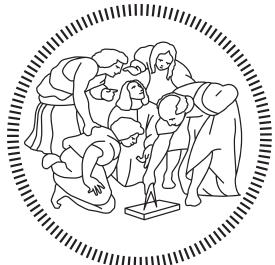


POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica e Informazione



**Un metodo per la predizione della
struttura di edifici parzialmente esplorati
per robot mobili autonomi**

Relatore: Prof. Francesco Amigoni
Correlatore: Dr. Matteo Luperto

Tesi di Laurea Magistrale di:
Valerio Arcerito, matricola 841951

Anno Accademico 2015-2016

Sommario

Uno degli obiettivi nell’ambito della robotica mobile consiste nella creazione e nello sviluppo di robot che siano in grado di muoversi ed operare in maniera efficiente nell’ambiente circostante. Un requisito fondamentale per raggiungere tale scopo consiste nel dotare un robot di autonomia, ossia fornire al robot capacità che gli permettano di compiere scelte, eseguire azioni e relazionarsi con l’ambiente esterno senza bisogno di un continuo intervento umano. Uno degli strumenti necessari per un robot autonomo è una rappresentazione dello spazio fisico nel quale esso si trova ed opera. Tale rappresentazione dell’ambiente consiste in una *mappa*.

Lo scopo di questo lavoro di tesi è sviluppare un sistema che fornisca ad un robot la capacità di stimare una probabile struttura di un ambiente, indoor, non ancora conosciuto, data una mappa non completa dell’ambiente stesso, basandosi interamente sui dati relativi alla porzione dell’ambiente già visitato in precedenza.

La mappa non completa dell’ambiente viene elaborata ricostruendo, a partire da essa, il *layout* delle stanze, il quale mostra la suddivisione geometrica e la disposizione delle stanze che compongono l’edificio. Il grado di astrazione fornito dal layout viene quindi utilizzato per fornire una predizione sulle stanze parzialmente esplorate, e quindi non totalmente viste, presenti nel layout stesso.

Le attività sperimentali hanno dimostrato che il sistema raggiunge un alto livello di accuratezza nel predire la forma e l’area delle porzioni di stanze che completano quelle parziali del layout.

Indice

Sommario	I
1 Introduzione	1
2 Stato dell'arte	5
2.1 Il paradigma Sense Plan Act	6
2.2 Una rappresentazione dell'ambiente: la mappa metrica	7
2.2.1 I sensori	7
2.2.2 La mappa metrica	8
2.2.3 L'esplorazione	10
2.3 La mappa multilivello	14
2.4 Il layout e la segmentazione	17
2.4.1 Il metodo basato sul grafo di Voronoi	18
2.4.2 Il metodo basato sul graph partitioning	19
2.4.3 Il metodo basato su feature	20
2.4.4 Il metodo basato su operatori morfologici	22
2.4.5 Il metodo basato sulla distance transform	23
2.4.6 Il metodo basato sulla ricostruzione del layout	25
2.5 La predizione	33
3 Impostazione del problema di ricerca	39
3.1 Lo scopo del lavoro	39
3.2 Le assunzioni	40
3.3 Il rapporto del lavoro con lo stato dell'arte	42
4 La predizione del layout	45
4.1 La mappa metrica	46
4.2 La creazione del layout	46
4.2.1 Riconoscimento dei segmenti	49
4.2.2 Segmenti estesi	51
4.2.3 Creazione celle	60

4.2.4	Riconoscimento del contorno	62
4.2.5	Clustering celle	63
4.3	La predizione	65
4.3.1	Pre-processing	67
4.3.2	Ricerca frontiere	71
4.3.3	Riconoscimento stanze parziali	72
4.3.4	Azioni geometriche	75
4.3.5	Azione geometrica complessa	77
5	Architettura del sistema	85
5.1	La mappa metrica ed il layout	85
5.1.1	La mappa metrica	86
5.1.2	Il layout	89
5.1.3	Visualizzazione dei dati	93
5.2	Architettura del modulo di predizione	94
5.2.1	Crea oggetto layout	95
5.2.2	Pre-processing	96
5.2.3	Ricerca Frontiere	96
5.2.4	Riconoscimento stanze parziali	97
5.2.5	Predizione	97
6	Realizzazioni sperimentali e valutazione	99
6.1	Il setting	99
6.2	Le metriche	102
6.2.1	Intersection over Union	103
6.2.2	Le metriche sulle frontiere	104
6.3	I risultati sperimentali	106
6.3.1	Test sulle aree	108
6.3.2	Intersection over Union	112
6.3.3	Primo test sulle frontiere	113
6.3.4	Secondo test sulle frontiere	116
6.3.5	Complessità temporale dell'azione complessa	117
6.4	Esempi del metodo	118
7	Direzioni future di ricerca e conclusioni	125
Bibliografia		129
A Risultati		139

Capitolo 1

Introduzione

Uno degli obiettivi della robotica mobile autonoma consiste nella creazione e nello sviluppo di robot che siano in grado di muoversi ed operare in maniera efficiente nell’ambiente circostante. Un requisito fondamentale per raggiungere tale scopo consiste nel dotare il robot di autonomia, ossia fornire al robot capacità che gli permettano di compiere scelte, di eseguire azioni e di relazionarsi con l’ambiente esterno senza bisogno di un continuo intervento umano. Queste abilità, nel lungo termine, potranno permettere di creare dei robot che siano in grado di aiutare, ed in alcuni casi sostituire, l’essere umano in compiti quotidiani come, ad esempio, robot di servizio per la pulizia di ambienti o robot in grado di operare in situazioni di pericolo e salvataggio (Urban Search and Rescue) [19].

Uno degli strumenti necessari per un robot autonomo è una rappresentazione dello spazio nel quale esso si trova ed opera. Tale rappresentazione dell’ambiente consiste in una *mappa* [89]. La mappa è tipicamente ricavata dai dati acquisiti tramite i sensori montati sul robot, quali ad esempio scanner laser, kinect o telecamere. Essa rappresenta la struttura di un ambiente, le porzioni libere dello spazio e quelle occupate da ostacoli. Senza la mappa, un robot mobile non sarebbe in grado di muoversi in modo efficiente nell’ambiente, poiché non sarebbe in grado di orientarsi ed anche un compito come quello di muoversi tra due punti prefissati risulterebbe complesso.

Affinché un robot sia in grado di acquisire autonomamente una mappa di un ambiente inizialmente ignoto, deve poterlo esplorare [99]. Durante il processo di esplorazione il robot non possiede una rappresentazione dell’ambiente completa, ma solo della porzione che, attraverso i suoi sensori, ha percepito fino a quel momento.

Lo scopo di questo lavoro di tesi è sviluppare un sistema che fornisca

ad un robot la capacità di stimare una probabile struttura di un ambiente, indoor, non ancora conosciuto, data una mappa non completa dell’ambiente stesso, basandosi interamente su dati provenienti dalla porzione dell’ambiente già visitato in precedenza. La predizione della struttura di un ambiente parzialmente esplorato, come il completamento della struttura di una stanza parzialmente vista, può essere usata per migliorare l’esplorazione, identificando quei luoghi che potrebbero essere non particolarmente informativi e quelli che, invece, potrebbero fornire maggiori informazioni sull’ambiente ignoto, se visitati.

Il punto da cui parte il sistema realizzato in questo lavoro di tesi corrisponde alla mappa metrica parziale. Questa mappa è il risultato di una esplorazione non ancora terminata di un ambiente indoor (e, quindi, fortemente strutturato) e viene costruita acquisendo e integrando dati tramite i sensori montati sul robot. Oltre che alle informazioni di carattere metrico, che riguardano l’occupazione o meno di un’area da parte di un ostacolo, all’interno di una mappa metrica parziale si possono trovare anche informazioni riguardanti le *frontiere*, ossia i confini tra le regioni di spazio già esplorate e quelle non esplorate dal robot, come mostrato in [99].

Il livello di conoscenza fornito dalla mappa metrica viene in seguito raffinato costruendo il *layout* delle stanze, definito come in [13], che a sua volta trae ispirazione da [62]. Il layout delle stanze rappresenta il risultato di un processo di astrazione dell’ambiente che elabora le informazioni metriche e che ha come obiettivo quello di mostrare la suddivisione e disposizione delle stanze che compongono l’edificio.

Il grado d’astrazione fornito dal layout viene quindi utilizzato per calcolare una predizione sulle stanze parziali, ossia quelle stanze nelle quali viene riscontrata la presenza di una frontiera. L’approccio proposto per la generazione di una predizione consiste nel formulare un insieme di ipotesi di completamento per ogni stanza parziale. Le ipotesi create vengono valutate sulla base di una *score function*, permettendo così di selezionare l’ipotesi che è più coerente con il layout stesso.

Il lavoro di tesi nasce dal tentativo di superare alcune delle limitazioni del lavoro presentato in [50] che, seppur in grado di ricostruire il layout di una mappa di un ambiente quasi interamente esplorato, esclude dalla soluzione finale stanze parzialmente viste. Al contrario, in questo lavoro di tesi, l’esistenza di porzioni osservate permette di affermare che la stanza, nella quale viene riscontrata tale osservazione, esiste realmente.

Le attività sperimentali svolte per valutare il sistema oggetto di questa tesi, oltre a valutare la forma e l’area delle stanze predette, vertono anche

sul valutare la scelta della miglior frontiera all'interno di una mappa metrica parziale, ossia quella frontiera che potrebbe portare ad una migliore osservazione dell'ambiente in un processo di esplorazione. La valutazione mostra che la predizione della struttura dell'ambiente raggiunge un alto livello di accuratezza e che nella maggioranza dei casi, applicando una predizione dell'ambiente, è possibile selezionare la miglior frontiera da esplorare.

La tesi è strutturata nel modo seguente.

Nel Capitolo 2 viene descritto lo stato dell'arte relativo al mapping nell'ambito della robotica mobile autonoma in ambienti indoor, mostrandone gli approcci principali. Si pone inoltre una particolare attenzione sul significato di predizione di ambienti non ancora del tutto esplorati e come questa sia stata affrontata in lavori precedenti.

Nel Capitolo 3 si descrive lo scopo del lavoro, le assunzioni fatte ed il rapporto del lavoro svolto con lo stato dell'arte.

Nel Capitolo 4 viene descritto in maniera dettagliata l'approccio logico dello sviluppo del sistema, illustrando ogni passaggio che porta ad una predizione delle stanze parziali.

Nel Capitolo 5 viene mostrata l'architettura del progetto svolto. Vengono descritti i moduli che la compongono e le loro interazioni dal punto di vista tecnologico ed implementativo.

Nel Capitolo 6 vengono illustrate le prove sperimentali effettuate sull'implementazione del sistema sviluppato. Vengono inizialmente descritte le metriche usate per la valutazione delle performance e l'impostazione generale secondo cui sono state svolte le prove sperimentali. Successivamente vengono mostrati i test effettuati sul sistema, fornendo misure sia qualitative (mostrando esempi visuali) che quantitative (secondo le metriche introdotte) che valutano il sistema nel suo complesso.

Nel Capitolo 7 viene riassunto il contenuto della tesi e vengono tratte alcune conclusioni riguardanti il lavoro svolto. Il capitolo si conclude elencando alcuni spunti per sviluppi futuri.

In Appendice A sono mostrati alcuni risultati aggiuntivi ottenuti nelle prove sperimentali svolte.

Capitolo 2

Stato dell'arte

In questo capitolo viene fornita una descrizione dello stato dell'arte riguardante il mapping, nell'ambito della robotica autonoma in ambienti interni.

Un robot al fine di essere autonomo deve interagire con l'ambiente che lo circonda. Questa interazione richiede che il robot percepisca e costruisca una rappresentazione dell'ambiente. Possedere un modello dell'ambiente è necessario al fine di potervici operare efficacemente all'interno e quindi eseguire molti compiti, tra i quali si possono citare trasporto, pulizia [9] ed USaR (Urban Search and Rescue) [19].

La forma primaria di rappresentazione di un ambiente è la *mappa*. Esistono diversi tipi di rappresentazioni dell'ambiente, cui si riferiscono diversi tipi di mappe. Il mapping consiste nell'integrare le informazioni raccolte dai sensori del robot mobile all'interno di una data rappresentazione, creando così la mappa. Durante il mapping si focalizza l'attenzione su come l'ambiente debba essere rappresentato e su come interpretare i dati provenienti dai sensori del robot.

Il mondo circostante può essere rappresentato in diversi modi. Le rappresentazioni possono essere utilizzate singolarmente o integrate una sull'altra all'interno di un'unica mappa, denominata mappa multilivello.

Di seguito viene fornita un'introduzione, non esaustiva, di alcuni metodi di rappresentazione. Nella maggior parte dei lavori svolti nello stato dell'arte, la rappresentazione dell'ambiente si riferisce solo ed esclusivamente alla parte del mondo circostante già percepita dal robot. Questo significa che non vi è alcuna conoscenza delle parti di ambiente che il robot potrebbe vedere in futuro. A volte però, possedere informazioni sulle parti di ambiente sconosciute potrebbe essere utile al fine di fornire una migliore e più robusta rappresentazione dell'ambiente circostante [48].

Il lavoro oggetto di questa tesi verte a riconoscere tutte le informa-

zioni utili, che vengono, in seguito, utilizzate per fornire una predizione dell’ambiente circostante non ancora esplorato dal robot mobile.

2.1 Il paradigma Sense Plan Act

Come un essere umano percepisce l’ambiente attraverso i propri sensi, anche un robot, seguendo il medesimo comportamento, percepisce l’ambiente attraverso i propri sensori. Se, in linea di massima, è possibile dire che un essere umano è in grado di interpretare quasi immediatamente le informazioni acquisite tramite i propri sensi, lo stesso non si può affermare per un robot. Infatti, tendenzialmente i dati ottenuti direttamente dai sensori del robot, affinché possano essere utilizzati, devono prima essere interpretati e compresi.

Una volta compreso il significato dei dati acquisiti è dunque possibile pianificare una risposta ed in seguito eseguirla, grazie all’utilizzo degli attuatori. Questo comportamento viene descritto dal paradigma *Sense Plan Act* [97], un modello mentale che rappresenta un concetto fondamentale, che ricorda le tre capacità che ogni robot deve possedere per operare efficientemente:

- **Sense.** Il robot deve avere l’abilità di percepire l’ambiente in cui opera e raccogliere informazioni su di esso, come la presenza di ostacoli o altre informazioni utili alla navigazione. Con navigazione si intende l’abilità di un robot di determinare la propria posizione all’interno di un quadro di riferimento¹ e di pianificare successivamente un percorso che abbia come obiettivo il raggiungimento di un luogo specifico. In questa fase la conoscenza dell’ambiente è limitata alle sole parti dell’ambiente osservate fino a quel momento.
- **Plan.** Ottenuti i dati provenienti dalla fase precedente il robot deve essere in grado di analizzarli e rappresentarli appropriatamente, per poter in seguito determinare una serie di azioni che consentano al robot di raggiungere il proprio obiettivo. L’obiettivo viene stabilito in base ai task che il robot deve compiere.
- **Act.** Infine, il robot deve essere in grado di agire eseguendo le azioni, che la fase di Plan aveva previsto.

Ogni fase è dipendente da quella precedente e l’intero processo viene iterato per tutta la durata di una missione, come si può vedere in Figura 2.1.

¹Il quadro di riferimento consiste in un sistema astratto di coordinate che fissano punti di riferimento fisici al suo interno e standardizzano le misurazioni.

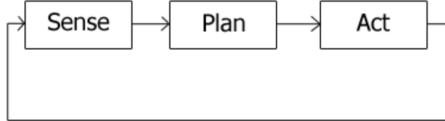


Figura 2.1: In figura viene mostrato come le fasi nel paradigma Sense Plan Act interagiscono tra di loro.

Scopo di questo lavoro di tesi è fornire un metodo che permetta di estendere la fase di Sense non solo alla parte dell’ambiente che è già stata osservata dal robot, ma anche ad ambienti solo parzialmente percepiti, al fine di poter fornire maggiori informazioni alla fase di Plan. Ciò verrà realizzato facendo inferenza sui dati ottenuti nella fase di Sense.

2.2 Una rappresentazione dell’ambiente: la mappa metrica

Un esempio tipico di come il robot interpreta ed utilizza i dati raccolti dai sensori è quello della costruzione di una mappa che rappresenta l’ambiente.

2.2.1 I sensori

I robot, perché possano raccogliere dati esterni, devono essere muniti di appositi sensori. Ne esistono di diversi tipi, tra i quali: i sensori di basso livello come i sensori di prossimità ad ultrasuoni [24], magnetici e ottici, con i quali il robot è in grado di rilevare oggetti nelle immediate vicinanze ed evitare di conseguenza eventuali collisioni; i sensori laser, con i quali è possibile rilevare modelli tridimensionali di oggetti con risoluzioni differenti e sensori più avanzati usati per la visione, come i sensori RGBD [18], con i quali è possibile ricostruire l’aspetto e la forma di un oggetto all’interno dell’ambiente. Questo avviene catturando immagini RGB (Red Green Blue) ed informazioni sulla profondità (depth, D) di ogni pixel.

Sensori differenti catturano aspetti diversi dell’ambiente. Usare fonti multiple per ottenere dati può portare ad una migliore accuratezza nella rappresentazione della mappa. Ad esempio i sensori visivi sono una fonte insostituibile di informazioni distintive del luogo. Queste informazioni risultano complesse da analizzare, poiché sono sensibili, ad esempio, ai cambi di luminosità e di illuminazione dovuti all’alternarsi tra giorno e notte, luce e ombra o cambi di stagione. Così, accanto ai sensori visivi, i robot sono spesso equipaggiati anche con scanner laser. L’integrazione di sensori diversi in

un robot consente che le informazioni geometriche ricavate siano più stabili e robuste. Il vantaggio nell'integrare dati provenienti da sorgenti differenti è presente nel lavoro svolto in [72]. Quando si integrano informazioni provenienti da più sensori diversi fra loro nell'elaborazione di una mappa, si parla di *sensor fusion*.

2.2.2 La mappa metrica

La *mappa metrica* rappresenta il più basso grado di astrazione nella rappresentazione dell'ambiente, da essa è possibile ricavare informazioni sull'occupazione delle aree di un ambiente e le distanze tra gli oggetti all'interno. La mappa metrica costituisce una delle informazioni più importanti per garantire un comportamento autonomo del robot, in quanto consente di stimare la posizione del robot mobile all'interno della mappa stessa.

