere nuove funzionalità basta semplicemente affiancare nuovi thread e collegarli alle risorse già presenti. Come detto anche

il software, che gira sulla Ground Station, è stato sviluppato a thread per sfruttare a pieno la potenza della macchina su cui girava. La struttura del software ricorda quella di una catena di montaggio come visibile dalla figura:



I vari thread si scambiano i dati ancora una volta attraverso l'uso di "concurrent bounded queue", ma stavolta potendo disporre della libreria .NET 4 la coda concorrente limitata è solo una decorazione della coda concorrente messa a disposizione dalla libreria .NET. All'avvio il sistema chiede all'utente l'indirizzo IP del dispositivo al quale intende connettersi grazie

ad un'interfaccia grafica realizzata tramite la libreria Qt5. Se l'indirizzo IP è corretto e il sistema si è connesso correttamente al veicolo allora l'intero sistema si avvia.



La successiva schermata presenta all'utente le informazioni principali per il controllo del mezzo e le immagini acquisite dalla telecamera di bordo. Nel frattempo in background i singoli thread cominciano a svolgere le loro mansioni specifiche. Receive è direttamente connesso all'ascolto dei dati in arrivo dal MAV. Questi dati vengono salvati nella stessa struttura dati usata dal software sul UAV e vengono passati al thread Decode. Questo thread ha ora il compito di trasformare questi dati e creare una nuova struttura dati più adatta a raccogliere le nuove informazioni che l'elaborazione fornirà. Inoltre facendo riferimento ad un flag presente in FrameData potrà conoscere il protocollo di compressione usato e quindi attuare una procedura inversa con il quale decodificare l'immagine da consegnare ai thread successivi. Qui ffMpeg ha a disposizione un'architettura hardware molto più potente che gli permette di ottenere decodifiche dei dati praticamente immediate. Questi dati vengono passati al thread PTAM, che esegue al suo inteno una versione modificata di PTAM, fatta a posta per poter ricevere e trattare dati forniti in ingresso da terzi, anziché cercare un dispositivo di acquisizione dal quale automaticamente cercare di ricavare i dati. Ad ogni elaborazione PTAM aggiorna le sue informazioni sulla localizzazione della camera rispetto all'origine e al piano origine da lui selezionato ed aggiorna una mappa interna dell'ambiente, arricchendola qualora possibile con nuovi punti. Le informazioni di localizzazione, il numero di frame e di keyframe e il tempo di calcolo utilizzato per

il tracking della scena vengono salvati nella struttura dati che viene poi passata al thread View. View preleva da questa struttura dati le informazioni più utili all'utente, quindi immagini e assetto, misurato dalla piattaforma inerziale di bordo, e le mostra all'utente in una semplice interfaccia grafica.

6.4 Esperimenti

In questa sezione verranno descritti gli esperimenti e i parametri valutati durante lo sviluppo di questa Tesi.

6.4.1 Parametri di valutazione

Per studiare il comportamento del PTAM tre parametri principali sono stati valutati, la precisione e l'accuratezza, le prestazioni e la robustezza del sistema. Di seguito sono descritti nel dettaglio.

Precisione e accuratezza

Uno dei parametri più importanti ma anche tra i più difficili da stimare è la precisione e l'accuratezza. La prima misura, la precisione, permette di valutare la dispersione dei dati raccolti. Questa infatti se dovesse essere troppo alta renderebbe le letture effettuate dal sistema difficilmente utilizzabili. L'accuratezza invece calcola la distanza fra le misurazioni effettuate e la verità, che nel nostro caso, in assenza di un sistema di motion tracking esterno, che avrebbe garantito un'accuratezza superiore, è data dai dati registrati dall'IMU e dal GPS, precedentemente calibrati.

Robustezza

La robustezza del sistema rappresenta la sua capacità di resistere a condizioni estreme. Questo genere di condizioni può essere rappresentato da cambi di luce, offuscamenti temporanei dell'ottica o forti accelerazioni. A mettere in crisi questo tipo di algoritmi possono essere anche situazioni in cui l'ambiente si presenta troppo omogeneo o non sufficientemente statico.

Prestazioni

Il frame rate della telecamera non era tanto elevato da dover mettere in crisi l'algoritmo, ciò nonostante abbiamo salvato parametri come il tempo impiegato per effettuare il tracking fondamentale per capire quanto margine

abbiamo a disposizione se volessimo utilizzare telecamere più veloci. Inoltre il numero di punti generati sulla mappa e il numero di keyframe salvati sono altri due parametri prestazionali importanti. A differenza dell'implementazione per ROS questa implementazione non è limitata e dovrebbe essere capace di acquisire nel tempo un numero realmente elevato di questi due fattori. Aiutando quindi la robustezza del sistema a ritrovarsi in ambienti già visitati.

6.4.2 Descrizione Esperimenti

Le prove erano state strutturate e programmate come descritto di seguito.

- Prova di laboratorio per verificare la robustezza del sistema: queste prove servono a verificare che il sistema parta e funzioni correttamente prima di procedere in fasi più avanzate del lavoro.
- Prove all'aperto in percorsi aperto: in queste prove possiamo verificare la capacità del sistema di riconoscere la corretta traiettoria dai dati acquisiti dall'ottica, e di misurare il drift che si accumula.
- Prove all'aperto di percorsi chiusi: per verificare la capacità del sistema di riconoscere i loop e riaggregare i dati.

Queste erano le prove previste, purtroppo durante le prove di laboratorio si è evidenziata una debolezza del sistema. Il lavoro si è quindi concentrato sul cercare la risoluzione di questi problemi laddove possibile o tanto meno scoprirne le cause per conoscere le aree specifiche su cui concentrare i prossimi sviluppi.