In alcuni casi, come in questo lavoro di tesi, la mappa metrica può essere nota a priori. Nel caso in cui la mappa metrica non sia nota a priori, essa può essere ricavata attraverso la tecnica dello SLAM (Simultaneous Localization And Mapping) [81], che tratta il problema dell'integrazione dei dati per la costruzione di una mappa metrica in ambiente sconosciuto. Durante la navigazione, il robot acquisisce informazioni sull'ambiente costruendo la mappa, e al contempo cerca di localizzarsi all'interno. Lo SLAM può dare risposta ad una necessità, che riguarda contemporaneamente l'interesse ad ottenere un modello accurato dell'ambiente e a localizzare la posizione di un robot mobile all'interno dell'ambiente stesso. La localizzazione consiste nell'abilità di un robot di ricostruire con buona approssimazione la propria posizione e il proprio orientamento all'interno di un sistema di riferimento. Un lavoro nel quale si risolve il problema della localizzazione è [6], nel quale si fa uso di specifici marcatori, chiamati *landmark*, presenti all'interno della mappa dell'agente. Questi aiutano il robot a trovare la propria posizione all'interno della mappa stessa.

Nonostante esistano differenti metodi di rappresentazione per una mappa metrica, una di quelle più utilizzate in letteratura è la mappa a griglia o *occupancy grid map* [89]. Questo tipo di rappresentazione modella l'ambiente come una matrice bidimensionale [87]. La probabilità che una cella sia occupata o meno viene rappresentata dal valore assegnatogli, che può variare tra zero ed uno. La Figura 2.2 mostra due esempi di mappa a griglia. Ulteriori informazioni sulle mappe a griglia possono essere trovate nei lavori [58], [71] e [27].

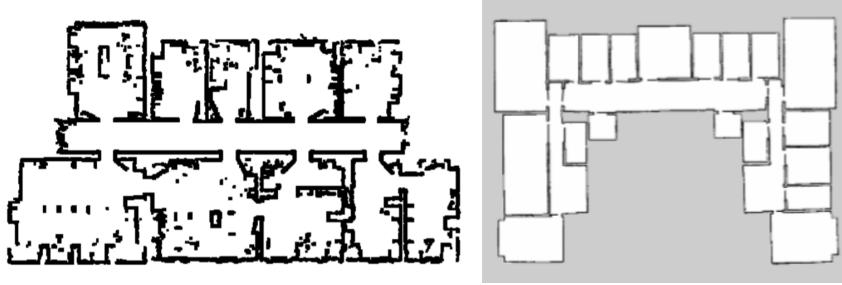


Figura 2.2: Due esempi differenti di mappa a griglia. A sinistra le celle occupate e quelle libere vengono rappresentate utilizzando rispettivamente i due colori opposti nero e bianco, tratto da [59]. A destra, invece, vengono indicati i vari gradi di occupazione di una cella usando una scala di grigi, nella quale i colori bianchi rappresentano regioni di spazio libero, il nero rappresenta regioni di spazio occupate dagli ostacoli ed il grigio quelle sconosciute.

Una diversa metodologia per rappresentare le mappe metriche consiste nell'aggiungere feature, come punti e segmenti, all'ambiente, man mano che queste vengono acquisite dai sensori del robot. L'insieme delle feature viene poi ancorato ad un sistema di riferimento assoluto. Alcuni esempi sulla creazione di segmenti, i quali rappresentano i bordi degli ostacoli, possono esser trovati nei lavori [39], [45] e [100], nei quali si fornisce un metodo di raggruppamento in cluster di punti provenienti da una scansione laser, che verranno in seguito approssimati con dei segmenti.

Le mappe metriche appena citate forniscono un tipo di rappresentazione basica, non vi è infatti alcun riferimento tra una porzione all'interno della mappa e ciò che essa rappresenta. Con ciò si intende che non vi è alcuna differenza tra un corridoio ed una stanza qualunque. Tuttavia le mappe metriche contengono informazioni spaziali sulla struttura del mondo e possono essere dunque utilizzate per la localizzazione o la pianificazione di una traiettoria priva di ostacoli (*path planning*). Il path planning consiste nell'abilità di un robot di pianificare un percorso, che parta dalla propria posizione, ottenuta in fase di localizzazione, e che termini con la posizione di uno specifico goal, determinato a priori. Lo stato dell'arte in questo campo focalizza l'attenzione principalmente sulla ricerca del percorso più breve, che sia al contempo privo di collisioni con ostacoli. Alcuni approcci al path planning vengono descritti in [47] e [38].

La navigazione di un ambiente incognito viene fatta attraverso l'esplorazione, che rappresenta l'insieme degli approcci con i quali il robot decide come muoversi in un ambiente sconosciuto.



Figura 2.3: Esempio di frontiere. I pixel colorati, non con scale di grigio, rappresentano le frontiere della mappa metrica.

2.2.3 L'esplorazione

L'esplorazione robotica è un campo di ricerca nel quale è stato svolto un considerevole lavoro. L'esplorazione robotica venne inizialmente condotta in ambienti simulati [44] o con robot che osservavano l'ambiente controllati da un essere umano [25]. L'innovazione in quest'ambito ha in seguito portato a sistemi d'esplorazione autonoma più avanzati, con robot reali in grado di muoversi ed osservare autonomamente l'ambiente circostante. Affinché un robot sia in grado di acquisire autonomamente una mappa di un ambiente non ancora conosciuto e fare quindi mapping, deve dunque poter esplorare. Nei lavori [88] e [55] vengono descritti approcci per l'esplorazione.

Un esempio molto usato come metodo di esplorazione è il *frontier-based exploration* [99], nel quale si tenta di ottenere il maggior numero di informazioni possibili sull'ambiente muovendosi verso una *frontiera*, ovvero il confine tra le regioni di spazio già esplorate e quelle non ancora esplorate dal robot. Siccome i sensori di un robot hanno un range limitato, le frontiere si creano dove non esistono ostacoli all'interno del raggio d'azione del sensore. I pixel colorati nelle immagini in Figura 2.3 mostrano un esempio di cosa sia una frontiera.

Perché un robot mobile possa muoversi verso una frontiera, è necessario che la riconosca. Il lavoro in [99], come in questo lavoro di tesi, assume una rappresentazione dello spazio a griglia. Ciò rende possibile classificare una cella come:

- **libera:** una cella nella quale è stato riscontrato uno spazio libero;

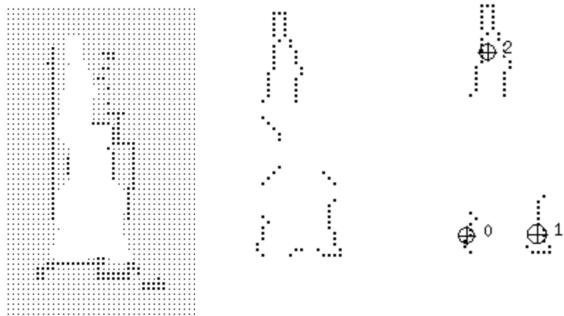


Figura 2.4: L'intero processo come descritto in [99] permette di riconoscere tutte le porzioni di una mappa a griglia, nella quale esiste una possibile frontiera; per poi selezionare tutte e solo le frontiere aventi una dimensione maggiore di una soglia minima.

- **occupata:** una cella nella quale è stato riscontrato un ostacolo;
- **sconosciuta:** una cella non ancora esplorata.

In questo modo ogni cella aperta adiacente ad una cella sconosciuta viene etichettata come frontiera. La Figura 2.4 mostra un esempio. L'immagine a sinistra rappresenta la mappa metrica a griglia, nella quale è possibile distinguere chiaramente quali siano le celle della griglia. L'immagine nel centro rappresenta tutte quelle celle che potrebbero rappresentare una frontiera. Infine, l'immagine a destra rappresenta tutte e solo le frontiere la cui lunghezza supera una determinata soglia, stabilita a priori, che corrisponde ad esempio alla dimensione del robot.

Un robot che utilizza una metodologia d'esplorazione basata sulle frontiere ha il vantaggio di poter esplorare efficientemente, muovendosi verso quei luoghi che hanno una maggior probabilità di aggiungere nuove informazioni alla mappa, aumentando la propria conoscenza del mondo.

Il lavoro in [85] compie il primo passo verso un tipo di esplorazione nella quale si sfrutta una pregressa conoscenza dell'ambiente durante l'esplorazione, usando ambienti precedentemente esplorati e conosciuti, per ragionare su cosa si possa trovare nella parte sconosciuta di un ambiente in fase di esplorazione. La mappa dell'ambiente vista dal robot viene fusa con una porzione di mappa di un ambiente precedentemente esplorato, la quale presenta delle similarità con le parti già viste dell'ambiente che il robot sta esplorando. Questo tipo di informazione consente di predire possibili strutture. L'immagine a destra di figura 2.5 mostra un esempio di una possibile struttura (in rosso), sovrapposta alla mappa fino a quel momento esplorata

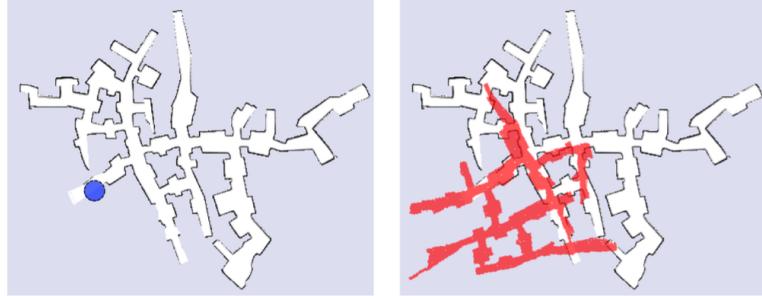


Figura 2.5: Le due immagini rappresentano entrambe una mappa di un ambiente parzialmente esplorato. Nell'immagine a sinistra il cerchio in blu evidenzia la frontiera presa in considerazione. Nell'immagine a destra viene mostrata una possibile struttura, in rosso, associata alla frontiera considerata e sovrapposta alla mappa.

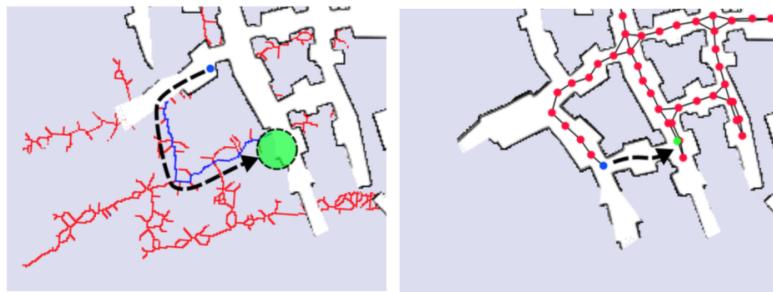


Figura 2.6: Esempio di *loop-closure*. Il percorso blu nell'immagine a sinistra rappresenta una possibile predizione con *loop-closure*. A destra, il robot esplora seguendo la struttura e percepisce la reale struttura dell'ambiente.

e associata alla frontiera indicata dal cerchio blu nell'immagine a sinistra. Le strutture vengono poi usate per valutare le frontiere, consentendo al robot di raggiungere la migliore. Le frontiere vengono valutate in base alla probabilità che esse conducano ad un *loop-closure*. Ciò significa che il robot, pianificando un percorso che parte da una frontiera all'interno della parte esplorata e seguendo la struttura, sia in grado di ritornare all'interno della mappa passando per un'altra frontiera, diversa da quella precedente. La Figura 2.6 mostra un esempio di *loop-closure*.

Il lavoro in [33] affronta i problemi di navigazione e di localizzazione in ambiente sconosciuto introducendo il concetto di *safe region*. Queste sono le regioni più grandi che si garantisce siano libere e sono individuate direttamente sui dati ottenuti dalla lettura dei sensori. Grazie all'uso delle *safe region* gli autori hanno proposto un algoritmo che costruisca iterativamente una mappa, applicando l'unione di *safe region* successive. Inoltre le *safe*

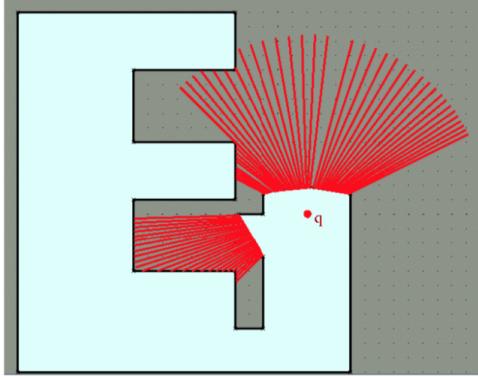


Figura 2.7: Esempio tratto da [33]. Il potenziale information gain di un candidato q è l'area $A(q)$ della regione esterna alla safe region corrente, che potrebbe essere visibile attraverso le frontiere. Questa area viene rappresentata dall'unione dei segmenti rossi.

region possono essere usate per calcolare la posizione di grandi aree inesplose. Ad ogni posizione q , che il robot potrebbe raggiungere per completare l'esplorazione acquisendo nuovi dati, viene attribuito un valore $A(q)$. Questo valore è strettamente dipendente dal guadagno stimato in termini di area aggiunta alla mappa. Il punto rosso interno alla safe region denominato dalla lettera q in Figura 2.7 rappresenta la posizione dalla quale viene calcolata $A(q)$. Essa viene calcolata emanando un numero finito di raggi, con centro in q e lunghezza pari all'estensione massima raggiungibile dal sensore montato sul robot. Per ogni raggio:

- Si considera il segmento di lunghezza r_{max} partendo da q . Se il segmento interseca un muro precedentemente osservato viene eliminata la porzione di segmento successiva all'intersezione.
- Si calcola la lunghezza della porzione di raggio rimanente che risiede fuori dalla safe region.

$A(q)$ viene stimata sommando tutte le lunghezze ricavate.

L'esplorazione di ambienti sconosciuti può essere anche effettuata simultaneamente da robot differenti, che collaborano per ottenere una rappresentazione dell'ambiente, come in [98], [43] e [84]. Questo processo prende il nome di esplorazione multirobot. L'uso di più robot può fornire diversi vantaggi rispetto a sistemi con un singolo robot [22], poiché la cooperazione robotica permette di compiere uno specifico task in un tempo minore [83]. Ad esempio, in [84] si può vedere come il tempo necessario per compiere il task di esplorazione venga considerevolmente ridotto, assegnando ad ogni

robot che fa parte del team una differente posizione target potenzialmente esplorabile, distribuendo così il team di robot all'interno dell'ambiente.

2.3 La mappa multilivello

Fino ad ora si è parlato di mappa metrica, una rappresentazione dell'ambiente che permette al robot di compiere i task di localizzazione e navigazione. Questo tipo di rappresentazione, infatti, non è in grado di fornire alcun tipo di associazione tra una porzione della mappa e ciò che essa rappresenta. Le uniche informazioni che fornisce una mappa metrica indicano, infatti, se una determinata posizione sia occupata o meno, senza specificare chi o cosa occupi quella posizione.

Come un umano è in grado di capire l'ambiente, categorizzando concettualmente lo spazio nel quale si trova, anche un robot, perché incrementi le proprie abilità autonome, deve essere in grado di farlo. Uno dei mezzi più usati in robotica prevede l'utilizzo di diversi livelli d'astrazione, nei quali si forniscono al robot informazioni differenti. In questo caso, come in [71], si parla di *sistemi di rappresentazione multilivello*. La Figura 2.8 mostra un esempio di architettura multilivello. Il livello più basso della rappresentazione multilivello proviene dai dati sensoriali, con i quali si costruisce la mappa metrica descritta nella sezione precedente, rappresentata dall'immagine in basso della Figura 2.8. Ogni livello della mappa multilivello viene costruito a partire dalle informazioni raccolte nel livello precedente. La Figura 2.9 fornisce un esempio che mette in mostra la sequenza con la quale si ottengono i livelli della mappa multilivello.

Nel lavoro presentato in [100] e [71] i livelli di rappresentazione successivi alla mappa metrica sono:

- **Navigation graph.** Il grafo di navigazione fornisce un modello dello spazio libero e della sua connettività. È basato sul concetto di *road-map*, come descritto in [46] e [63], definito come un insieme di marker, chiamati nodi di navigazione, che vengono ancorati alla mappa metrica ognuno ad una distanza predefinita dal marker precedente. I nodi di navigazione sono connessi seguendo l'ordine secondo cui vengono generati. L'ordine viene definito dalla traiettoria seguita dal robot durante il processo di esplorazione. In questa fase, inoltre, la mappa metrica viene arricchita inserendo informazioni semantiche sull'ambiente. Con l'utilizzo di algoritmi di classificazione, vengono estratte le categorie semantiche degli spazi, associando una porzione dello spazio ad un'e-

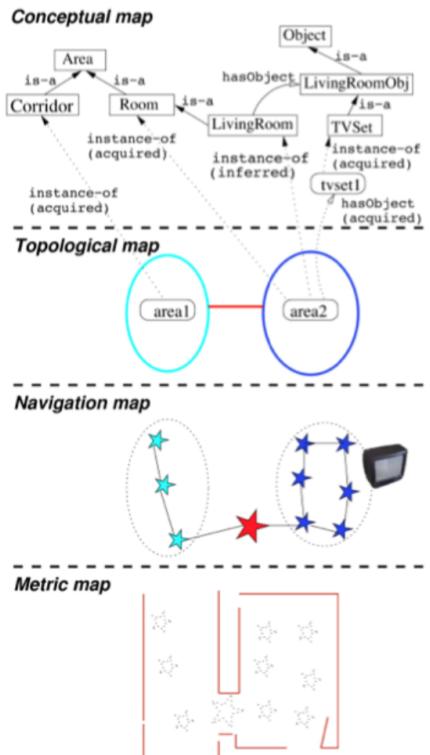


Figura 2.8: In figura viene mostrato un esempio di un sistema multilivello, come tratto da [71].

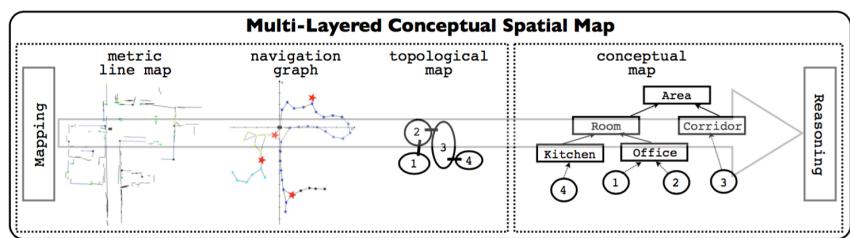


Figura 2.9: In figura viene mostrato un esempio della sequenza con la quale si ottengono i livelli di un sistema multilivello, come tratto da [100].

tichetta che ne indica il ruolo all'interno dell'ambiente stesso, come ad esempio differenziare un punto all'interno di una stanza o di un corridoio. Queste informazioni vengono salvate all'interno dei nodi del grafo di navigazione. Ogniqualvolta il robot attraversa un passaggio stretto, vengono inoltre aggiunti nodi al grafo di navigazione, i quali rappresentano le porte.