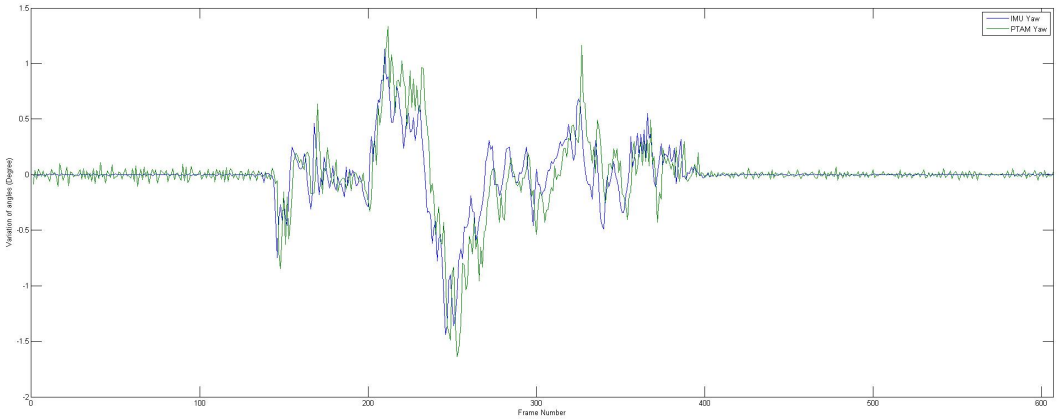
6.4.3 Analisi Risultati

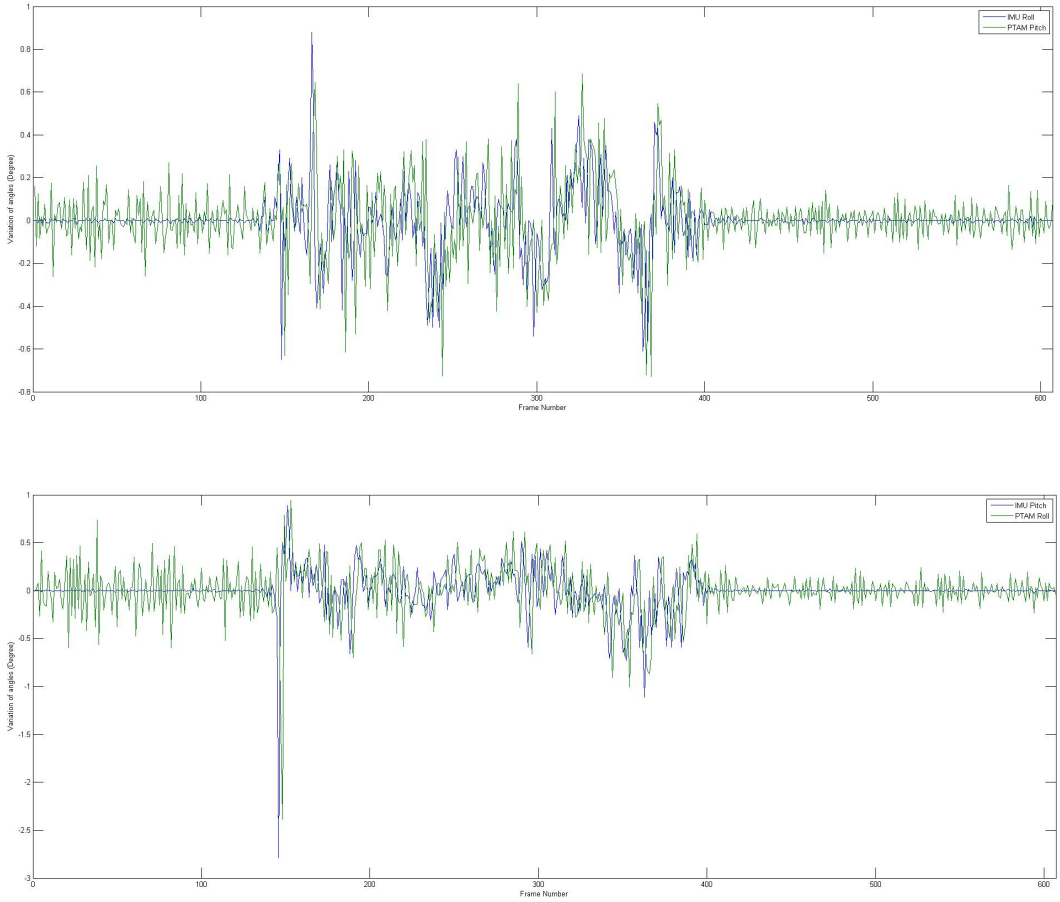
Prima di esporre i risultati è bene precisare che le diverse analisi effettuate hanno confermato la fonte dei problemi riscontrati. L'utilizzo di Windows come sistema operativo ha impattato sulle prestazioni di calcolo. Molte delle librerie utilizzate dal PTAM dichiarano già nelle specifiche bug e cali di prestazione sulle piattaforme Windows noti ma ancora non risolti. La nostra è stata una scelta obbligata dalla piattaforma di calcolo embedded a nostra disposizione, che facendo parte di un prodotto commerciale non potevamo modificare oltre certi limiti. In futuro, se si dovesse decidere di acquistare una piattaforma di calcolo embedded scorrelata da altri prodotti commerciali si potrebbe agevolmente portare l'intero codice di sistema sotto una distribuzione Linux per poter sfruttare a pieno le librerie di calcolo. Un altro punto critico è stato invece rappresentato dalla telecamera a nostra disposizione. Pur trattandosi di un prodotto industriale è pensato per impieghi molto specifici. Un frame rate di almeno 30fps è uno dei requisiti minimi per uno stabile funzionamento del PTAM. Grazie alla riduzione della risoluzione fino a 640x480, nelle giornate soleggiate potevamo, insieme alla riduzione dei tempi di esposizione ottenere valori molto vicini a questo requisito. Abbiamo però compreso che il tempo di esposizione fisso, regolabile manualmente da codice, non ci ha aiutato nell'acquisire immagini con una qualità adeguata. Infatti, pur ponendo molta attenzione alla scelta dei tempi di esposizione adeguati, ci si ritrovava all'esterno con condizioni di meteo che seppur poco variabili portavano all'acquisizione di immagini a volte troppo scure e a volte troppo sovraesposte. Ciò nonostante si sono ottenuti risultati incoraggianti che fanno ben sperare per i prossimi sviluppi. Soprattutto dagli esperimenti ottenuti con le prime modifiche al sistema. Prima di procedere ai risultati bisogna anche descrivere la correlazione fra i dati ottenuti dai due sistemi di acquisizione, l'IMU e la telecamera. La piattaforma inerziale è posizionata in modo tale che gli assi x e y formino un piano parallelo al suolo, con l'asse x che punta in avanti, l'asse y verso sinistra e l'asse z direzionato verso l'alto. La scelta deriva dal fatto che la telecamera è posta frontalmente in posizione verticale con la lente puntata verso il suolo e la parte alta della stessa è in

direzione concorde all'asse x dell'IMU. PTAM genera un sistema di tre assi ortogonali con l'asse z solidale all'asse della telecamera e direzionato verso la vista. L'asse x punta verso la destra della telecamera e l'asse y verso il basso. Utilizzando i simboli x, y e z per indicare gli assi dell'IMU e i simboli x', y', z' per indicare gli assi del PTAM abbiamo che x = -y', y = -x' e z = -z'. Sapendo che per roll si intendono le rotazioni sull'asse x, pitch quelle sull'asse y e yaw quelle sull'asse z abbiamo che il roll dell'IMU è correlato a pitch del PTAM, il pitch è correlato a roll e lo yaw è correlato allo yaw. Abbiamo poi trattato i dati per far coincidere lo stato iniziale dell'assetto a (Roll: 0°, Pitch: 0°, Yaw: 0°). In questo modo abbiamo messo in diretta correlazione le rotazioni effettivamente misurate nelle fasi in cui l'algoritmo PTAM è in esecuzione.

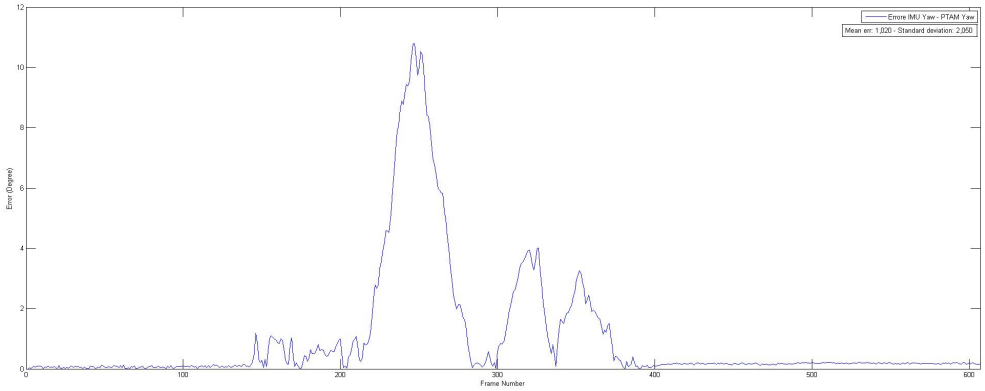
Precisione e accuratezza

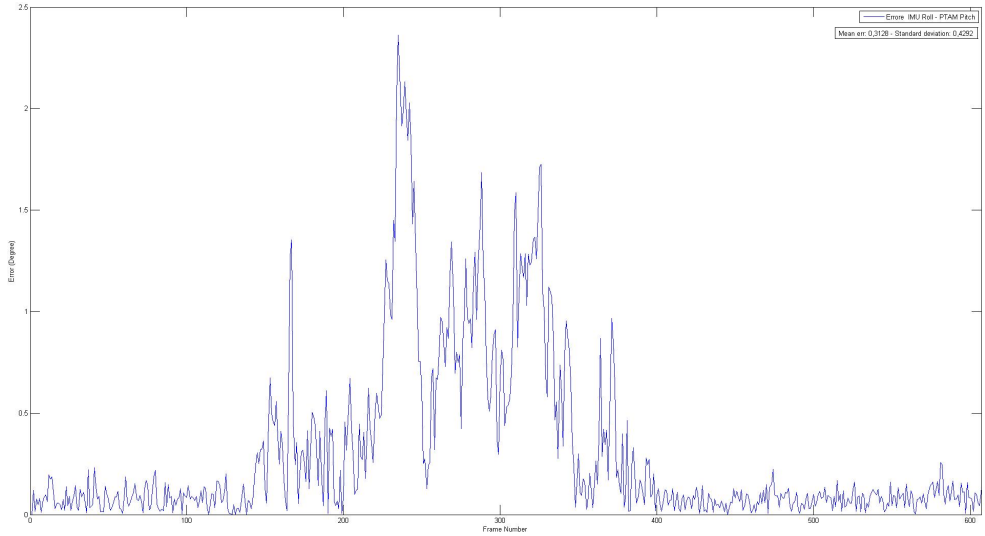
La poca robustezza alle scene outdoor ha impedito di effettuare test approfonditi sull'accuratezza dell'assetto in outdoor. In indoor siamo riusciti, se pur a fatica, le condizioni di scarsa illuminazione portavano un calo del frame rate della BlueFox fin sotto i 15 frame per secondo, ad ottenere dei dati di assetto da confrontare con le informazioni forniteci dalla piattaforma inerziale. Di seguito riportiamo a titolo d'esempio la rappresentazione grafica dei dati ottenuti da una delle sessioni di laboratorio.