- **Mappa topologica.** Il grafo di navigazione fornisce la base per ottenere la mappa topologica. La mappa topologica viene rappresentata con un grafo, nel quale ogni nodo rappresenta una stanza e gli archi le varie connessioni tra di esse. La mappa topologica è costruita raggruppando i nodi del grafo di navigazione, che non sono separati dai nodi che rappresentano le porte. Nella mappa topologica raffigurata in Figura 2.8, l'area 1 rappresenta il raggruppamento dei nodi azzurri del grafo di navigazione, mentre l'area 2 rappresenta il raggruppamento dei nodi blu del medesimo grafo. Il nodo del grafo di navigazione rappresentato con una stella rossa in Figura 2.8 indica un passaggio stretto, dunque una probabile porta, sulla base della quale è stata attuata la separazione in aree differenti della mappa topologica. Questo livello d'astrazione equivale alla capacità di un essere umano di classificare lo spazio in cui si trova, come ad esempio riconoscere di trovarsi all'interno di una stanza o di un corridoio. In questo livello le informazioni semantiche raccolte sull'ambiente vengono valutate per suddividere le aree della mappa topologica in precise categorie semantiche, che permettono di classificare regioni differenti in stanze o corridoi.

Le aree topologiche, formate in questa fase della mappa multilivello, vengono passate alla mappa concettuale, dove sono rappresentate come istanze delle rispettive categorie.

- **Mappa concettuale.** La mappa concettuale rappresenta il livello d'astrazione più alto e fornisce un'ulteriore interpretazione della mappa metrica. Le aree topologiche con le proprie categorie spaziali formano le entità spaziali di base, come ad esempio i corridoi e le stanze. Per inferire maggiori informazioni semantiche alle aree, viene utilizzato un *reasoner* logico. Il reasoner integra la conoscenza, estratta dall'ambiente con un'ontologia concettuale sugli ambienti, che definisce le relazioni tra oggetti e stanze e come queste possano essere collegate tra loro, dal momento che esiste una forte connessione tra oggetti tipici di una stanza e significato semantico della stanza stessa (ad esempio

è molto facile che un piatto si trovi in cucina rispetto che in bagno, o che un forno sia in cucina e non in camera da letto). Questo livello può essere usato per valutare le aspettative su quali oggetti possono essere rilevati all'interno di una determinata stanza, posto che sia noto il concetto base della stanza.

Un altro lavoro svolto con l'obiettivo di fornire una rappresentazione multilivello dell'ambiente è [13], nel quale i livelli d'astrazione sono rispettivamente: mappa metrica, layout delle stanze, grafo topologico e mappa semantica. In [13], partendo da una mappa metrica, viene costruita un'ulteriore rappresentazione nella quale è riportata la suddivisione dell'ambiente in stanze, che prende il nome di layout. Successivamente il grado d'astrazione viene incrementato attraverso la costruzione di un grafo topologico in cui i nodi rappresentano le stanze e gli archi le rispettive connessioni. Infine, l'ultimo livello classifica semanticamente le stanze, associando ai nodi del grafo un'etichetta, che ne rappresenta la tipologia della regione di spazio (stanza, corridoio, ecc.).

Il layout e la mappa metrica a griglia, citata nel paragrafo precedente, possiedono un ruolo fondamentale nel lavoro svolto in questo lavoro di tesi, poiché rappresentano gli input del sistema di predizione. Pertanto viene fornita una descrizione dello stato dell'arte riguardante gli approcci per ottenere il layout.

2.4 Il layout e la segmentazione

Il layout di un edificio è una rappresentazione geometrica che mostra la disposizione e suddivisione delle stanze all'interno; includendo informazioni su muri, stanze e porte. Ogni stanza della mappa può essere rappresentata sia come un poligono, nel caso bidimensionale, sia come serie di piani o *box model*², nel caso tridimensionale. Il layout di un edificio, quindi, rappresenta le stanze che ne compongono la struttura senza considerare l'arredamento, i dati mancanti o rumorosi e le occlusioni che a volte sono presenti all'interno della mappa metrica.

La *ricostruzione del layout* è il task che consente di ottenere il layout partendo da una rappresentazione metrica dell'ambiente e può essere svolta sull'intero edificio o su una singola stanza partendo da una mappa metrica a griglia nel caso bidimensionale, come in [50] o da *point cloud* (perfettamente

²Il box model rappresenta un modello, nel quale le stanze che compongono l'edificio vengono rappresentate unendo i poligoni del pavimento e del soffitto paralleli tra loro, con poligoni ortogonali ad essi, i quali rappresentano le pareti.

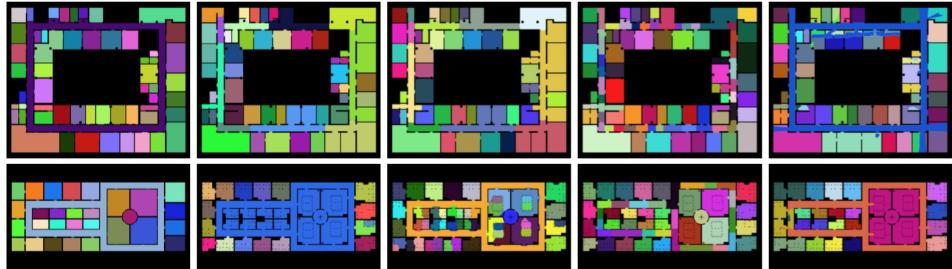


Figura 2.10: Esempio di segmentazione tratta da [10]: la prima colonna rappresenta una perfetta segmentazione, chiamata ground truth, ottenuta da un'etichettatura umana. La seconda colonna mostra il risultato di una segmentazione morfologica. La terza si basa su un approccio di segmentazione distance-based. La quarta mostra una segmentazione basata sul grafo di Voronoi e infine la quinta colonna si basa su un approccio di segmentazione feature-based.

allineati), nel caso tridimensionale, come nei lavori [3] e [62]. I metodi di ricostruzione del layout forniscono, partendo dalla mappa metrica, una rappresentazione più astratta della struttura dell’edificio, in modo da poter ragionare su una conoscenza più “pulita” e stabile.

Un processo strettamente legato alla ricostruzione del layout che prende il nome di *segmentazione* [10], è stato sviluppato per separare la mappa metrica in parti differenti, ognuna corrispondente ad una stanza diversa; ad esempio identificando passaggi stretti come se fossero porte e dividendo di conseguenza le due stanze connesse. La segmentazione etichetta i pixel della mappa metrica per fare in modo che quelli all’interno della stessa stanza siano etichettati nello stesso modo. Dunque, il processo di segmentazione non è altro che un raggruppamento, in cluster, dei dati in ingresso. Ciò corrisponde al partizionamento dell’ambiente in regioni distinte. In [93], [62] e [92] possono essere trovati ulteriori lavori, nell’ambito della segmentazione.

Di seguito vengono presentati i metodi di segmentazione e ricostruzione del layout più rilevanti. La Figura 2.10 mostra i risultati ottenuti da alcuni metodi di segmentazione, che andremo a illustrare nelle prossime sezioni di questo lavoro di tesi.

2.4.1 Il metodo basato sul grafo di Voronoi

Come viene affermato in [10], il *Voronoi-based partitioning* è probabilmente il metodo di segmentazione più popolare. Il grafo di Voronoi è una partizione spaziale di una mappa metrica. Viene costituito dai punti dello spazio libero nella mappa metrica che hanno più di un ostacolo alla stessa distanza minima. In Figura 2.11 in alto a sinistra si può vedere un esempio di grafo

di Voronoi. Questo approccio viene solitamente applicato direttamente sulla occupancy grid map. Un esempio si può trovare in [87], nel quale gli autori hanno deciso di segmentare la mappa usando un grafo di Voronoi, dal quale hanno estratto specifici punti, detti *punti critici*, che vengono usati per ottenere delle *linee critiche*.

Seguendo la definizione proposta in [88] e [87] si ha che:

- **Grafo di Voronoi.** Per ogni punto p nello spazio libero, esistono uno o più punti nello spazio occupato che si trovano il più vicino possibile al punto p . Questi punti prendono il nome di *basis point* di p . Il grafo di Voronoi è l'insieme di punti nello spazio libero che hanno almeno due basis point differenti.
- **Punti critici.** I punti critici sono punti del grafo di Voronoi che sono più vicini agli ostacoli rispetto ad altri punti. Questi punti possono essere usati per identificare passaggi stretti come le porte.
- **Linee critiche.** Le linee critiche si ottengono connettendo ogni punto critico con i propri basis point. Le linee critiche vengono usate per segmentare lo spazio libero in un certo numero di regioni, che verranno in seguito unite.

Ulteriori esempi sul grafo di Voronoi possono essere trovati in [98], [5], [29] e [41]. In Figura 2.11 si può trovare un esempio tratto dal lavoro svolto in [10].

2.4.2 Il metodo basato sul graph partitioning

Un ulteriore approccio usato per la segmentazione della mappa metrica consiste nell'utilizzo di grafi. Questo approccio può essere usato anche sul grafo di Voronoi. Il graph partitioning consiste nel partizionare i nodi di un grafo in sottoinsiemi disgiunti.

Nel lavoro svolto in [101] viene applicato il graph partitioning per ottenere una gerarchia topologica della mappa, che può essere usata con metodi di path planning gerarchici. Gli autori inizialmente considerano un grafo di navigazione delle aree accessibili con le loro connessioni, che partizionano in seguito utilizzando criteri di *graph cut* o *spectral clustering* [64].

Lo spectral clustering è un algoritmo di partizione dei grafi. Esso divide i nodi di un grafo in gruppi in modo che la connettività dei nodi all'interno dello stesso gruppo sia massimizzata e che la connettività di nodi in gruppi

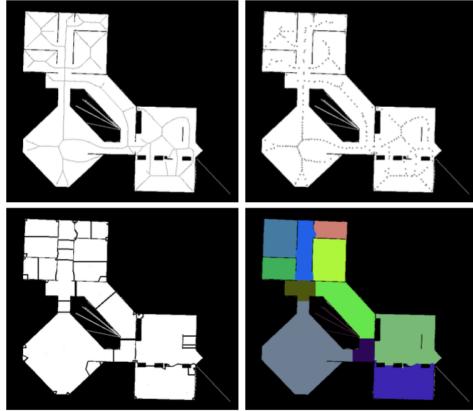


Figura 2.11: Esempio di approccio basato sul grafo di Voronoi come descritto in [10]. La figura in alto a sinistra mostra il grafo di Voronoi. La figura in alto a destra mostra l'insieme dei punti critici estratti. La figura in basso a sinistra mostra le linee critiche. Infine, la figura in basso a destra mostra un esempio del risultato della segmentazione, ottenuta dopo aver unito le regioni createsi dalle linee critiche.

differenti sia minimizzata. In [11] viene proposto un sistema per la generazione di mappe topologiche nel quale si fa uso del graph partitioning con la tecnica dello spectral clustering.

Il risultato finale della segmentazione della mappa in sottomappe, che si ottiene con questo processo, è molto simile a quello ottenuto con il grafo di Voronoi. In Figura 2.12 ne viene riportato un esempio.

In [7] e [36] vengono usati approcci simili, usando *Normalized Cut* per partizionare il grafo.

2.4.3 Il metodo basato su feature

Questo metodo di segmentazione si basa sulle feature, ossia caratteristiche provenienti dall'aspetto locale della mappa metrica, estratte direttamente dai dati sensoriali del robot. Esso fa affidamento sull'individuazione delle feature, etichettando la loro posizione all'interno della mappa e propagando le etichette verso altre posizioni.

I lavori descritti in [59] e [60] affrontano il problema della classificazione dell'ambiente usando sia dati provenienti da sensori di prossimità che immagini. Questo accade per la seguente ragione: molti spazi hanno una geometria particolare e possono essere riconosciuti sulla base dei dati provenienti da uno scanner laser; altri, invece, possono essere riconosciuti solo in base agli oggetti trovati all'interno, come un monitor per un ufficio. L'i-

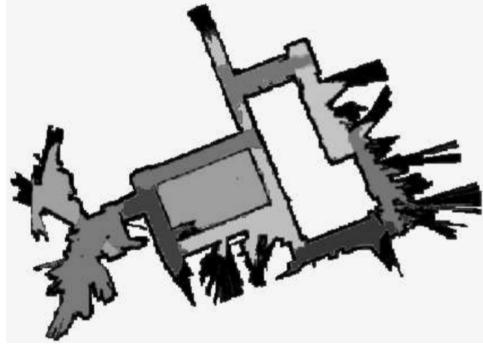


Figura 2.12: Esempio di approccio basato sul graph partitioning come ottenuto nel lavoro svolto in [11]. I diversi gradi di grigio all'interno della mappa ne identificano le differenti regioni.

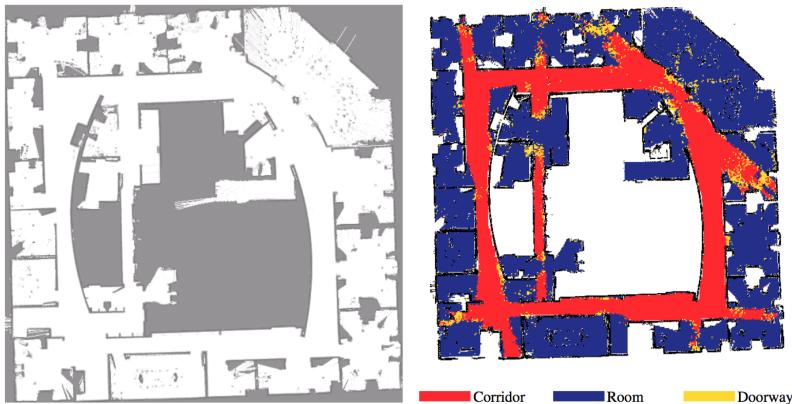


Figura 2.13: Esempio di classificazione con l'algoritmo AdaBoost, tratto da [59]. L'immagine a sinistra mostra una mappa metrica di test. L'immagine a destra mostra il risultato della classificazione con AdaBoost.

dea base dell'approccio risiede infatti nel classificare la posizione del robot basandosi sulle osservazioni correnti sia del laser che della telecamera. La classificazione viene fatta grazie all'uso dell'algoritmo AdaBoost [76]. Questo classificatore è costruito in modo supervisionato partendo da semplici elementi geometrici, estratti grazie all'uso dei sensori del robot. Utilizzando un ampio insieme di dati in ingresso all'algoritmo di AdaBoost, il *training set* che contiene le feature, è possibile associare ad ogni punto della mappa un'etichetta spaziale, come ad esempio: stanza, corridoio o porta. Infine si procede con la segmentazione, propagando le etichette semantiche derivate dall'ambiente. La Figura 2.13 mostra un esempio.

Il lavoro svolto in [30] combina il lavoro sopracitato con la segmentazione

tramite Voronoi, in modo da assegnare un'etichetta ad ogni posizione nella mappa metrica, definita in relazione al tipo di regione a cui appartiene. Queste etichette forniscono una segmentazione intrinseca della mappa, la quale rappresenta la struttura topologica di un ambiente.

Un altro lavoro che ha l'obiettivo di riconoscere le stanze di un edificio, apprendendo l'aspetto di porte o passaggi stretti, è quello in [23] nel quale si ipotizza la separazione in spazi differenti ogni qualvolta il robot attraversa un passaggio stretto.

Altri esempi di approcci basati sulle feature possono essere trovati nei lavori in [65] e [82].

2.4.4 Il metodo basato su operatori morfologici

La *fuzzy mathematical morphology* descritta in [8] e [54] è un insieme di metodi teorici per l'analisi delle forme geometriche. Si occupa infatti dell'estrazione delle forme all'interno di un'immagine digitale. Il processo si basa sugli operatori di dilatazione ed erosione che permettono di variare la forma e la dimensione di un pattern di riferimento, il quale viene poi confrontato con le forme all'interno dell'immagine digitale.

Gli autori dei lavori svolti in [12] e [28] hanno applicato gli operatori morfologici al task di segmentazione dell'ambiente, in modo da poter estrarre da una mappa a griglia iniziale grandi spazi aperti, usando un elemento strutturale canonico per esprimere il concetto di spazio aperto. Da una mappa a griglia *fuzzy* [67], definita come array bidimensionale di valori compresi nell'intervallo $[0,1]$, vedi Figura 2.14 (a), viene applicato un algoritmo di pattern matching, ottenendo il risultato in Figura 2.14 (b). Esso rappresenta una nuova mappa a griglia, nella quale il valore di ogni cella rappresenta quantitativamente quanto essa appartenga ad un grande spazio aperto. Per catturare la struttura topologica e segmentare la mappa, applicano l'algoritmo di *watershed* [96], che partendo dal valore di una cella della mappa a griglia e passando ad altre con valori inferiori, determina quali appartengono alla stessa stanza fermandosi nei minimi locali, come in Figura 2.14 (c). Infine, la Figura 2.14 (d) mostra il corrispondente grafo topologico, nel quale gli archi corrispondono alle porte riconosciute attraverso i passaggi stretti.

Un altro esempio di segmentazione con operatori morfologici è presente nel lavoro svolto in [10]. L'algoritmo proposto lavora su una mappa a griglia

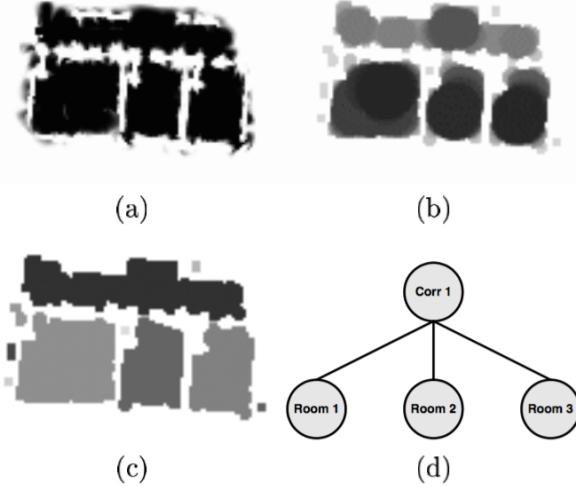


Figura 2.14: Esempio di approccio con operatori morfologici, tratto da [12]: (a) mappa a griglia originale, (b) spazi morfologici fuzzy, (c) segmentazione con l'uso dell'algoritmo watershed, (d) mappa topologica.

iniziale, nella quale ogni pixel libero viene etichettato come accessibile ed ogni pixel occupato come inaccessibile; come mostra l'immagine in alto a sinistra nella Figura 2.15. In essa si evidenziano con il colore nero tutte le celle inaccessibili e con quello bianco le accessibili. I muri della mappa metrica vengono incrementalmente allargati di un pixel con l'operatore morfologico di erosione. Questo permette di controllare quali aree, che precedentemente erano collegate, ora non lo siano più, come mostra l'immagine in alto a destra di Figura 2.15. Tutte le porzioni createsi possono ora essere identificate come regioni separate, come si può vedere nell'immagine in basso a sinistra in Figura 2.15. Infine, grazie ad una propagazione a fronte d'onda, ogni area etichettata viene estesa a tutta l'area non etichettata della stessa regione. Il risultato viene mostrato nell'immagine in basso a destra della Figura 2.15.