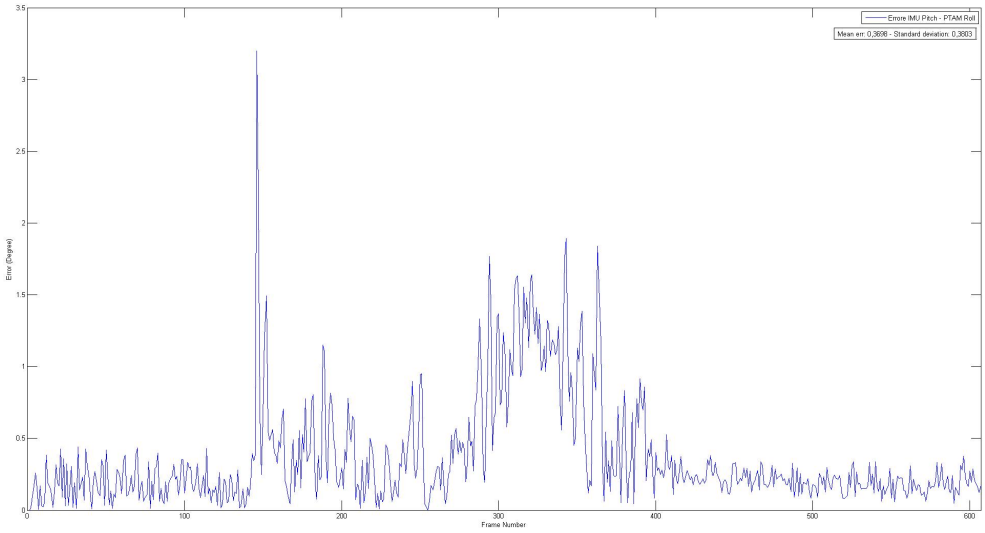




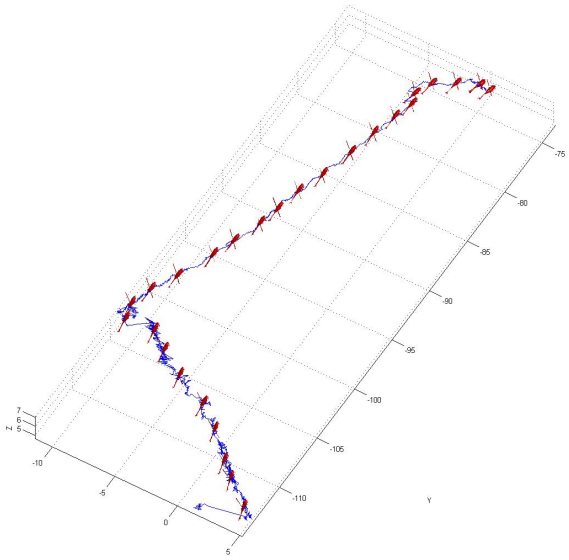
Com'è possibile vedere l'algoritmo, nonostante le difficoltà di natura tecnica, descritte nel dettaglio nella prossima sezione, riesce con un buon grado di accuratezza e precisione a seguire i cambi di assetto della telecamera riproducendo abbastanza fedelmente i dati letti dall'unità inerziale. Effettuando una misurazione dell'errore, calcolato come differenza fra l'assetto misurato dal PTAM e l'assetto misurato dall'unità inerziale, si ottiene che in tutte e tre le prove l'errore medio non è molto elevato e che in particolare la deviazione standard rimane bassa. E' curioso inoltre notare l'andamento costruttivo della mappa, è infatti quasi possibile evidenziare gli istanti in cui operazioni di local o global bundle adjustment vengono effettuate, migliorando di conseguenza l'informazione sulla localizzazione della camera.


Errore IMU Yaw - PTAM Yaw
Mean err: 1,020 - Standard deviation: 2,050
Error (Degree)
Frame Number


Errore IMU Roll - PTAM Pitch
Mean err: 0,3128 - Standard deviation: 0,4292
Error (Degree)
Frame Number


Errore IMU Pitch - PTAM Roll
Mean err: 0,3696 - Standard deviation: 0,3893
Error (Degree)
Frame Number

L'errore medio è inferiore ad un grado per tutti e tre gli assi, con una deviazione standard che rimane molto contenuta. Una sessione all'aperto, a seguito di alcuni miglioramenti apportati che desciveremo nella sezione dedicata alla robustezza, ci ha comunque permesso di provare la capacità dell'algoritmo di riconoscere il pattern del percorso. Come si può vedere dal grafico sottostante la passeggiata nel piazzale frontale all'edificio dell'inquadratura è stata correttamente riconosciuta e riportata. Purtroppo a terra non è stato possibile utilizzare il GPS, quindi non è stato possibile misurare con esattezza l'errore.



Prestazioni

Sul sistema operativo Windows sono da segnalare inspiegabili frame rate occasionali, come segnalato anche dagli autori originali, che possono minare leggermente l'esperienza utente, senza compromettere in modo grave la computazione. A parte questa nota però non vi sono particolari importanti da riscontrare. L'applicazione ha un tempo di tracking medio che si assesta fra i 4 e 5 millisecondi. Nelle sperimentazioni più proficue abbiamo gestito nuvole con più di 4000 punti e quasi 200 keyframe in memoria senza soffrire di particolari problematiche legate all'hardware. Paradossalmente in outdoor abbiamo raccolto meno punti rispetto alle sperimentazioni originali di [53] in

indoor, questo grazie alle modifiche apportate descritte nella sezione dedicata alla robustezza.

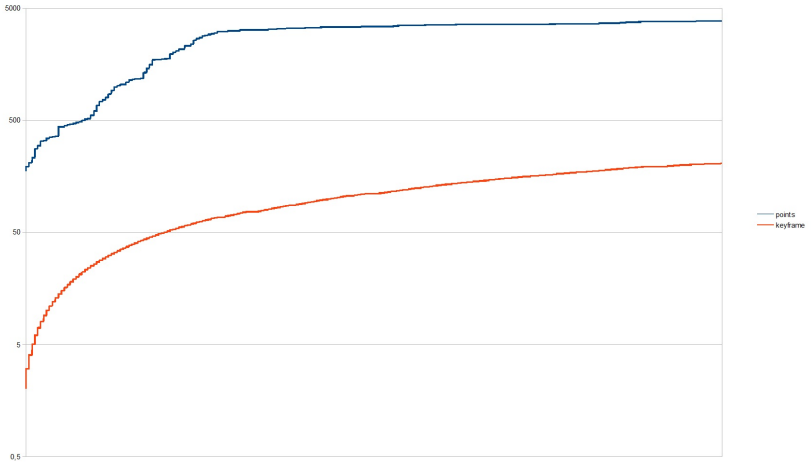


Figure 6.8: Rappresentazione grafica che mostra tramite colonne il numero di punti acquisiti e tramite linea il numero di keyframe durante una delle sessioni più lunghe all'aperto.