Un altro lavoro, che in maniera analoga a [10] fa uso di questo metodo, viene presentato in [9].

2.4.5 Il metodo basato sulla distance transform

Un altro approccio di segmentazione si basa sull'uso della distance transform. La distance transform è una trasformazione che associa ad ogni pixel etichettato come accessibile nell'immagine un valore in base alla distanza rispetto al suo più vicino pixel etichettato come inaccessibile. Un esempio di distance transform applicata ad una mappa a griglia si trova nell'imma-

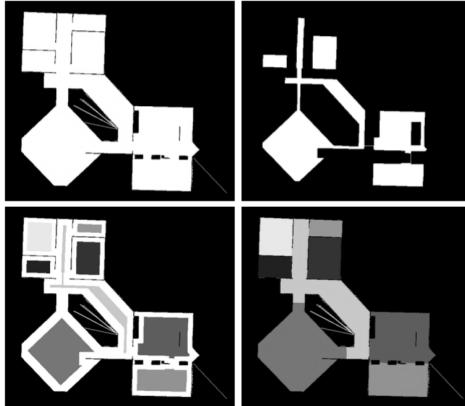


Figura 2.15: Esempio di approccio con operatori morfologici, tratto da [10]: la figura in alto a sinistra rappresenta la mappa a griglia originale, l'immagine in alto a destra rappresenta la mappa erosa dall'operatore di erosione, l'immagine in basso a sinistra rappresenta un etichettamento iniziale della mappa e l'immagine in basso a destra la segmentazione finale

gine in alto a sinistra della Figura 2.16. I massimi locali di una distance transform risiedono solitamente nel centro di una stanza. In passaggi stretti o in corridoi, il massimo locale è sempre minore rispetto a quello di stanze più grandi. Se viene trovata una soglia ottimale per la distance transform, è possibile trovare il centro di ogni stanza, come si può vedere nelle immagini in alto a destra e in basso a sinistra della Figura 2.16. Una soglia viene scelta in modo tale da poter ottenere il numero di centri che massimizzano il numero di stanze correttamente identificate. Una volta trovata la soglia ottimale è possibile espandere l'area con una propagazione a fronte d'onda segmentando così l'ambiente, come è stato fatto in [10].

In [21] viene descritto un approccio semi-automatico per la segmentazione, il quale fa uso della distance transform. Il robot segue l'utente che assegna etichette a svariate posizioni. Alla fine del processo di mapping, viene usata la distance transform per raggruppare le celle della mappa metrica in stanze, per poi assegnarle all'etichetta più vicina. La distance transform genera massimi locali e tutti i pixel che sono attratti dallo stesso massimo locale durante la salita del gradiente vengono considerati come appartenere alla stessa stanza.

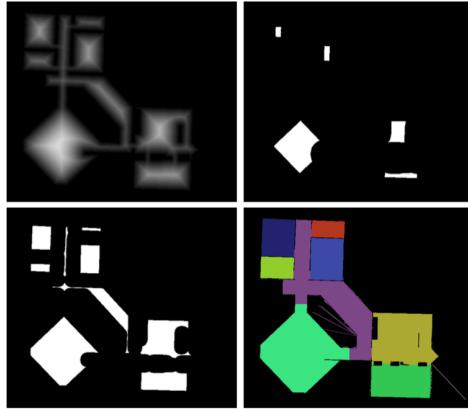


Figura 2.16: Esempio di approccio con distance transform, tratto da [10]: la figura in alto a sinistra rappresenta la distance transform applicata alla mappa a griglia originale, l'immagine in alto a destra rappresenta l'insieme dei centri delle stanze utilizzando una soglia non ottimale, l'immagine in basso a sinistra rappresenta l'insieme dei centri delle stanze utilizzando una soglia ottimale e l'immagine in basso a destra la segmentazione finale.

2.4.6 Il metodo basato sulla ricostruzione del layout

Tutti i metodi precedentemente descritti eseguono la segmentazione dell’ambiente direttamente sulla mappa metrica. Altri metodi, invece, si basano su dei processi preliminari da compiere sulla mappa a griglia, per poi fornire una suddivisione in stanze dell’ambiente stesso. Queste metodologie prendono il nome di metodi di ricostruzione del layout.

Nonostante nel lavoro svolto in [15] gli autori non facciano riferimento alla ricostruzione del layout, viene costruita una rappresentazione a griglia bidimensionale dell’ambiente, chiamata A-Grid, che infatti può essere utile per la ricostruzione del layout, come è evidente nei lavori [13] e [61]. La mappa a griglia A-Grid rappresenta una discretizzazione della mappa metrica, costruita estraendo linee in corrispondenza dei muri dell’edificio in questione. Questa rappresentazione dell’ambiente è caratterizzata da due proprietà:

- Non è uniforme nel relazionarsi con la mappa metrica originale, poiché la dimensione della cella varia in funzione delle parti identificate.
- È accessibile come una semplice matrice.

Usando questa struttura è possibile suddividere l’edificio in aree funzionali e passaggi, fornendo così le informazioni necessarie per la creazione di una mappa topologica, nella quale le etichette possono eventualmente essere as-

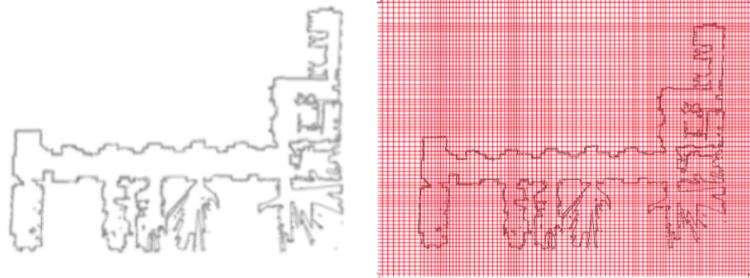


Figura 2.17: Esempio di mappa a griglia di tipo A-Grid ottenuta con l'algoritmo proposto in [15].

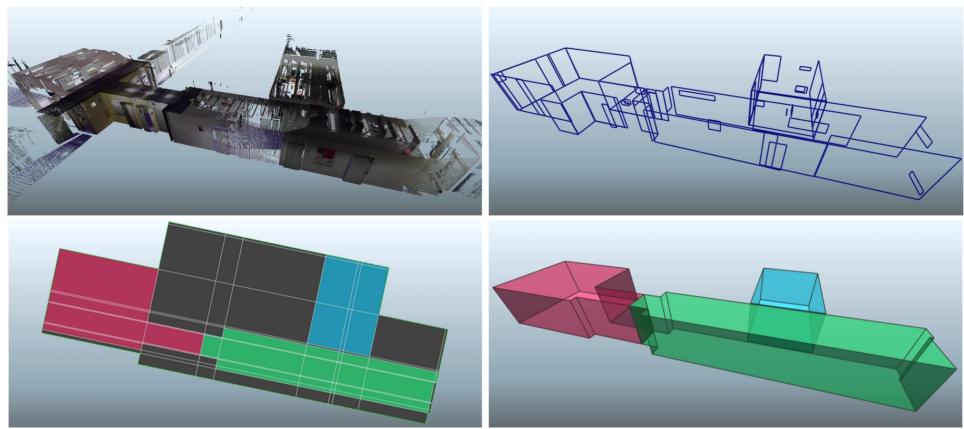


Figura 2.18: Esempio del processo di ricostruzione del layout tratto da [61]. L'immagine in alto a sinistra rappresenta il modello in input. L'immagine in alto a destra rappresenta l'estrazione dei muri candidati, che vengono proiettati sul piano 2D, sul quale viene fatta la segmentazione. Piano visibile nell'immagine in basso a sinistra. Infine nell'immagine in basso a destra vengono rappresentate le stanze in 3D.

sociate ai nodi. In Figura 2.17 viene fornito un esempio.

Sebbene i lavori in [62] e [61] non siano stati svolti nell'ambito della robotica, essi sono di particolare rilevanza in questo lavoro di tesi poiché pongono un approccio simile alla ricostruzione del layout in ambienti indoor. L'input dell'algoritmo di ricostruzione consiste in un insieme di scansioni tridimensionali, prese in posizioni conosciute e che rappresentano l'interno di un edificio. Di seguito vengono illustrati i vari passi dell'algoritmo, visibili in Figura 2.18.

Selezione dei muri candidati. Le regioni planari verticali che indicano un potenziale muro, vengono estratte dalle scansioni 3D di una telecamera.

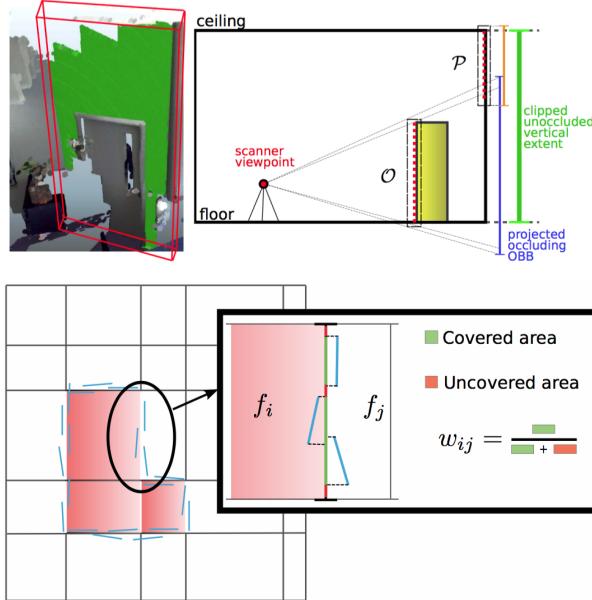


Figura 2.19: Esempio di come in [61] viene associato il peso ad un muro candidato.

Segmentazione. Questo passaggio viene interamente eseguito sul piano xy , sul quale sono stati proiettati i muri candidati. In primis, le proiezioni dei muri, chiamate segmenti, sul piano xy vengono raggruppate per ottenere un minor numero di linee rappresentative per i muri. Queste linee rappresentative si ottengono dividendo inizialmente l’insieme dei segmenti in cluster che ne caratterizzano l’orientamento, per poi raggruppare ulteriormente i segmenti vicini. In seguito viene costruita una griglia le cui celle sono il risultato dell’intersezione tra le linee rappresentative. Ad ogni bordo delle celle createsi viene associato un peso, corrispondente alla probabilità che esso sia o meno un vero muro. Lo svolgimento di questa operazione viene mostrato nella Figura 2.19. Con un processo di diffusione si ottengono le similitudini tra tutte le celle adiacenti, che vengono classificate come appartenenti alla stessa stanza grazie all’uso iterativo dell’algoritmo *k-medoids* [68], come mostra l’immagine in basso a sinistra della Figura 2.18.

Ricostruzione del modello. Viene in seguito calcolata la geometria dei muri. Questo permette di creare per ogni stanza un poligono, intersecando i muri ricostruiti con i piani che rappresentano il soffitto e il pavimento, come mostra l’immagine in basso a destra della Figura 2.18.

In [13] è stato usato un approccio simile alla costruzione della mappa A-Grid che, differentemente dal lavoro svolto in [15], viene usato come obiettivo

primario per la ricostruzione del layout, utilizzando tecniche simili ai lavori [62] e [61], descritti in precedenza. Il lavoro svolto in [13] è di particolare importanza per questo lavoro di tesi, poiché rappresenta la base di partenza sulla quale viene fatta la nostra predizione dell'ambiente. Per tale motivo, di seguito, verrà fornita una breve introduzione al metodo, scendendo più nel dettaglio nel Capitolo 4.

Il metodo proposto in [13] ricerca i muri all'interno di una mappa metrica iniziale a griglia. I muri vengono usati per segmentare la mappa in stanze e ricostruire il layout dell'ambiente. Il processo si basa su un numero di passaggi eseguiti sequenzialmente. La Figura 2.20 mostra l'intero processo. La prima operazione compiuta sulla mappa metrica consiste nell'identificare i bordi significativi usando l'algoritmo *Canny edge detection* [14], che partitiona le celle della mappa metrica in spazio libero ed occupato (Figura 2.20 (b)). Grazie all'algoritmo *Hough line transform* [42] viene calcolato l'insieme dei segmenti che approssima i bordi trovati con l'algoritmo di Canny (Figura 2.20 (c)). In seguito, viene calcolato il contorno della mappa usando un algoritmo per il riconoscimento dei contorni in un'immagine [86] (Figura 2.20 (d)). I segmenti riconosciuti con l'algoritmo di Hough vengono raggruppati in cluster. Prima per orientamento, identificando i cluster angolari. Poi, per ogni cluster angolare viene attuato un ulteriore raggruppamento in base alla prossimità spaziale, identificando così i muri dell'edificio (Figura 2.20 (e)). Ad ogni cluster spaziale viene in seguito associata una linea rappresentativa (Figura 2.20 (f)). Le linee rappresentative sono delle rette identificate da un punto e dall'orientamento. L'orientamento viene dato dal cluster angolare (al quale il cluster spaziale corrispondente alla retta appartiene), mentre il punto coincide con il mediano dei punti medi dei segmenti del cluster spaziale. Le linee che si formano (linee rosse in Figura 2.20 (f)) vengono successivamente utilizzate per suddividere l'area della mappa, formando delle facce. Queste facce sono il risultato dell'intersezione tra le linee rappresentative. Raggruppando le facce ottenute è così possibile ottenere le stanze. Le facce adiacenti divise da un muro devono appartenere a stanze diverse, mentre facce adiacenti divise da un bordo che non corrisponde a nessun muro dovrebbero essere raggruppate all'interno della stessa stanza (Figura 2.20 (g)). Così si ottiene un layout ricostruito (Figura 2.20 (h)).

Un metodo usato per la ricostruzione del layout basato su dati estratti con una stereo camera viene illustrato in [31]. La ricostruzione del layout viene fatta riconoscendo i piani verticali all'interno delle immagini, i quali vengono classificati come possibili muri di una stanza. A causa dell'alto numero di muri riconosciuti vengono generate diverse combinazioni in ma-

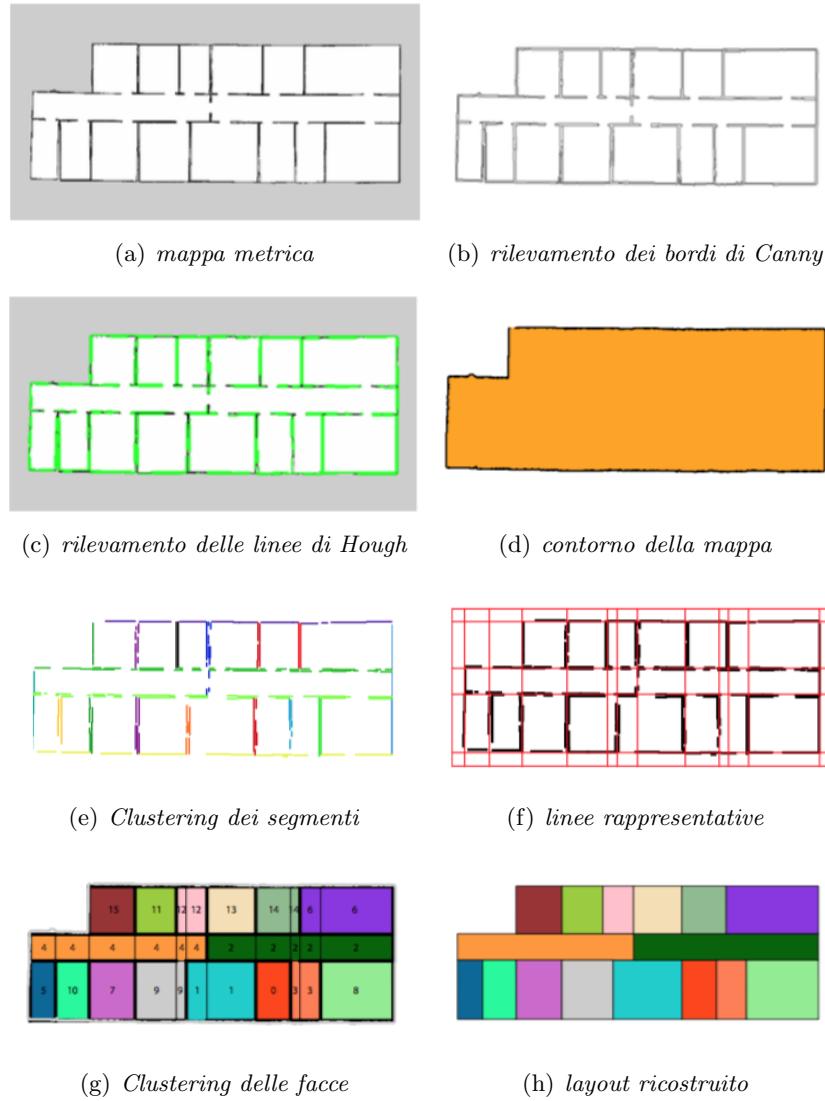


Figura 2.20: Processo di ricostruzione del layout tratto da [13].

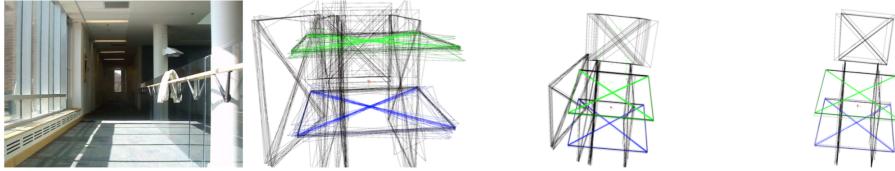


Figura 2.21: Esempio del processo con il quale il lavoro in [31] ottiene il layout a partire dai dati provenienti da una stereo camera.

niera casuale dette ipotesi. Ogni ipotesi di layout viene in seguito valutata attraverso la propria compatibilità con le osservazioni ottenute ed i vincoli geometrici fra fotogrammi. Durante questo processo, ogni layout viene perturbato ottimizzando, unendo o separando i componenti che lo costituiscono. Questo processo viene attuato grazie ad un framework probabilistico con una struttura *particle filter*. L'intero processo con il quale si ottiene il layout viene mostrato in Figura 2.21.

Un altro metodo, presentato in [35], consiste nel ricercare linee rette all'interno di un'immagine monoculare e raggrupparle in tre punti di fuga, che servono a specificare l'orientamento di una box con cui si approssima una stanza, fornendo vincoli sul layout finale. Campionando le linee, vengono generate diverse box candidate. Queste box candidate vengono in seguito valutate sulla base dei dati provenienti dall'immagine stessa e da modelli precedentemente appresi, per poi selezionare la migliore. Viene successivamente assegnata un'etichetta a tutte le superfici che compongono la box estratta. Le etichette permettono di distinguere tra muri, pavimento e soffitto.