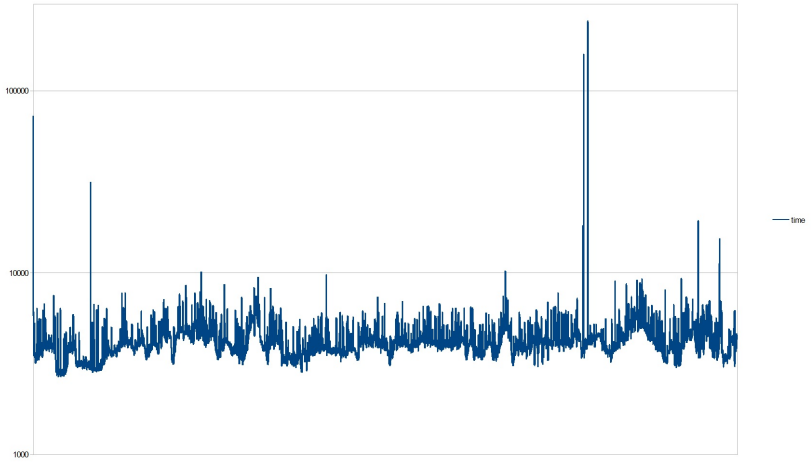


Figure 6.9: Da questo grafico se ne evince come, a meno di picchi sporadici, il tempo di elaborazione del tracking si mantenga pressapoco costante intorno ai 5000 microsecondo, cioè soli 5 millisecondi. Molto inferiori agli almeno 30 millisecondi che un frame rate a 30 frame per secondo metterebbe a disposizione.

Questo ha avuto un duplice vantaggio, perché permetterà al sistema di girare su macchine meno potenti, senza però dover attuare politiche forti di limitazione come svolto nell'implementazione ROS (dove vi è un limite al numero di keyframe salvato in memoria). Per ora abbiamo lasciato l'algoritmo PTAM libero di accumulare le informazioni di cui necessitava senza limiti per poi valutare le risorse necessarie una volta ottenuti i risultati di affidabilità desiderati.

Robustezza

Questa è stata una delle caratteristiche più critiche della sperimentazione. Dalle sperimentazioni sono emersi diversi punti critici che richiederanno ulteriori approfonditi sviluppi. Se il sistema si mostra resistente ai disturbi temporanei, si è dimostrato meno robusto a cambi di luminosità ambientale più lenti. Questo non è un problema dell'algoritmo, ma della configurazione del nostro sistema. La Bluefox permette di settare il parametro del tempo di esposizione in modo completamente manuale attraverso il codice. Se per certe applicazioni questo è l'ideale, nel nostro caso ha rappresentato un ostacolo. Semplici variazioni di meteo infatti cambiavano di molto la qualità delle immagini ottenute dalla telecamera. L'aumento dell'intensità luminosa ci portava ad acquisire immagini sovraesposte, al punto di saturare ampie porzioni dell'ambiente rappresentate completamente in bianco. L'ombra portata dalle nuvole di passaggio generava invece una situazione completamente opposta e altrettanto negativa. Le zone d'ombra ora apparivano molto scure, con tonalità degli ambienti non abbastanza contrastate. Per quanto riguarda l'algoritmo si sono notate invece criticità sugli ambienti omogenei. Ad esempio prati, pavimentazioni, asfalti. Questi ambienti portano con se un elevatissimo numero di dettagli ma con poco contenuto informativo. Spesso infatti porzioni di scena nuove venivano riconosciute dal PTAM come aree già esplorate, spingendo quindi per un riposizionamento all'interno della mappa virtuale interna completamente errato. Per cercare di limitare questo problema una modifica del sistema di mapping è stata implementata. Si è scelto di escludere dal mapping il livello più dettagliato della

piramide di immagini del keyframe elaborato dal mapper. Questo livello, il livello zero, contiene l'immagine alla sua risoluzione originale. Su questa immagine il mapper può cercare molti punti significativi da etichettare e salvare in memoria. Gran parte di questi punti significativi riconosciuti sono però dettagli molto fini come fili d'erba, trame specifiche delle mattonelle e via via dicendo. Questi dettagli però tendono a ripetersi in più aree in modo quasi identico. Escludendo dal mapping il livello zero della piramide si ottiene quindi l'esclusione dalla mappa di tutti questi punti, con un duplice effetto. Da una parte rafforziamo il tracking, il tracker quando proietterà la mappa non dovrà cercare punti troppo dettagliati nell'ambiente, che rischiano di confonderlo. Al tempo stesso si risparmia tempo di calcolo e memoria. Soprattutto quest'ultima viene ridotta di molto, poiché è proprio il livello zero quello che genera la maggior mole di informazioni. Vi è poi un altro trick che può essere utilizzato per ottenere risultati leggermente migliori, soprattutto in quelle ambientazioni che presentano grandi dettagli che tendono a ripetersi, come le facciate di un edificio. Il tracking come descritto nel capitolo cinque è diviso in due fasi. Una di aggiornamento della posa più grossolana ed una fine. E proprio la prima può compromettere leggermente il lavoro del tracker. Ampie scene simili che si ripetono possono essere confuse quando si fa una ricerca delle patch al livello più alto della piramide, il quarto livello, dove l'immagine ha una risoluzione otto volte inferiore. Nel nostro caso, poiché avevamo settato la camera a 640x480 si tratta quindi di immagini a 80x60. Se si è in questo genere di ambienti e si nota una certa confusione da parte del tracker, è bene allora disattivare il tracking grossolano ("coarse tracking"). Si noterà un notevole miglioramento. Infine, ma non per questo meno importanti, ci sono le modifiche apportate al sistema di aquisizione di nuovi keyframe da parte del mapper. Gli attuali parametri prevedono che un nuovo keyframe sia passato al mapper se sono passati almeno venti frame dall'ultimo keyframe, se la qualità del tracking è buona, se il mapper non è già sovraccarico di keyframe e soprattutto se il mapper ha bisogno di un nuovo keyframe. Ma quando il mapper ha bisogno di un nuovo keyframe? Stando all'attuale implementazione quando la distanza è superiore ad una soglia impostata fra i parametri di configurazione del sistema. Il parametro

sulla distanza e il parametro sulla qualità del tracking si sono rivelati due aspetti delicati in condizioni outdoor. Due nuovi parametri sono stati aggiunti per sensibilizzare il mapper al contesto, intervenendo in nuove situazioni di criticità per cercare di prevenire la perdita del tracking. Il parametro associato alla qualità del tracking suddiviso nelle categorie buono, scadente, male è stato ritenuto troppo netto. Esso infatti non aiuta a comprendere come la scena si sta evolvendo. E' stata introdotta una fascia di transizione fra il livello buono, dal quale si effettua ancora il mapping, e il livello scadente, dove il mapping viene disattivato. Quando ci si trova in questa fascia, definibile con un parametro di tolleranza, da noi impostato al 1%, si forza il tracker a generare un keyframe e consegnarlo al mapper, nel tentativo di rafforzare una situazione che rischia di divenire scadente, e che quindi potrebbe portarci in seguito a perdere le nostre tracce. Questo ha aiutato particolarmente, infatti dopo la rimozione del livello più dettagliato dal mapping le mappe generate contengono molti meno punti. Processare keyframe in più per non trasformare il vantaggio dell'eliminazione dei dettagli in uno svantaggio è stata una azione fondamentale per irrobustire l'algoritmo. La seconda modifica invece cerca di effettuare una valutazione sull'aspetto che la mappa presenta all'utente. La vista della camera viene suddivisa in quattro quadranti. A questo punto se uno dei quadranti dovesse risultare completamente privo di punti si forza il mapper ad elaborare un nuovo keyframe. L'idea dietro questa soluzione è che se la telecamera comincia a muoversi, uno o più dei quattro quadranti in breve tempo si svuoterà. Riconosciuta questo stato, si può intervenire per mitigare le possibili problematiche del movimento della telecamera. Una modifica non effettuata potrebbe addirittura prevedere dei livelli di tolleranza locali come quelli applicati a livello globale alla mappa, per intervenire ancor prima che un quadrante sia completamente sgombro e quindi in una situazione più critica. Queste due modifiche hanno irrobustito molto il comportamento in outdoor del PTAM, senza però renderlo comunque perfetto, a causa delle problematiche hardware di cui già ampiamente discusso. Si ritiene comunque ipotizzabile che la sostituzione del features detector possa portare dei vantaggi anche in presenza di immagini di qualità non per forza elevate.

Chapter 7

Conclusioni

In questa Tesi è stata realizzata una piattaforma hardware concepita appositamente per essere impiegata su un *Micro Aerial Vehicle* (MAV), un esacoptero della DJI. Diverse componenti hardware a nostra disposizione sono stati fisicamente interconnesse e successivamente fissate su una infrastruttura metallica. Per coordinare e gestire tutte le periferiche sono stati implementati in C++ diversi moduli software, utilizzati per realizzare un'infrastruttura capace di acquisire i dati, inviarli dalla piattaforma connessa al MAV ad un computer remoto il quale, dopo averli elaborati, visualizza i più significativi all'utente finale. Durante questo lavoro si sono incontrate, ed affrontate, numero difficoltà tecniche e le soluzioni sviluppate hanno permesso di ottenere una trasmissione di video e informazioni d'assetto tramite trasmissione Wifi e con un ridotto lag. Utilizzando questa piattaforma abbiamo svolto esperimenti di laboratorio sull'algoritmo di autolocalizzazione PTAM, opportunamente modificato per aumentarne la robustezza nell'utilizzo in *outdoor*. I test eseguiti hanno fornito risultati incoraggianti sulla capacità di autolocalizzazione spaziale evidenziando, tuttavia, i limiti dell'hardware a nostra disposizione. D'altra parte, ciò ha consentito di comprendere le caratteristiche fondamentali per la futura selezione di nuove periferiche da acquisire per potenziare la piattaforma. Questo è il primo lavoro che si prefigge l'obbiettivo di sviluppare le funzionalità necessarie per garantire ad un veicolo autonomo la possibilità di completare compiti, con un livello di complessità superiore a

quanto effettuato fino ad oggi, con il minimo intervento umano. La capacità di conoscere la propria posizione nello spazio consente di acquisire informazioni nuove, progressivamente durante l'esplorazione, e ciò rappresenta uno dei requisiti fondamentali che questo genere di veicoli deve possedere. L'attuale implementazione con le modifiche effettuate ha consentito in parte di superare le limitazioni dell'hardware e ci ha permesso di ottenere buoni risultati. L'approccio che prevede le due fasi iterative di localizzazione della posizione ed il mapping dell'ambiente (noto come SLAM) è stato ampiamente definito nella sua formulazione teorica. Le difficoltà, come spesso avviene, sorgono durante l'applicazione reale e sono legati all'hardware utilizzato e alla divergenza dall'idealità dei dati acquisiti, che richiede elevata potenza di calcolo. Se fino ad ora si sono usate principalmente stazioni di terra per l'elaborazione del flusso di immagini, lo sviluppo di nuove architetture embedded ha finalmente fornito strumenti più potenti ed a basso consumo energetico in grado di rappresentare un'alternativa verso la completa indipendenza e autonomia. Ora è nostro compito saperle sfruttare.