In [49] viene fornito un metodo per la ricostruzione del layout, con l'obiettivo di fornire una mappa dell'ambiente astratta ed annotata semanticamente, ma ancora probabilistica. Per far ciò gli autori hanno utilizzato una tecnica di sampling dal nome *Markov Chain Monte Carlo* (MCMC) per generare campioni da una funzione di densità di probabilità, catturando la distribuzione di mondi probabili che il robot potrebbe incontrare. In seguito, usando un approccio di stima del massimo a posteriori, viene selezionata la configurazione che più rispecchia i dati percepiti.

Il modello astratto creato in questo lavoro ha la forma di uno *scene graph*³. Nel caso in questione lo scene graph consiste in stanze e passaggi.

³Uno scene graph è una struttura in cui si mostra la rappresentazione spaziale e logica di una scena grafica. Uno scene graph è una collezione di nodi in un grafo o in una

Tale rappresentazione viene denotata attraverso un vettore di parametri nascosti W , che specifica lo stato del mondo che ha generato l'occupancy grid map M in questione. Nel framework bayesiano è possibile usare un approccio a posteriori per inferire lo stato più probabile $W^* \in \Omega$ dallo spazio dei possibili mondi probabili Ω , data una determinata mappa M .

$$W^* = \arg \max_{w \in \Omega} p(W|M) \quad (2.1)$$

con

$$p(W|M) \propto p(M|W)p(W) \quad (2.2)$$

dove $p(W|M)$ rappresenta la distribuzione a posteriori di W data la mappa M conosciuta. $p(M|W)$ indica il likelihood con il quale si descrive quanto probabile sia la mappa osservata M , dati differenti mondi probabili identificati dal vettore di parametri W . Con $p(W)$ si intende la probabilità a priori, la quale indica il modello W attuale. Per risolvere l'equazione (2.1) è necessario ricercare all'interno di un grande e complesso spazio delle soluzioni Ω . A tale scopo gli autori, sulla base del lavoro svolto in [34], hanno proposto un approccio *data driven Markov chain Monte Carlo* (DDMCMC). Questo approccio consiste nel creare una catena di Markov che generi campioni W_i dallo spazio delle soluzioni Ω , seguendo la distribuzione $p(W|M)$. Un approccio popolare nella costruzione della catena di Markov è l'algoritmo Metropolis-Hastings (MH), descritto in [16]. Nelle tecniche di MCMC le catene di Markov vengono costruite sequenzialmente eseguendo transizioni da un mondo W ad uno successivo W' grazie ai kernel di transizione. I kernel di transizione sono in grado di modificare il mondo W e sono organizzati in coppie reversibili tali che sia sempre possibile tornare indietro ed ottenere configurazioni precedenti. Un esempio di kernel di transizione viene mostrato nella Figura 2.22.

In [49] vengono considerati per la costruzione del modello finale solo le parti della occupancy grid map iniziale già esplorate. Questo significa che tutte le stanze osservate parzialmente non vengono inserite all'interno della soluzione, come mostra Figura 2.23.

Un'estensione del lavoro svolto in [49] è descritta in [50], che svolge parte fondamentale in questo lavoro di tesi. Infatti, esso rappresenta un esempio di come sia possibile fare inferenza sulla mappa metrica, per fornire al robot un'abilità maggiore nel gestire l'incertezza dei dati acquisiti sull'ambiente. Questo viene fatto eseguendo una predizione di porzioni di ambiente che il robot non ha perfettamente esplorato in precedenza. Per tal motivo il lavoro struttura ad albero.

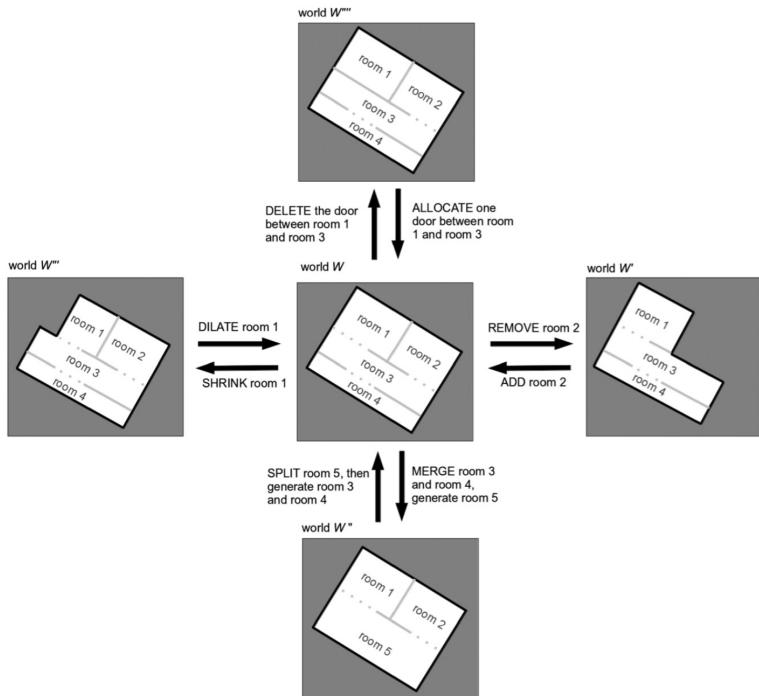


Figura 2.22: *Transitional kernel reversibili, come tratti da [49]. Le coppie reversibili vengono identificate come ADD/REMOVE, SPLIT/MERGE, SHRINK/DILATE, ALLOCATE/DELETE.*

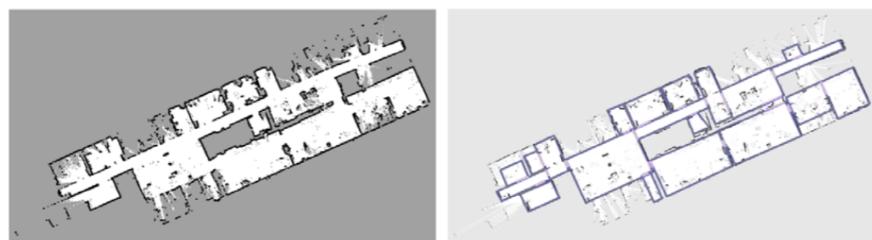


Figura 2.23: *Esempio tratto da [49] nel quale il layout ricostruito esclude le stanze parziali.*

in [50] merita di essere menzionato a parte all'interno della sezione seguente dedicata alla predizione.

2.5 La predizione

Predire ciò che si trova in una parte inesplorata dell'ambiente può migliorare le prestazioni dei sistemi robotici. Molti dei task robotici, come Urban Search and Rescue (USaR) in ambienti sconosciuti o l'esplorazione guidata, attualmente non considerano lo spazio sconosciuto. Infatti, esso non viene preso in considerazione fino a quando non lo si visita. Perché il robot aumenti le proprie capacità autonome, è però necessario che sia in grado di ragionare anche sugli spazi sconosciuti. La qualità dell'esecuzione dei compiti in ambienti inizialmente sconosciuti dipende dall'accuratezza con cui il robot predice le parti inesplorate dell'ambiente.

Con lo scopo di illustrare l'importanza del task di predizione si immagini un robot progettato per operare in situazioni di pericolo e salvataggio (USaR), che entra all'interno di un edificio sconosciuto con obiettivo di localizzare sopravvissuti. In questo caso il goal del robot è quello di individuare il maggior numero di persone nel minor tempo possibile [19]. Un robot senza una mappa completa dovrebbe voler in primo luogo capire che cosa c'è all'interno della parte di spazio inesplorata, ricercando quali siano le aree in cui, con maggior probabilità, potrebbero esserci sopravvissuti.

Un altro esempio è un robot mobile di servizio, che ha il compito di pulire la cucina di un edificio di cui non conosce la mappa. Nel suo primo giorno il robot deve pianificare una serie di azioni, che gli permettano di compiere il task assegnato. Quindi deve essere, in primis, in grado di trovare la cucina attraverso l'esplorazione. Un piano d'esplorazione significativo considererebbe un corridoio come luogo che collega diverse stanze, come avviene in molti ambienti indoor. Le aspettative del robot di trovare una cucina esplorando parti di un corridoio inesplorato dovrebbero essere maggiori rispetto al ricercare la cucina come stanza non affacciata sul corridoio, ma connessa ad una qualsiasi delle altre stanze trovate durante l'esplorazione.

Tale alto grado di autonomia richiede una comprensione generale di come gli ambienti interni siano tipicamente strutturati. Nel lavoro in [4] viene utilizzato un approccio *data-driven* per la costruzione di modelli di ambienti interni per prevedere quello che ci si aspetta di incontrare in futuro, in una determinata tipologia d'ambiente. Gli autori hanno deciso di modellare il mondo come un grafo. Il sistema, dato un grafo parziale, è in grado di predire la categoria della prossima stanza ed anche a cosa essa sia connessa.

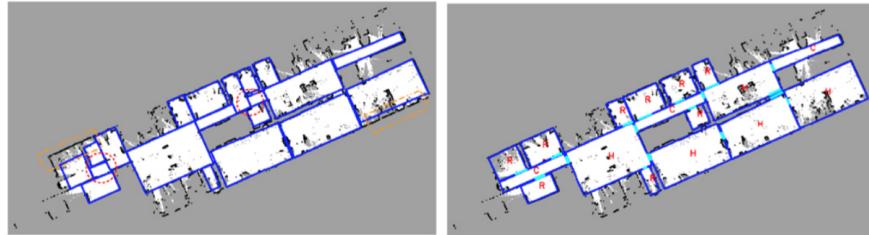


Figura 2.24: Esempio tratto da [50]. La prima immagine si riferisce al risultato ottenuto applicando il precedente metodo degli autori, descritto in [49]. Gli errori strutturali vengono evidenziati con le linee tratteggiate rosse, mentre gli errori dovuti alle corrispondenze locali vengono evidenziati con le linee tratteggiate arancioni. La seconda immagine rappresenta un esempio di risultato ottenuto con il metodo presentato in [50].

Un'altra abilità importante per un robot mobile è quella di saper gestire l'incertezza e la complessità dei dati provenienti dal mondo reale. Il lavoro svolto in [50] fornisce una metodologia che permette ad un sistema autonomo intelligente di fare inferenza in accordo con una specifica conoscenza di base. A tale scopo, gli autori, hanno deciso di sfruttare un approccio in grado di combinare il campionamento Data Driven Markov Chain Monte Carlo (DDMCMC), definito nel loro lavoro precedente [49], con una rete logica di Markov (MLN). Questo perché il metodo DDMCMC rappresenta uno strumento efficiente per ottenere campioni provenienti da una distribuzione complessa e sconosciuta, mentre la rete logica di Markov è uno strumento per modellare la conoscenza incerta.

Nonostante i risultati ottenuti in [49] fornissero una buona rappresentazione dei dati in ingresso, tali risultati mostravano errori su corrispondenze locali all'interno della mappa. Questo non permetteva di gestire correttamente stanze esplorate parzialmente, poiché escludeva dal modello finale porzioni di stanze. Ciò viene sottolineato dalle linee tratteggiate arancioni nell'immagine a sinistra della Figura 2.24. Al contrario, la nuova metodologia permette di gestire questi errori attraverso un più raffinato *bottom-up feature detector*⁴, fornendo risultati più accurati, come mostra l'immagine a destra della Figura 2.24.

Un ulteriore lavoro nel quale si tenta di fornire una predizione dell'ambiente, inserendo valori mancanti in osservazioni parziali, è quello svolto in

⁴Gli autori hanno seguito l'approccio espresso in [91], usando un bottom-up feature detector di tipo *discriminante* per riconoscere caratteristiche rilevanti all'interno dell'ambiente, come ad esempio muri e porte.

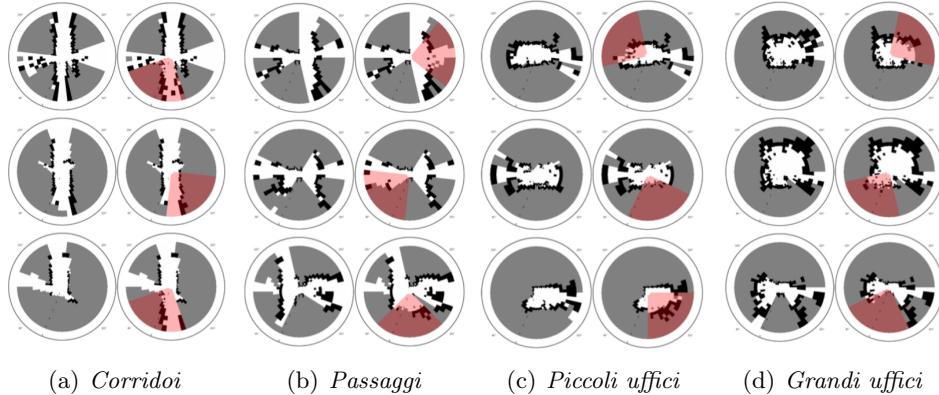


Figura 2.25: Esempio tratto da [73] di completamento sia con sia senza successo di osservazioni con dati mancanti, raggruppate per categoria semantica. Per ogni coppia viene mostrata sulla sinistra la mappa completa, mentre la mappa ottenuta con il processo di predizione viene mostrata sulla destra. Viene inoltre evidenziata in rosso l'angolazione di 90 gradi su cui è stata fatta la predizione

[73]. Gli autori hanno proposto un metodo probabilistico per la generazione di porzioni locali di un ambiente. Questo metodo apprende la distribuzione congiunta tra una rappresentazione di basso livello dell'ambiente (una occupancy grid map) e la propria interpretazione semantica. A tal proposito gli autori hanno deciso di affrontare il problema usando una Sum-Product Networks (SPN), un'architettura probabilistica descritta in [70], [32] e [69]. La Figura 2.25 mostra un esempio di come viene affrontata la predizione in [73], divisa nelle quattro categorie: corridoi, passaggi, uffici piccoli ed uffici grandi. Nonostante la predizione del modello approssimi considerevolmente la mappa a griglia originale, il sistema proposto potrebbe sbagliare. Questo tipicamente succede se le informazioni critiche, utili a determinare la classe di appartenenza della porzione in considerazione, non sono presenti.

Un lavoro svolto in letteratura che offre una metodologia per la generazione automatica di layout, viene presentato in [57]. Nonostante esso non fornisca una vera e propria metodologia per la predizione di porzioni sconosciute all'interno di una mappa ottenuta attraverso l'esplorazione robotica, fornisce una metodologia utile per capire quanto un dato layout rispecchi la realtà, basandosi sulla geometria dell'ambiente e sulle scelte attuate dagli architetti durante la fase di progettazione di un determinato edificio. Questo perché molti edifici come scuole, ospedali ed uffici possiedono strutture simili e prestabilite.

Anche gli autori di [57], per ottenere il layout migliore all'interno del-

lo spazio di tutti i layout possibili, hanno usato una metodologia di campionamento MCMC, applicando l'algoritmo Metropolis Hastings. Questo algoritmo, al contrario di altre tecniche greedy, è in grado di accettare o rifiutare mosse, capaci di aumentare il valore di una funzione di costo, specificatamente definita come Boltzmann objective function

$$f(x) = \exp(-\beta C(x)) \quad (2.3)$$

dove x rappresenta il layout da valutare, $C(x)$ la funzione di costo e β una costante. Ad ogni iterazione l'algoritmo genera un nuovo layout x^* che verrà in seguito accettato con probabilità

$$\begin{aligned} \alpha(x^*|x) &= \min\left(1, \frac{f(x^*)}{f(x)}\right) = \\ &= \min\left(1, \exp(\beta C(x) - \beta C(x^*))\right) \end{aligned} \quad (2.4)$$

L'ottimizzazione parte da una configurazione di alto costo, in cui i rettangoli, che rappresentano le stanze, hanno ugual dimensione. Vengono poi attuate una serie di mosse in grado di esplorare lo spazio dei layout. Nel lavoro in questione è stato dimostrato, inoltre, che si può raggiungere ogni possibile layout partendo da un altro, usando due semplici mosse che differiscono dal lavoro in [49]. Queste mosse sono:

- **Spostare un muro.** Permette di muovere in diverse direzioni un intero muro o di spezzarlo muovendone solo una parte.
- **Scambio stanze.** Permette di scambiare il significato semantico di due stanze.

La procedura è attenta a minimizzare la funzione di costo $C(x)$ che valuta la qualità di un layout. La funzione di costo penalizza violazioni architettonali, come stanze malformate ed incompatibilità tra piani. La Figura 2.26 mostra un esempio di alcune violazioni architettonali.

Il lavoro sopracitato fornisce dunque un esempio di come sia possibile valutare un determinato layout. Grazie appunto al valore della funzione è possibile selezionare il miglior layout tra tutti quelli possibili. Ed è proprio per questo motivo che il lavoro [57] risulta essere di particolare importanza in questo lavoro di tesi. Infatti, in quest'ultimo abbiamo seguito un approccio simile, nel quale si fa uso di una funzione di costo progettata ad hoc per la valutazione di tutti i layout generati permettendo di selezionare il migliore.



Figura 2.26: Esempio di layout generati tratto da [57]. Nella prima immagine le porte vengono inserite in posizioni errate, poiché non permettono di raggiungere alcune stanze. Nella seconda immagine viene violato il vincolo sulle aree, in quanto esistono stanze troppo piccole. Nella terza e quarta immagine vengono violati rispettivamente il vincolo sull'aspetto e quello sulla forma. Nell'ultima immagine è presente un esempio di layout corretto, nel quale non viene violato alcun vincolo.

Capitolo 3

Impostazione del problema di ricerca

In questo capitolo viene descritto il concetto di predizione dal punto di vista del lavoro oggetto di questa tesi. Si fornisce la motivazione che ha spinto la ricerca in questo campo, dando una spiegazione sul perché tale questione sia stata affrontata. Nella parte iniziale di questo capitolo vengono illustrati lo scopo del lavoro e le fasi del modello che hanno reso possibile ottenere una predizione dell’ambiente. Vengono inoltre introdotti e spiegati i vincoli sotto cui il sistema opera ed infine si confronta il metodo sviluppato con altri lavori svolti.

3.1 Lo scopo del lavoro

La volontà dell’essere umano di fornire ai robot capacità sempre più simili alle proprie è di particolare rilevanza nel campo della robotica mobile autonoma. In questo lavoro di tesi si vuole fornire una metodologia di predizione che aiuti un sistema di intelligenza artificiale a prevedere ciò che può esserci in porzioni di ambiente che non ha ancora esplorato, basandosi interamente sulla propria esperienza passata.