7.1 Considerazioni e proposte sulle piattaforme di calcolo

L'esperienza di questa Tesi ha permesso di maturare molta conoscenza sulle necessità hardware e software utili allo sviluppo di questo tipo di applicazioni. Diversi sviluppi possono essere proposti che partendo da quanto ottenuto in questa Tesi possano mirare a migliorare l'esperienza finale. Nuovi lavori possono provare a sostituire il PC embedded utilizzato con uno completamente nuovo. In questi anni, sull'onda del Raspberry PI, molte nuove piattaforme di embedded PC si sono sviluppate. Il termine corretto con cui chiamarli è single-board computers ed è un mercato in forte espansione, come si può vedere dalla lista aggiornata presente su Wikipedia [64]. Recentemente, molte di queste hanno cominciato ad evolversi per presentare hardware adatto all'impiego sugli UAV, in particolare sui MAV. E' il caso ad esempio di Navio [65], uno shield per Raspberry che permette di utilizzarlo

facilmente come autopilot. Vi sono poi le potenti unità Odroid, azienda sud-coreana, che produce SBPC molto potenti ma al contempo economiche [66], già usate in molti esperimenti di computer vision on board su MAV. Usando una di queste piattaforme pensata appositamente per dialogare con un Flight Controller si potrebbero implementare una serie di applicazioni (ad esempio, sistemi di *obstacle detection*, *indoor/outdoor autonoumus exploration*) che grazie alle informazioni ottenute da un algoritmo di V-SLAM guidi il mezzo di conseguenza. Da evitare possibilmente ogni soluzione basata su Windows. Questo infatti se pur essendo un ottimo sistema operativo per PC domestici non è adatto a scopi di ricerca e sviluppo. Molte delle librerie openSource vengono infatti principalmente sviluppate prima in Linux e poi portate su altri sistemi operativi, ma non sempre con buoni risultati. Il PTAM inoltre crea un mapping dell'ambiente. Questo mapping è riutilizabile per effettuare una ricostruzione tridimensionale, se pur grossolana dell'ambiente, che può divenire però una ricostruzione dettagliata mettendo in campo più di un MAV, coordinati da una logica, implementata per esempio con un linguaggio ad agenti, che occuperebbe il terzo livello dell'architettura proposta. Il PTAM è inoltre un algoritmo che stato sviluppato agli inizi della diffusione dei sistemi multi-core. Una ristrutturazione del sistema, realizzato seguendo un'attenta metodica di Ingegneria dei Sistemi Software, potrebbe portare giovamenti sull'efficienza complessiva del sistema massimizzando l'uso delle risorse.

Bibliography

[1] McFarland. (2014, Feb.) United arab emirates claims it'll make drone deliveries within a year. [Online]. Available: http://www.washingtonpost.com/blogs/innovations/wp/2014/02/10/united-arab-emirates-claims-itll-make-drone-deliveries-within-a-year/

[2] B. Dynamics. Bigdog - the most advanced rough-terrain robot on earth. [Online]. Available: http://www.bostondynamics.com/robot_bigdog.html

[3] Hanlon. (2005, Aug.) The gladiator: Us marinesúnmanned ground vehicle. [Online]. Available: http://www.gizmag.com/go/4484/

[4] Guizzo. (2010, Sep.) Autonomous vehicle driving from italy to china. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/robotics-software/autonomous-vehicle-driving-from-italy-to-china

[5] ——. Artificial vision and intelligent systems laboratory (vislab) of parma university (italy). [Online]. Available: http://vislab.it/whos-vislab/

[6] Maney. (2014, Jun.) Google has shown that self-driving cars are inevitable - and the possibilities are endless. [Online]. Available: http://www.independent.co.uk/life-style/motoring/features/google-has-shown-that-selfdriving-cars-are-inevitable–and-the-possibilities-are-endless-9547231.html

[7] Tregaskis. (2014, May) The first generation of selfdriving cars in pictures. [Online]. Available: http://www.theguardian.com/technology/gallery/2014/may/29/first-generation-self-driving-cars-google-in-pictures

[8] Kelly. (2014, Apr.) Driverless car tech gets serious at ces. [Online]. Available: http://edition.cnn.com/2014/01/09/tech/innovation/self-driving-cars-ces/

[9] Mercedes. Autonomous long-distance drive. [Online]. Available: http://www5.mercedes-benz.com/en/innovation/autonomous-long-distance-drive-research-vehicle-s-500-intelligent-drive/

[10] D. Paula. (2013, Sep.) Autonomous driving tech package will be an option on mercedes vehicles by 2020. [Online]. Available: http://www.forbes.com/sites/matthewdepaula/2013/09/30/autonomous-driving-will-become-an-option-on-regular-mercedes-models-by-2020/

[11] Lavrinc. (2014, Jul.) Bmw builds a self-driving car that drifts. [Online]. Available: http://www.wired.com/2014/01/bmw-builds-self-drifting-car/

[12] (2014, Jan.) Ces 2014: Bmw shows off 'drifting' self-drive cars. [Online]. Available: http://www.bbc.com/news/technology-25653253

[13] Hagon. (2013, Mar.) Driverless cars possible by 2015. [Online]. Available: http://www.drive.com.au/motor-news/driverless-cars-possible-by-2015-20130313-2fz96.html

[14] Lavrinc. (2013, Dec.) Ford unveils its first autonomous vehicle prototype. [Online]. Available: http://www.wired.com/2013/12/ford-fusion-hybrid-autonomous/

[15] Nissan-Innovations. The future is now. [Online]. Available: http://www.nissanusa.com/innovations/autonomous-car.article.html

[16] Simonite. (2013, Jan.) Toyota unveils an autonomous car, but says it'll keep drivers in control. [Online]. Available: http://www.technologyreview.com/news/509616/toyota-unveils-an-autonomous-car-but-says-itll-keep-drivers-in-control/

[17] T3. (2013, Jan.) Toyota unveils lexus 'driverless car'. [Online]. Available: http://www.t3.com/news/toyota-unveils-lexus-driverless-car

[18] Vijayenthiran. (2013, Aug.) Vw targets accident-free driving with autonomous tech. [Online]. Available: http://www.motorauthority.com/news/1086534_vw-targets-accident-free-driving-with-autonomous-tech

[19] VWERL. Autonomous driving. [Online]. Available: http://www.vwerl.com/our-work/view/34

[20] Wollaston. (2014, Jun.) Volvo's self-driving car: 'it's a slightly odd feeling' video. [Online]. Available: http://www.theguardian.com/technology/video/2014/jun/05/volvo-self-driving-car-video

[21] Spencer. (2014, May) Intel unveils new tech for self-driving cars. [Online]. Available: http://www.zdnet.com/intel-unveils-new-tech-for-self-driving-cars-7000030036/

[22] Clark. (2013, Jul.) States' self-driving car laws open door to more questions than answers. [Online]. Available: http://www.governing.com/news/state/sl-driverless-cards-open-door-to-questions.html