Si immagini un possibile scenario in cui opera un robot che sta esplorando un ambiente sconosciuto impiegando una metodologia d’esplorazione basata sulle frontiere. Ad ogni frontiera viene associato un valore, comunemente detto *information gain* [33]. L’information gain stima quanto sia promettente esplorare una frontiera. Maggiore è il valore dell’information gain, maggiore è la probabilità che il robot scelga esattamente quella frontiera come prossima ad essere esplorata, seguendo un piano di navigazione

che gli permetta di muoversi intelligentemente dalla sua attuale posizione fino a quella della frontiera selezionata, al fine di esplorare l’ambiente.

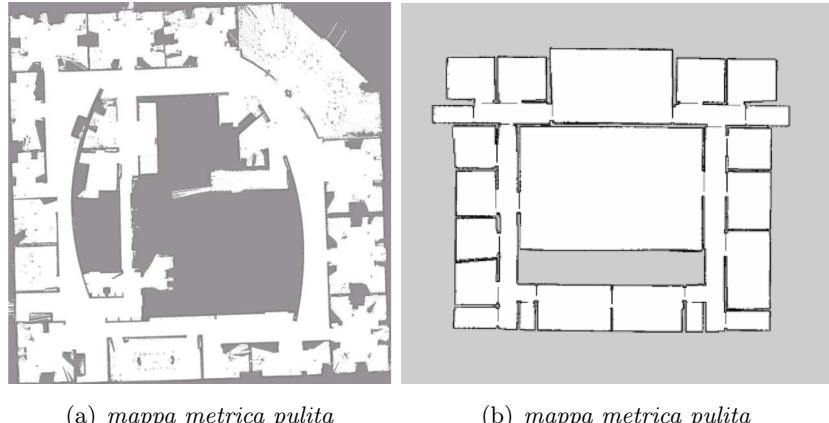
La scelta della migliore frontiera prossima ad essere esplorata è di particolare importanza quando si vuole esplorare l’ambiente nel minor tempo possibile. Nel caso in cui l’obiettivo del robot fosse appunto esplorare l’ambiente nel minor tempo possibile, cosa succederebbe se si dirigesse verso una frontiera che possiede maggior information gain ma che una volta raggiunta non apporta un considerevole grado di conoscenza, poiché la stima era imprecisa e il robot percepisce ostacoli a pochi metri di distanza? Da ciò ne conseguirebbe una perdita di tempo considerevole nell’esplorazione. Una soluzione a questo problema potrebbe essere quella di predire in modo più accurato cosa il robot potrebbe vedere da una determinata posizione, una frontiera, in modo da migliorare l’information gain.

Il lavoro oggetto di questa tesi introduce il concetto di predizione ambientale con la quale il robot, applicando una specifica inferenza sui dati acquisiti fino ad un determinato momento, è in grado di affermare con buona confidenza cosa potrebbe essere possibile osservare in futuro da una determinata posizione. Questo permette di capire quali sono le frontiere più promettenti.

Lo scopo del lavoro è quello di fornire un sistema che sia in grado di predire la struttura metrica di un ambiente, sotto l’assunzione di trovarsi in territorio indoor fortemente strutturato; basandosi su dati specificatamente processati, provenienti da una parte dell’ambiente stesso già esplorata in precedenza dal robot mobile. Il sistema fornisce un insieme di ipotesi di come una porzione della mappa metrica possa essere completata lavorando su una rappresentazione di più alto livello, come il layout. Il sistema è inoltre in grado di selezionare, all’interno dello spazio delle ipotesi ottenute, la migliore; quella che meglio approssima il layout predetto e che non violi vincoli di carattere geometrico, come ad esempio aggiungere una porzione di ambiente ad una stanza, eliminando un muro già osservato. La selezione della migliore ipotesi è possibile grazie alle simmetrie e alle regolarità che la porzione predetta condivide con la restante parte di mappa già esplorata.

3.2 Le assunzioni

Il vincolo principale sotto cui il sistema opera è quello di trovarsi all’interno di un ambiente indoor fortemente strutturato. In questo lavoro di tesi, il problema dell’occlusione visiva introdotta dagli oggetti nel mondo viene ignorato. Si assume infatti che la scena sia quasi completamente visibile o che, se una parte della mappa sia occlusa in un punto di vista, esista sicuramente un altro punto di vista da cui sia possibile recuperare i dati.



(a) *mappa metrica pulita*

(b) *mappa metrica pulita*

Figura 3.1: Esempi di occupancy grid map. Nell'immagine a) si notano imperfezioni dovute alla presenza di oggetti all'interno dell'ambiente. L'immagine b) è un esempio di ambiente pulito, nel quale non vi sono oggetti all'interno.

Questo porta all'assunzione che l'ambiente da cui viene estratta la mappa metrica fornita in input al sistema sia quasi completamente privo di oggetti. La Figura 3.1 (b) mostra un esempio di mappa metrica pulita.

Questa assunzione non è verificata in situazioni pratiche in cui si ha a che fare con occlusioni visive complesse o ambienti, nei quali sono presenti un numero considerevole di oggetti. La Figura 3.1 (a) mostra un esempio di mappa metrica con occlusioni dovute a oggetti all'interno dell'ambiente.

Come in altri lavori, anche in questo elaborato si vede la necessità di sfruttare una conoscenza a priori sulla struttura dell'edificio per ottenere una certa robustezza, sfruttando l'assunzione di *Manhattan World* [94]. Questa è un'assunzione che forza una figura planare in cui le pareti sono tra di loro ortogonali. La Figura 3.1 (a) mostra un esempio di mappa che non sfrutta l'assunzione Manhattan World, in quanto esistono sia pareti semicircolari che coppie di pareti adiacenti non poste ortogonalmente.

In questo lavoro di tesi si vede inoltre la necessità di assumere il lavoro in [13] come metodo principale di ricostruzione del layout. Questo metodo infatti permette di estrarre dalla mappa metrica a griglia iniziale un insieme di informazioni utili al compimento del task di predizione.

3.3 Il rapporto del lavoro con lo stato dell'arte

Nel capitolo precedente sono stati descritti i lavori presenti nello stato dell'arte inerenti al progetto svolto in questo lavoro di tesi. In questa sezione vengono illustrate le similitudini e le differenze con i vari approcci da cui si è preso spunto. La predizione di ambienti sconosciuti si situa all'interno di un ambito non ancora del tutto esplorato. Questo porta ad avere un numero limitato di lavori da cui è possibile attingere. Di seguito verranno introdotti i limiti di tali approcci e come questi siano stati affrontati in questo lavoro di tesi.

Come spiegato nella sezione precedente, il lavoro è fortemente condizionato dal metodo di ricostruzione del layout presentato in [13]. Questa metodologia, nonostante sia in grado di gestire efficientemente mappe a griglia parziali in input al sistema, non è, tuttavia, in grado di riconoscere e gestire in maniera adeguata gli ambienti parziali. Dunque, al fine di poter fornire un'ipotesi di completamento di una stanza parziale, è stato necessario affrontare il problema del riconoscimento delle stanze parziali all'interno della mappa metrica, applicando un meccanismo che fosse in grado di riconoscerle correttamente. Questo è stato possibile, prendendo spunto dal lavoro in [99], il quale fornisce una metodologia per il riconoscimento delle frontiere all'interno di una mappa metrica a griglia.

Il completamento delle stanze parziali prende spunto dal lavoro in [50], nel quale si fa uso di una metodologia basata sul campionamento Data Driven Markov Chain Monte Carlo. In [50] vengono generate delle ipotesi attraverso operazioni, dette kernel di transizione, che permettono di modificare il mondo aggiungendo, togliendo o modificando le celle che lo compongono. In [50] viene generato un insieme di layout, detto insieme delle ipotesi, dal quale viene estratto il layout migliore, selezionato grazie all'uso di una determinata funzione di costo [57]. La predizione nel lavoro oggetto di tesi viene affrontata in maniera simile. L'insieme delle ipotesi viene creato aggiungendo o togliendo le celle che compongono il layout. Le ipotesi vengono in seguito valutate per far sì che venga selezionata la migliore, ossia quella che meglio rispecchia il layout stesso.

Nonostante sussista un rapporto di similarità tra i due metodi, esiste una differenza nell'applicativo in cui questi vengono utilizzati. Infatti, il metodo sviluppato in [50] non è utile alla predizione di ambienti parziali, in quanto, come mostra la Figura 3.2, non permette di estrarre nella soluzione finale stanze insufficientemente visitate, ma permette di gestire efficientemente l'incertezza e la complessità dei dati provenienti dal mondo reale. Al con-

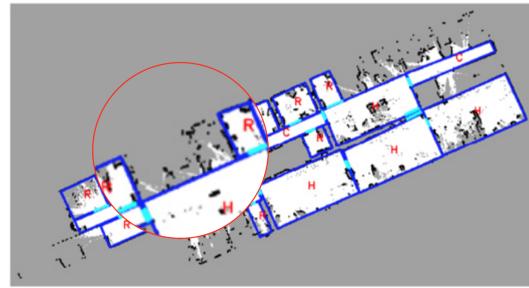


Figura 3.2: Immagine tratta da [50]. La sezione evidenziata dell'immagine mostra come, nonostante esistano dati osservati dell'ambiente, questi non vengano inclusi all'interno del layout ricostruito.

trario, nel metodo introdotto in questo lavoro di tesi, l'obiettivo principale è quello di fornire un'ipotesi di completamento di stanze parziali. L'esistenza, dunque, di porzioni osservate anche in maniera ridotta permette di dedurre che la cella, nella quale viene riscontrata tale osservazione, faccia parte di una stanza del layout ricostruito.

Capitolo 4

La predizione del layout

In questo capitolo viene descritto il progetto logico che ha portato allo sviluppo del sistema di predizione di ambienti indoor e che costituisce l'oggetto di questo lavoro di tesi. Vengono elencati gli elementi che compongono l'intero sistema, spiegando per ognuno di essi il funzionamento. In Figura 4.1 viene riportato, come riferimento, il diagramma dei moduli che compongono il sistema, i quali sono:

Mappa metrica. Il robot costruisce la mappa metrica (occupancy grid map), a partire dai dati provenienti dai sensori. Tale mappa rappresenta l'input del modulo di predizione. Le mappe considerate fanno riferimento ad ambienti parziali non ancora del tutto esplorati.

Creazione del layout. Il modulo di creazione del layout permette di fornire in input al sistema di predizione una rappresentazione di più alto livello di un ambiente rispetto alla mappa metrica. Questa rappresentazione prende il nome di layout e fornisce una collezione di informazioni, tra cui i muri dell'edificio e le stanze nelle quali viene suddiviso l'ambiente. Nel

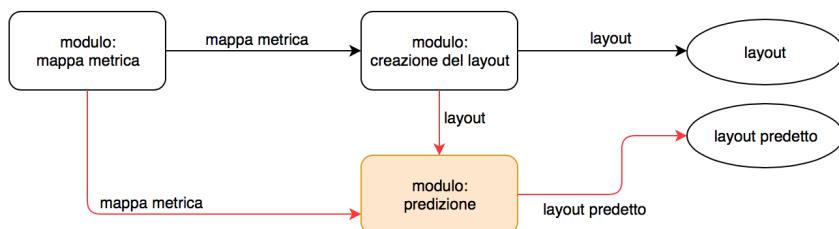


Figura 4.1: Progetto logico del sistema. Nel diagramma sono illustrati i moduli dell'intero sistema.

modulo che crea il layout vengono infatti riconosciuti tutti gli elementi utili al modulo di predizione per effettuare inferenza sul layout stesso. Questi elementi verranno affrontati in modo più approfondito nel corso di questo capitolo.

Predizione. Il modulo di predizione, principale oggetto del lavoro di tesi, avendo a disposizione in input una mappa metrica a griglia parziale ed il layout, permette di fornire un’ipotesi di completamento su tutte le stanze dell’ambiente non interamente esplorate.

La restante parte del capitolo è suddivisa in tre parti, una per ogni modulo che compone il sistema.

4.1 La mappa metrica

La mappa metrica è una rappresentazione a basso livello dell’ambiente. Ciò significa che fornisce conoscenza solo sulla struttura e sull’occupazione dell’ambiente. Ad ogni cella della griglia che costituisce la mappa metrica viene associata l’informazione riguardante la sua occupazione da parte di un ostacolo. Durante il processo d’esplorazione di un ambiente, che porta ad avere una mappa metrica completa, vengono costruite una serie di mappe parziali. Esse rappresentano lo stesso ambiente visto dal robot ad intervalli di tempo differenti. La Figura 4.2 mostra un esempio di mappe metriche parziali raccolte ad intervalli differenti durante l’esplorazione dello stesso ambiente. Le mappe metriche parziali vengono usate come input sia dal modulo che crea il layout sia da quello che ne effettua una predizione. Le mappe metriche parziali prese in considerazione in questo lavoro di tesi sono note a priori, ossia sono il risultato di processi d’esplorazione eseguiti in precedenza.

4.2 La creazione del layout

Il modulo layout come definito in [13] rappresenta il secondo livello di una rappresentazione multilivello, dove il primo livello viene fornito dalla mappa metrica. Come precedentemente affermato nel Capitolo 3 il modulo layout sviluppato in [13] viene utilizzato in questo lavoro di tesi come metodo principale di ricostruzione del layout, modificando alcuni dei processi interni al fine di migliorarne le prestazioni e adattarli ai nostri obiettivi.

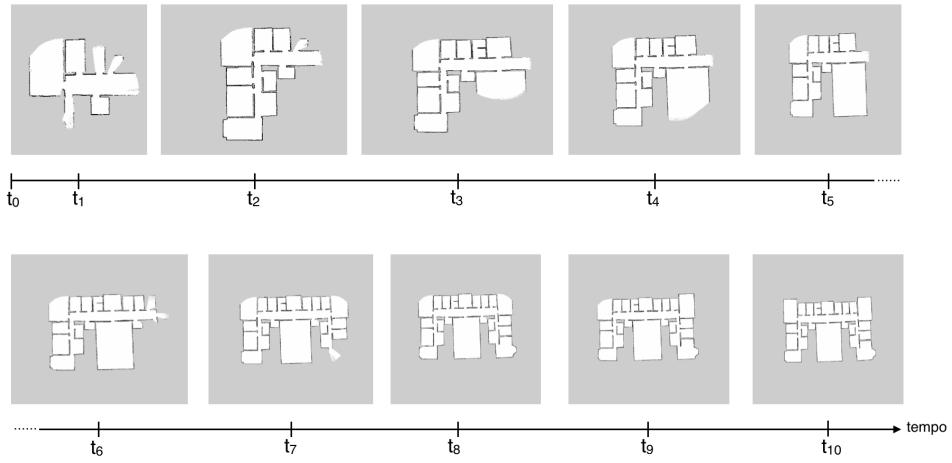


Figura 4.2: Esempio di mappe metriche parziali raccolte durante tutto il processo d'esplorazione relativo allo stesso ambiente.

Il modulo che crea il layout permette di ottenere una figurazione più dettagliata dei dati all'interno della mappa metrica a griglia, elevando la rappresentazione di un ambiente ad un livello d'astrazione superiore. Nel modulo che crea il layout le informazioni metriche della mappa metrica, ottenute grazie al modulo precedente (vedi Figura 4.1), vengono elaborate in modo da arricchire il contenuto informativo relativo ad un ambiente. Grazie a questo modulo, all'informazione metrica viene aggiunta una conoscenza sulla suddivisione dello spazio in aree differenti, ampliando di conseguenza la conoscenza che il robot possiede sull'ambiente circostante.

L'obiettivo finale di questo modulo è quello di ottenere una suddivisione ben precisa degli spazi che compongono l'ambiente nel quale il robot sta operando, fornendo come output l'insieme delle stanze con cui viene suddiviso l'edificio. L'idea che sta alla base del modulo di creazione del layout consiste nel:

- Ricercare all'interno della mappa metrica feature dette *segmenti*, le quali caratterizzano i muri dell'ambiente, permettendo di riconoscere e raggruppare gli ostacoli identificati nella mappa metrica.
- Raggruppare i segmenti trovati nella fase precedente all'interno di cluster, i quali separano segmenti che appartengono a muri differenti. Ad ognuno di questi cluster, in seguito, viene associata una *retta rappresentativa*.
- Riconoscere le intersezioni tra le rette rappresentative identificate nella fase precedente al fine di estrarre le *celle* che compongono l'ambiente.

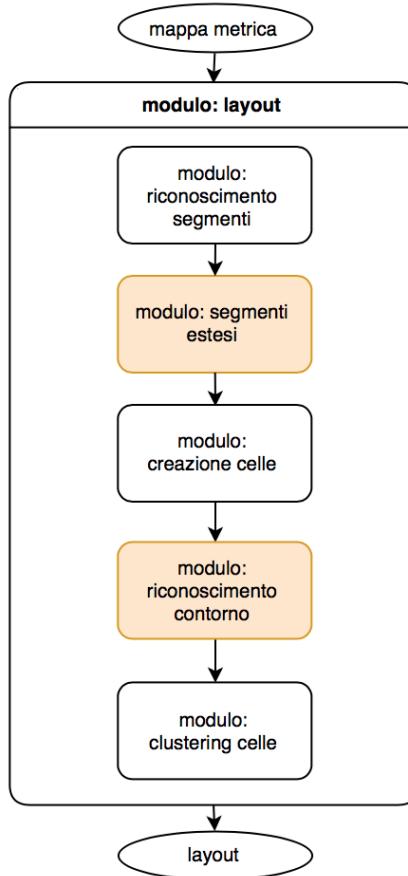


Figura 4.3: Modulo che crea il layout. L'immagine mostra il diagramma dei processi necessari al fine di ottenere una ricostruzione del layout, partendo da una mappa metrica parziale in input.

Le celle così create rappresentano le superfici nelle quali l'ambiente viene suddiviso.

- Raggruppare le celle ottenute all'interno di cluster differenti in modo da riconoscere correttamente le stanze, fornendo così l'output del sistema di ricostruzione del layout.

La Figura 4.3 mostra il modulo che crea il layout nel dettaglio. Le parti colorate in arancione rappresentano processi del sistema progettato in [13], che sono stati modificati in questo lavoro di tesi in modo da migliorare la costruzione del layout per le mappe metriche parziali in input. La restante parte di questa sezione volge a spiegare nel dettaglio i vari componenti del modulo che crea il layout come mostrato in Figura 4.3.

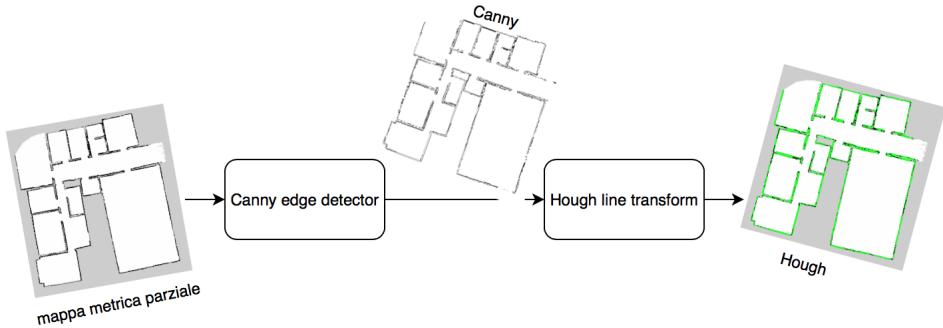


Figura 4.4: Processo di individuazione dei segmenti. L'input dell'algoritmo Canny edge detection è la mappa metrica. L'algoritmo di Canny riconosce i contorni dell'immagine, che vengono in seguito usati come input per l'algoritmo Hough line transform, il quale estrae i segmenti in corrispondenza dei contorni. Questi segmenti vengono visivamente identificati dal colore verde nell'ultima immagine a destra, la quale rappresenta l'output del processo.

4.2.1 Riconoscimento dei segmenti

Il modulo di riconoscimento dei segmenti all'interno della mappa metrica consiste nel primo problema affrontato nella fase di costruzione del layout delle stanze e permette di riconoscere quei segmenti che rappresentano le pareti all'interno della mappa. La Figura 4.4 mostra l'intero processo con cui sulla mappa metrica, ottenuta attraverso l'elaborazione dei dati provenienti dai sensori, vengono applicati sequenzialmente gli algoritmi di *Canny edge detection* [14] e *Hough line transform* [37].

L'algoritmo di Canny edge detection consente di estrarre i bordi della mappa metrica, permettendo di distinguere correttamente i muri all'interno di essa. L'algoritmo riconosce i bordi marcando i punti all'interno dell'immagine in cui l'intensità della luminosità cambia bruscamente. Questo indica che quel punto è parte di un muro. Il processo di edge detection serve a semplificare l'analisi di immagini riducendo drasticamente il numero di dati da processare, preservando allo stesso tempo le informazioni strutturali sui bordi degli oggetti che compongono l'immagine stessa. La Figura 4.5 (b) rappresenta il risultato dell'algoritmo di Canny usando come mappa metrica in input l'immagine in Figura 4.5 (a).

Il risultato viene in seguito usato come input per l'algoritmo di Hough line transform, il quale analizza i bordi estratti in precedenza e crea i segmenti necessari in corrispondenza dei muri dell'ambiente. Questo algoritmo è una tecnica di estrazione dei segmenti presenti all'interno di un'immagi-

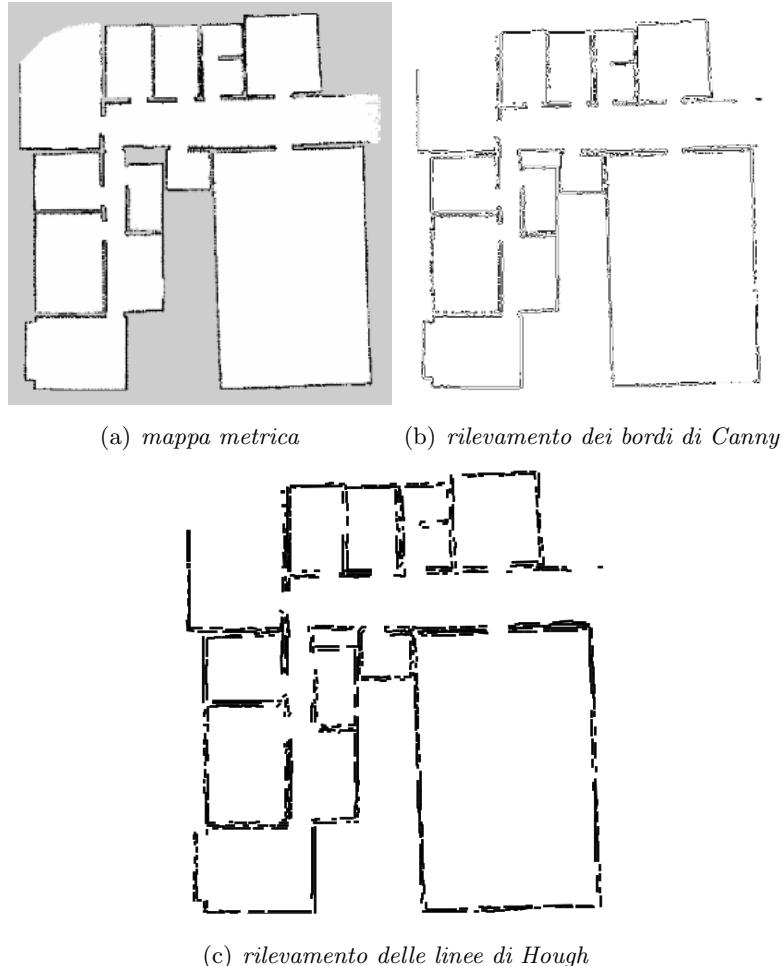


Figura 4.5: Processo con il quale è possibile riconoscere i muri all'interno della mappa metrica presente in (a). I muri corrispondono ai segmenti in (c).

ne. I segmenti vengono rappresentati attraverso le coordinate dei loro punti iniziali e finali.

La combinazione dei due algoritmi permette di riconoscere i muri. La Figura 4.5 (c) rappresenta il risultato dell'algoritmo di Hough line transform usando come input l'immagine in Figura 4.5 (b). L'intero processo, nel quale è possibile distinguere gli input di ogni fase che lo compongono, viene mostrato in Figura 4.4.

L'output dell'algoritmo di Hough line transform viene usato come input per il modulo successivo, che ha come obiettivo finale quello riconoscere le pareti dell'ambiente applicando un clustering sui segmenti e riconoscendo le

rette, tramite le cui intersezioni si può generare un insieme di celle.

4.2.2 Segmenti estesi

L'obiettivo principale del modulo segmenti estesi è quello di identificare i muri dell'edificio raggruppando in cluster i segmenti, identificati dall'algoritmo di Hough line transform, per creare un insieme di rette rappresentative, ognuna delle quali rappresenta un diverso cluster di segmenti.

La metodologia attualmente fornita dallo stato dell'arte nel lavoro in [13] raggruppa i segmenti che hanno coefficienti angolari simili all'interno dello stesso *cluster angolare* e permette di unire due segmenti all'interno dello stesso *cluster spaziale* se questi appartengono allo stesso cluster angolare e la distanza tra i loro punti medi, proiettati sulla retta perpendicolare all'orientamento del cluster angolare a cui entrambi i segmenti appartengono, non supera una soglia determinata a priori. Questa soglia prende il nome di distanza laterale tra segmenti. Il metodo proposto in [13] può portare al problema di un'attribuzione errata del cluster spaziale di appartenenza di un segmento, poiché raggruppa segmenti appartenenti a muri diversi assegnandoli alla stessa retta rappresentativa, portando così a rappresentare due muri differenti con un'unica retta.

Il modulo segmenti estesi rappresenta il primo dei moduli modificati per ovviare a questo problema. Il metodo proposto in questo elaborato riprende l'idea in [13], introducendo tuttavia un ulteriore passaggio che raggruppa i segmenti con ugual cluster angolare in base alla loro distanza, introducendo il concetto di *cluster muro*, da cui in seguito viene selezionato un segmento rappresentante. Sull'insieme dei segmenti rappresentanti viene successivamente eseguito il clustering spaziale. Questo passaggio intermedio permette di ridurre gli errori di assegnazione di un errato cluster spaziale ai segmenti.

Il metodo di raggruppamento dei segmenti proposto in questo lavoro di tesi viene diviso in tre stadi principali:

- **Il clustering angolare:** assegna ad ogni segmento un cluster angolare sulla base del proprio coefficiente angolare.
- **Identificazione dei muri:** raggruppa ulteriormente i segmenti con stesso cluster angolare assegnando ad ogni segmento il cluster muro. Dai cluster muro viene in seguito selezionato un segmento rappresentante.
- **Il clustering spaziale:** assegna ad ogni segmento rappresentante il proprio cluster spaziale.

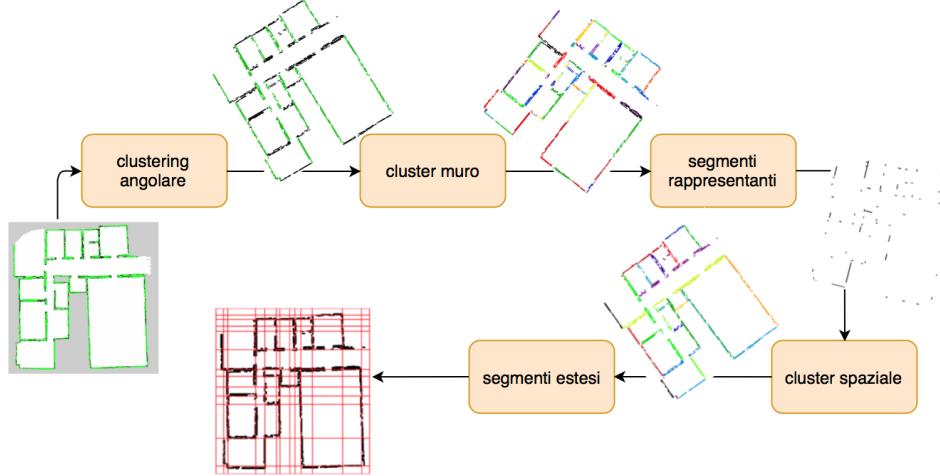


Figura 4.6: Processo nel quale si ottengono le rette che rappresentano i muri. Partendo dai segmenti identificati con l'algoritmo di Hough line transform, vengono calcolati in cascata per ogni segmento il cluster angolare ed il cluster muro. Per ogni cluster muro viene eletto un segmento rappresentante con il quale infine è possibile associare ad ogni segmento il proprio cluster spaziale, che verrà usato per la costruzione delle rette, identificate dal colore rosso nell'ultima immagine fornita dal processo.

La Figura 4.6 presenta un'esecuzione del metodo mostrando input ed output per ogni fase.

Infine si crea, grazie al clustering spaziale, una retta associata ad ogni cluster spaziale, fornendo in questo modo un insieme di rette grazie alle quali è possibile estrarre l'insieme di celle che costituisce il mondo.

Di seguito vengono forniti i dettagli degli stadi che costituiscono il processo all'interno del modulo segmenti estesi.

Il clustering angolare

Il clustering angolare viene effettuato sulla base dell'orientamento dei segmenti. L'orientamento di un segmento è l'angolo formatosi tra la retta su cui giace il segmento e l'asse x del sistema di riferimento. La Figura 4.7 mostra un esempio. Una volta associato l'angolo che identifica l'orientamento di ogni segmento al segmento stesso, i cluster angolari vengono ricavati utilizzando l'algoritmo di *mean shift*, descritto in [17], il quale prende in considerazione l'orientamento dei segmenti appena ricavato. L'algoritmo mean shift è una tecnica di clustering non parametrica che non richiede alcuna conoscenza preliminare sul numero di cluster. I dati in ingresso P all'algoritmo di mean shift rappresentano gli orientamenti dei segmenti. L'algoritmo viene appli-

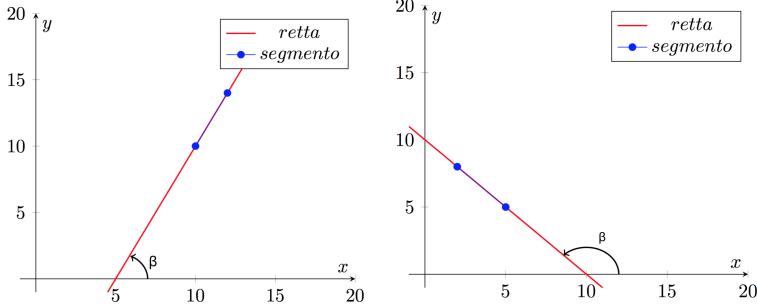


Figura 4.7: Esempio di come si associa un angolo ad un segmento.

cato in maniera iterativa. Il primo step prevede che il numero di *cluster center* coincida con tutti gli orientamenti possibili dei segmenti. Per ogni step i successivo, vengono calcolati i nuovi cluster center t_i come segue:

$$m(t_i) = \frac{\sum_{p \in P} K(p - t_i)p}{\sum_{p \in P} K(p - t_i)} \quad (4.1)$$

dove $m(t_i)$ rappresenta la *sample mean* e K rappresenta una *kernel function*.

La kernel function, usata per calcolare la sample mean espressa in (4.1) e dunque per estrarre i cluster angolari, è:

$$K(\theta) = \begin{cases} [1 - \frac{1-\cos(\alpha-\theta)}{h}]^2 & se \frac{1-\cos(\alpha-\theta)}{h} \leq 1 \\ 0 & se \frac{1-\cos(\alpha-\theta)}{h} > 1 \end{cases} \quad (4.2)$$

dove α corrisponde al cluster center corrente, θ corrisponde all'orientamento del segmento e h rappresenta una costante arbitraria. Il processo termina quando $\max(m(t_i) - t_i)$ è minore di una soglia minima, determinata a priori e sufficientemente piccola (nel nostro caso settata al valore di 0.00001).

Questo processo permette di inserire i segmenti con orientamento simile nello stesso cluster angolare, fornendo dunque una prima misura di similarità tra elementi nello stesso cluster. La Figura 4.8 (a) mostra un esempio, in cui sono stati riconosciuti correttamente due cluster angolari differenti, identificati dal colore verde e nero.

Identificazione dei muri

Una volta trovato il cluster angolare di ogni segmento, mostrato in Figura 4.8 (a) con i colori verde e nero, si è pronti per la fase successiva. In essa ogni cluster angolare viene raggruppato ulteriormente attraverso un metodo di clustering chiamato DBSCAN [26]. Questo produce cluster contenenti segmenti tra loro vicini, fornendo una prima approssimazione di quali siano

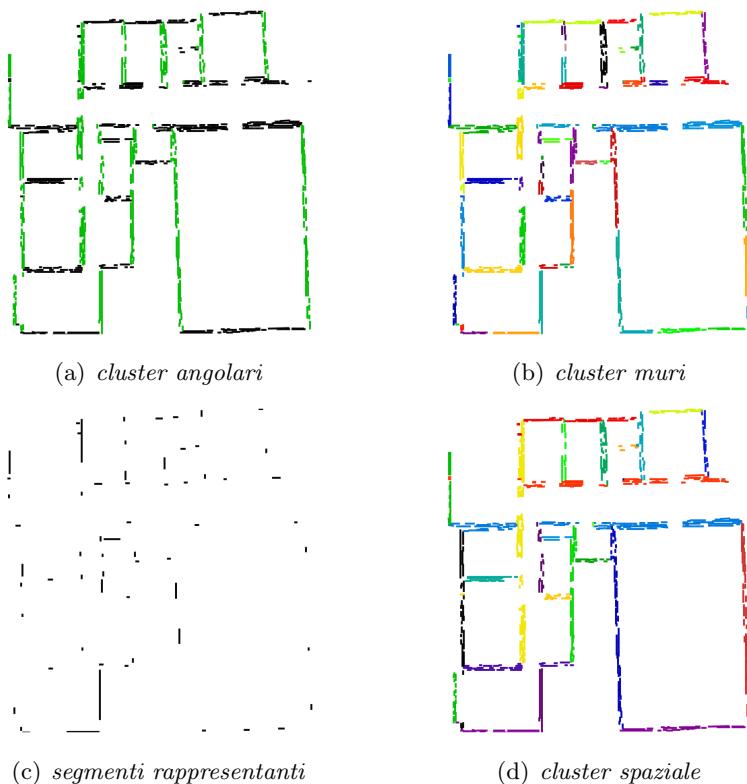


Figura 4.8: La divisione in cluster angolari viene mostrata in (a) in cui si possono distinguere sia il cluster orizzontale che quello verticale, rispettivamente contrassegnati dai colori nero e verde. (b) mostra la divisione in cluster muri. Colori diversi sono assegnati a cluster diversi. (c) mostra i rappresentanti dei cluster muri. (d) mostra i cluster spaziali, ai quali vengono assegnati colori differenti.

i muri dell’edificio, come viene mostrato in Figura 4.8 (b).

L’algoritmo DBSCAN presentato in [26] è un algoritmo di clustering che si basa sul concetto di densità. L’idea che sta alla base dell’algoritmo è che per ogni punto p , interno ad un cluster, debbano esistere un numero minimo di altri punti in un intorno di p (con raggio definito), che appartengono allo stesso cluster.

Una volta definito ϵ come il parametro che indica il raggio di vicinanza ad un punto p , è possibile definire $N_\epsilon(p)$ come ϵ -vicinato di un punto p come segue:

$$N_\epsilon(p) = \{q \in D \mid dist(p, q) \leq \epsilon\} \quad (4.3)$$

dove D rappresenta il dataset e $dist(p, q)$ rappresenta la funzione che fornisce una misura di distanza tra i punti p e q . Inoltre per ogni punto p di un cluster, il parametro $minPts$ indica il numero minimo di punti vicini compresi nell’ ϵ -vicinato di p .

Tutti i punti all’interno del dataset vengono classificati in 2 modi:

- **Core point.** Nel dataset ogni punto p che possiede un numero di punti nell’ ϵ -vicinato uguale o maggiore del parametro $minPts$ viene marcato come core point.
- **Border Point.** Nel dataset ogni punto p che possiede un numero di punti nell’ ϵ -vicinato minore del parametro $minPts$ viene marcato come border point.

Come descritto in [26] per comprendere meglio l’algoritmo, è necessario fornire le seguenti definizioni:

- **Directly density-reachable.** Un punto p nel dataset è directly density-reachable da un punto q nel dataset se:
 1. $p \in N_\epsilon(q)$. Ovvero se p è nell’ ϵ -vicinato di q .
 2. $|N_\epsilon(q)| \geq minPts$. Ovvero q è un core point.
- **Density-reachable.** Un punto p nel dataset è density-reachable da un punto q nel dataset, se esiste una catena di punti p_1, \dots, p_n con $p_1 = q$ e $p_n = p$ tale che p_{i+1} risulta essere directly density-reachable da p_i .
- **Density-connected.** Un punto p risulta essere density-connected ad un punto q se esiste un punto o nel dataset tale che sia p che q risultano essere density-reachable da o .

- **Cluster.** Sia D il dataset che contiene tutti i punti. Un cluster C è un sottoinsieme di D non vuoto che soddisfa le seguenti condizioni:
 1. $\forall p, q \in D$: se $p \in C$ e q è density-reachable da p , allora $q \in C$.
 2. $\forall p, q \in C$: p è density-connected a q .
- **Outlier.** Siano $C_1 \dots C_k$ i cluster di un dataset D . Gli outlier di un dataset D sono tutti quei punti in D che non appartengono ad alcun cluster. $Outlier = \{p \in D | \forall i : p \notin C_i\}$.