[23] Geomar. Auv abyss. [Online]. Available: http://www.geomar.de/en/centre/central-facilities/tlz/auv-abyss/overview/

[24] SeaRobotics. Products. [Online]. Available: http://www.searobotics.com/index.php/products/unmanned-surface-vehicles

[25] Arnsdorf. (2014, Feb.) Rolls-royce drone ships challenge $375 billion industry: Freight. [Online]. Available: http://www.bloomberg.com/news/2014-02-25/rolls-royce-drone-ships-challenge-375-billion-industry-freight.html

[26] (2014) Maritime unmanned navigation through intelligence in networks. [Online]. Available: http://www.unmanned-ship.org/munin/

[27] Boeing. X-37b overview. [Online]. Available: http://www.bloomberg.com/news/2014-02-25/rolls-royce-drone-ships-challenge-375-billion-industry-freight.html

[28] Cubesat kit. [Online]. Available: http://www.cubesatkit.com/

[29] ESA. Cubesats: Esa education. [Online]. Available: http://www.esa.int/Education/CubeSats

[30] Vorkoetter. (1997, Sep.) An electronic speed control primer. [Online]. Available: http://www.stefanv.com/electronics/escprimer.html

[31] Bachman. (2014, Jun.) Hollywood wants camera drones, and the faa might let them fly. [Online]. Available: http://www.businessweek.com/articles/2014-06-03/hollywood-wants-to-fly-camera-drones-and-the-faa-might-let-them

[32] Stelter. (2014, Jun.) Cnn to study drone use for reporting. [Online]. Available: http://money.cnn.com/2014/06/23/media/drones-news-research/

[33] DLR. Aerial manipulation. [Online]. Available: http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-7660/13033_read-32830/

[34] Guizzo. (2011, Dec.) Video: Watch flying robots build a 6-meter tower. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/diy/video-watch-flying-robots-build-a-6-meter-tower

[35] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice." *Knowledge Engineering Review*, vol. 10(2), p. 115–152, 1995.

[36] M. Magnusson, D. L. n; and P. Doherty, "Logical agents that plan, execute, and monitor communication," *Knowledge Engineering Review*, 2009.

[37] J. L. Ning Guo and J. Yu, "Research on the unmanned air vehicle team intelligence control system based on multi-agent," *Intelligent Robotics and Applications: First International Conference, ICIRA 2008 Wuhan, China, October 15-17, 2008 Proceedings*, vol. Part II, pp. 73–80, 2008.

[38] L. S. Mitch Ferry, Ziming Tu and G. Prickett, "Towards true uav autonomy," *2007 Infromation, Decision and Control*, 2007.

[39] M. T. Hama, "Uavas abstraction model: Jason agents as pilots and tripulation for unmanned aerial vehicles (uav)."

[40] S. Karim and C. Heinze, "Experiences with the design and implementation of an agent-based autonomous uav controller."

[41] Jack autonomous decision-making software. [Online]. Available: http://www.aosgrp.com/products/jack/

[42] Z. Ma and F. Hu, "Multi-agent systems formal model for unmanned ground vehicles."

[43] Ardupilot. [Online]. Available: http://ardupilot.com/

[44] Openpilot. [Online]. Available: http://www.openpilot.org/

[45] oddcopter. Flight controllers. [Online]. Available: http://oddcopter.com/flight-controllers/

[46] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The International Journal of Robotics Research*, vol. 5, no. 4, 1986.

[47] H. F. Durrant-Whyte, "Uncertain geometry in robotics," *Robotics and Automation, IEEE Journal of*, vol. 4, no. 1, pp. 23–31, Feb. 1988. [Online]. Available: http://dx.doi.org/10.1109/56.768

[48] *Simultaneous map building and localization for an autonomous mobile robot*, 1991.

[49] H. Durrant-Whyte, D. Rye, and E. Nebot, "Localization of autonomous guided vehicles," in *Robotics Research*. Springer London, 1996, pp. 613–625.

[50] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[51] N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, "The vslam algorithm for robust localization and mapping," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 24–29.

[52] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 6, pp. 1052–1067, 2007.

[53] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[54] F. Devernay and O. Faugeras, "Straight lines have to be straight: Automatic calibration and removal of distortion from scenes of structured enviroments," *Mach. Vision Appl.*, vol. 13, no. 1, pp. 14–24, Aug. 2001. [Online]. Available: http://dx.doi.org/10.1007/PL00013269

[55] C. Engels, H. Stewénius, and D. Nistér, "Bundle adjustment rules," in *Photogrammetric Computer Vision (PCV)*, Sep. 2006.

[56] J. Shi and C. Tomasi, "Good features to track," 1994, pp. 593–600.

[57] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time Onboard Visual-Inertial State Estimation and Self-Calibration of MAVs in Unknown Environments," in *ICRA 2012 : 2012 IEEE International Conference on Robotics and Automation : St. Paul, Minnesota, USA, May 14-18, 2012*. Piscataway, NJ: IEEE, 2012, pp. 957–964.

[58] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slam-based navigation for autonomous micro helicopters in gps-denied environments." *J. Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/jfr/jfr28.html#WeissSS11

[59] M. Achtelik, M. Achtelik, S. Weiss, and R. Y. Siegwart, "Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments," in *2011 IEEE International Conference on Robotics and Automation : (ICRA 2011) ; Shanghai, China, 9 - 13 May 2011*. Piscataway, NJ: IEEE, 2011, pp. 3056–3063.

[60] e volo. Vc200 – the first volocopter to carry two people. [Online]. Available: http://www.e-volo.com/ongoing-developement/vc-200

[61] Google. Snappy. [Online]. Available: https://code.google.com/p/snappy/

[62] OpenCV. Open source computer vision. [Online]. Available: http://opencv.org/

[63] ffmpeg. [Online]. Available: http://ffmpeg.org/

[64] Wikipedia. Sbpc. [Online]. Available: http://en.wikipedia.org/wiki/List_of_single-board_computers

[65] Emlid. Navio. [Online]. Available: http://www.emlid.com/

[66] Odroid. [Online]. Available: http://hardkernel.com/main/products/prdt_info.php

[67] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.