L'algoritmo per ogni punto p calcola la sua distanza dagli altri punti del dataset e marca come core point tutti i punti il cui ϵ -vicinato contiene un numero di punti maggiore o uguale al parametro $minPts$. A questo punto si crea un nuovo cluster per ogni core point c che non è stato ancora assegnato ad alcun cluster. In seguito vengono assegnati allo stesso cluster di c anche tutti i punti density-connected da c . Terminato questo processo, tutti i punti che non appartengono ad un cluster vengono marcati come outlier.

Nel progetto svolto in questa tesi, il dataset su cui viene eseguito l'algoritmo DBSCAN corrisponde all'insieme dei segmenti che possiedono lo stesso cluster angolare. Questo significa che l'algoritmo DBSCAN viene eseguito un numero di volte pari al numero di cluster angolari individuati nella fase precedente, per ottenere una migliore classificazione dei segmenti che appartengono allo stesso cluster angolare. La Figura 4.8 (b) mostra un esempio nel quale ogni cluster viene rappresentato con un colore diverso.

Prima di poter effettivamente eseguire l'algoritmo DBSCAN, è necessario fornire una misura di affinità tra segmenti. Questa misura viene calcolata per ogni possibile coppia di segmenti l_i ed l_j all'interno dell'insieme di segmenti aventi lo stesso cluster angolare C_a e viene rappresentata con una matrice bidimensionale di dimensione $|C_a| \times |C_a|$:

$$A(C_a) = \begin{pmatrix} d_{11}(l_1, l_1) & \dots & d_{1N}(l_1, l_N) \\ \vdots & \ddots & \vdots \\ d_{N1}(l_N, l_1) & \dots & d_{NN}(l_N, l_N) \end{pmatrix}$$

dove $N = |C_a|$ e $d(l_i, l_j)$ rappresenta la distanza tra i segmenti l_i ed l_j , definita come la distanza minima tra i punti appartenenti a due segmenti. Ogni elemento $A(C_a)_{ij}$ dunque rappresenta il valore di affinità locale tra i segmenti l_i ed l_j . Questa matrice ha un ruolo fondamentale per ottenere i cluster che rappresentano i muri, perché l'obiettivo è quello di raggruppare

segmenti all'interno dello stesso cluster che abbiano un'alta affinità. L'algoritmo DBSCAN crea inoltre un cluster degli outlier, ossia segmenti che non appartengono ad alcun cluster. Per non perdere informazioni, ogni segmento outlier viene in seguito assegnato al cluster muro più vicino. In Figura 4.8 (b) viene riportato il risultato di tale operazione, dove ad ogni cluster muro viene associato un colore diverso. Terminata questa fase si è pronti a selezionare il segmento rappresentante di ogni cluster.

Selezione del segmento rappresentante

Una volta ottenuto il cluster muro di ogni segmento, si passa alla fase di selezione di un segmento come rappresentante del cluster, la Figura 4.8 (c) ne mostra un esempio. Ogni cluster muro viene dunque rappresentato solamente da uno dei suoi segmenti interni che verrà utilizzato nella fase successiva per identificare i cluster spaziali. Il rappresentante di ogni cluster muro viene selezionato in base alla propria posizione nello spazio. In particolare viene selezionato il segmento mediano del cluster, cioè quello che si trova nel mezzo del cluster. Questo segmento viene selezionato come segue.

Per ogni cluster muro viene scelto, in modo casuale, un segmento candidato l_c di cui viene calcolato il punto medio x_{cm}, y_{cm} . Viene tracciata la retta r ortogonale alla retta su cui giace il segmento candidato l_c e passante per il punto medio x_{cm}, y_{cm} .

Per ogni altro segmento l_i appartenente allo stesso cluster muro, viene calcolato il punto medio x_{im}, y_{im} e si definisce la retta r_i avente lo stesso coefficiente angolare del segmento candidato l_c e passante per il punto x_{im}, y_{im} . Il punto d'intersezione tra la retta r e la retta r_i rappresenta la proiezione del punto medio di ogni segmento sulla retta r .

Applicando una rototraslazione del sistema di riferimento originario xOy , usando come origine il punto $P(x_{cm}, y_{cm})$,

$$\begin{cases} x' = x_{cm} + x \cos \alpha - y \sin \alpha \\ y' = y_{cm} + x \sin \alpha + y \cos \alpha \end{cases} \quad (4.4)$$

dove α rappresenta l'angolo di rotazione, si ottiene un nuovo sistema di riferimento $x'O'y'$, dove $O' = \{x_{cm}, y_{cm}\}$ (vedi grafico di Figura 4.9).

I punti originariamente proiettati sulla retta r saranno trasformati in punti sull'asse x' . Il punto mediano corrisponderà così al segmento scelto come rappresentante del cluster. Nel caso in cui il numero di elementi all'interno dell'insieme dei segmenti del cluster fosse pari, esisteranno due segmenti mediani e l'algoritmo ne selezionerà uno soltanto.

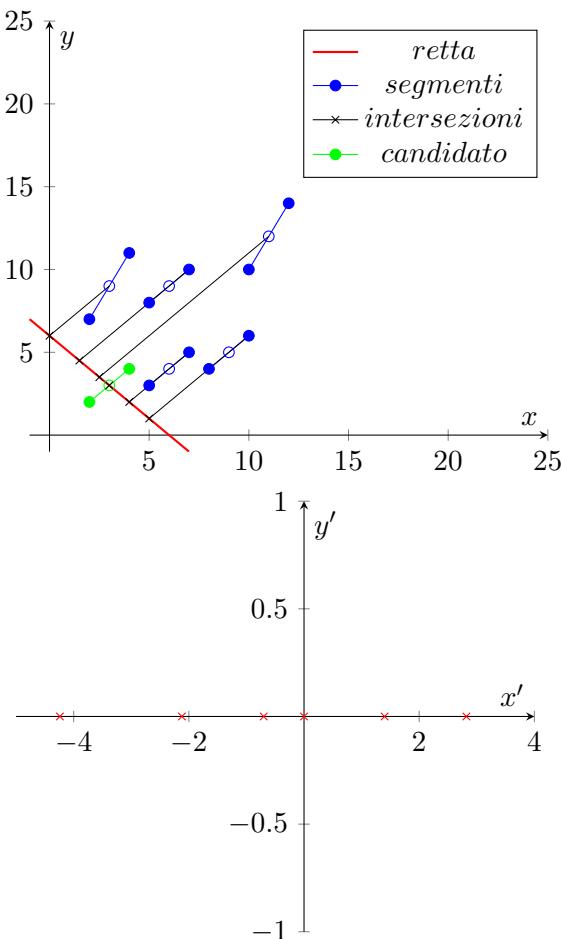


Figura 4.9: Nell'immagine in alto si possono distinguere i centri dei segmenti proiettati lungo la retta (in rosso) ortogonale alla retta su cui giace il segmento candidato (in verde) e passante per il suo punto medio. Nell'immagine in basso il risultato della rototraslazione applicata al primo segmento, che viene dunque situato nell'origine del nuovo spazio.

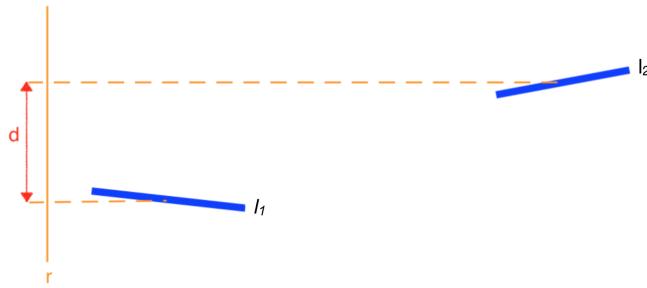


Figura 4.10: Esempio di distanza laterale. l_1 ed l_2 rappresentano due segmenti rappresentanti con ugual cluster angolare. La distanza laterale d viene calcolata come distanza tra i punti medi di l_1 ed l_2 proiettati sulla retta r .

Il clustering spaziale

L'obiettivo del clustering spaziale è quello di riconoscere i muri. A tale scopo ogni segmento rappresentante viene unito ad altri rappresentanti che appartengono allo stesso cluster angolare, se e solo se la loro distanza laterale è minore di una soglia stabilita a priori. Questo procedimento permette di assegnare un cluster spaziale ad ogni segmento rappresentante.

Perché a due segmenti rappresentanti possa essere attribuito lo stesso cluster spaziale, è necessario considerare una misura che indichi la prossimità spaziale tra i due. Questa misura è relativa alla loro distanza laterale d . Seguendo la definizione fornita in [13], dati due segmenti l_1 ed l_2 , la loro distanza laterale d corrisponde alla distanza tra i punti medi dei due segmenti proiettati sulla retta perpendicolare all'orientamento del cluster angolare a cui entrambi i segmenti appartengono. La Figura 4.10 mostra un esempio. I segmenti l_1 ed l_2 rappresentano due segmenti rappresentanti che possiedono un orientamento simile, i quali possono essere approssimati a segmenti orizzontali. I punti medi dei due segmenti vengono proiettati sulla retta r , che rappresenta la retta perpendicolare alla retta con orientamento uguale al cluster angolare dei due segmenti l_1 ed l_2 . Dunque se due segmenti l_1 ed l_2 possiedono una distanza d inferiore ad una soglia arbitraria definita a priori, possono essere considerati come appartenenti allo stesso cluster angolare. Terminata la fase appena descritta, i rappresentanti di ogni cluster devono comunicare il proprio cluster spaziale a tutti i segmenti che hanno il loro stesso cluster muro. Questo processo permette di riconoscere i muri dell'edificio. Il risultato di tale processo viene mostrato in Figura 4.8 (d) in cui i segmenti che possiedono lo stesso colore appartengono anche allo stesso cluster spaziale. La Figura 4.11 mostra un esempio più dettagliato

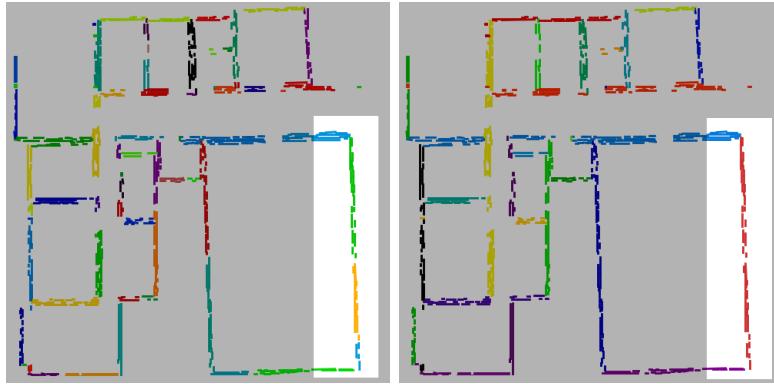


Figura 4.11: Esempio di come i cluster muri vengono uniti a formare un cluster spaziale. La parte evidenziata dell'immagine a sinistra mostra tre cluster muri verticali (azzurro, arancione, verde) che vengono uniti in un unico cluster spaziale (in rosso) visibile nella parte evidenziata dell'immagine a destra.

nel quale viene evidenziato un cluster spaziale risultato dell'unione di tre cluster muro.

Segmenti estesi

Ottenuti i cluster spaziali, ad ognuno di essi viene associata una retta, la quale rappresenta il muro a cui il cluster spaziale si riferisce. Le rette rappresentative vengono identificate da un punto e dall'orientamento. Il punto costituisce il punto mediano dei punti medi dei segmenti che fanno parte dello stesso cluster spaziale, mentre l'orientamento viene dato dal cluster angolare dei segmenti appartenenti allo stesso cluster spaziale. I segmenti estesi, linee rosse in Figura 4.12, si ottengono dalle rette rappresentative prendendo solo la porzione di retta compresa nella *bounding box* dell'immagine e vengono rappresentate con linee caratterizzate da un punto iniziale ed uno finale.

4.2.3 Creazione celle

L'insieme delle rette rappresentative genera, per mezzo delle loro intersezioni, un insieme di celle, come mostra la Figura 4.12. L'obiettivo finale è quello di raggruppare le celle e trovare dunque le stanze, in modo da ottenere un modello che mostri la disposizione e la separazione delle diverse stanze che compongono l'edificio. Si assume infatti che una stanza sia composta da almeno una cella. Per raggruppare le celle in stanze, come è stato fatto nella fase precedente per raggruppare i segmenti in base all'orientamento ed alla distanza laterale, si fa uso di metodi di clustering che richiedono una

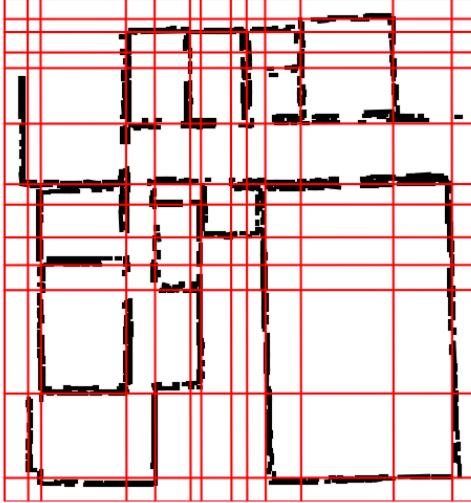


Figura 4.12: Insieme dei segmenti estesi, distinguibili dal colore rosso. Le intersezioni tra le rette rappresentative dei cluster spaziali, dalle quali vengono estratti i segmenti estesi, formano inoltre l’insieme delle celle che suddividono l’ambiente. I segmenti estesi esterni rappresentano la bounding box dell’immagine.

misura di affinità tra le celle. Una soluzione per valutare l'affinità tra due celle adiacenti consiste nel fornire un peso all'edge che le separa. Gli edge sono i segmenti con i quali si rappresentano i bordi di una cella e che si ottengono dalle intersezioni tra le rette rappresentative. Il peso associato agli edge, nel lavoro in [13] come in [62], rappresenta la probabilità che tale edge sia realmente un muro.

Assegnamento del peso agli edge delle celle

Seguendo la definizione fornita in [62]: siano f_i ed f_j due celle adiacenti, che hanno in comune l'edge e_{ij} . All'edge e_{ij} viene assegnato un peso w_{ij} che indica la probabilità che esso appartenga realmente ad un muro presente all'interno della mappa metrica. Per far ciò si considerano tutti i relativi muri candidati (tutti i segmenti che hanno lo stesso cluster spaziale dell'edge e_{ij}), proiettandoli sull'edge e_{ij} . La Figura 4.13 mostra un esempio nel quale viene evidenziato un edge. Tutti i segmenti vengono rappresentati in blu. Si definisce $cov(e_{ij})$ la porzione di edge coperta dalla proiezione dei segmenti su di essa, rappresentata con il colore verde, mentre la parte in rosso rappresenta la porzione di edge non coperta da proiezioni. È così possibile definire il peso dell'edge come segue:

$$w_{ij} = \frac{cov(e_{ij})}{length(e_{ij})} \quad (4.5)$$

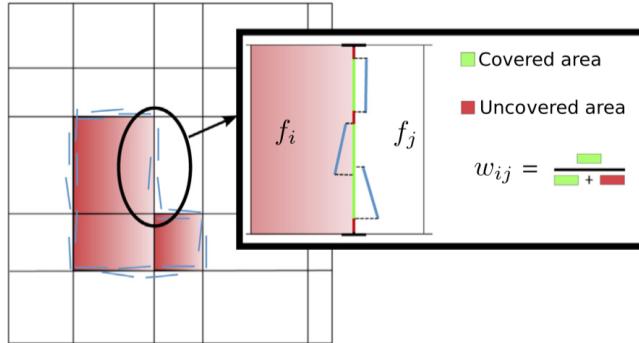


Figura 4.13: Calcolo della copertura di un edge tratto da [62]. I segmenti con lo stesso cluster spaziale dell'edge (in blu) vengono proiettati sull'edge stesso. Infine viene calcolato il peso dell'edge come il rapporto tra la lunghezza dell'edge occupata dalla proiezione dei segmenti e la lunghezza totale dell'edge.

dove $length(e_{ij})$ è la lunghezza del segmento e_{ij} , che coincide con la somma delle porzioni rosse e verdi nell'edge evidenziato in Figura 4.13.

Perché il processo di clustering abbia successo nell'identificare le stanze, è però necessario che venga eseguito solo sulle celle interne all'edificio, cioè sulle celle che sono state almeno in parte esplorate dal robot. Tali celle sono identificate riconoscendo il contorno della mappa metrica, permettendo di riconoscere quelle regioni di spazio realmente esplorate dal robot.

4.2.4 Riconoscimento del contorno

Il processo di riconoscimento del contorno permette di riconoscere la superficie della mappa metrica corrispondente alla porzione interna dell'edificio. Ciò può essere eseguito individuando i contorni di un'immagine. Nel caso in questione l'immagine corrisponde alla mappa metrica, come mostra la Figura 4.14. La prima operazione che compie il processo consiste nel manipolare l'immagine della mappa metrica colorando lo spazio libero (spazio bianco della mappa metrica di Figura 4.14) di nero e lo spazio sconosciuto (spazio grigio di Figura 4.14) di bianco. Questo meccanismo permette di creare una netta distinzione tra le zone libere e quelle sconosciute. Successivamente il processo estrae l'insieme dei contorni all'interno dell'immagine e seleziona quello la cui area risulta essere maggiore rispetto agli altri. Questo perché il contorno con area maggiore possiede una maggior probabilità di essere quello che avvolge le mura perimetrali dell'edificio e che racchiude dunque tutta (e solo) la parte già esplorata dal robot. In questa fase, le frontiere

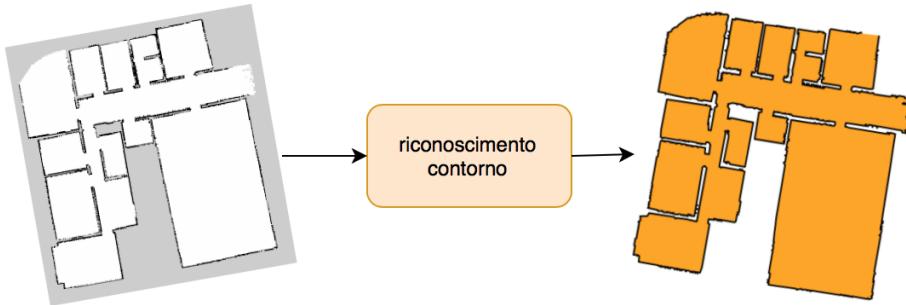


Figura 4.14: Processo con il quale si ottiene il contorno che avvolge le mura perimetrali della mappa metrica parziale fornita in input.

della mappa metrica parziale vengono considerate come parte del contorno esterno.

Grazie al riconoscimento del contorno è possibile suddividere le celle, ottenute grazie alle intersezioni tra le rette rappresentative, in *interne* ed *esterne*. Le celle esterne rappresentano regioni dello spazio sulle quali il robot non ha raccolto abbastanza dati sensoriali. Questo avviene analizzando l'intersezione tra le celle ed il contorno esterno. Tutte le celle, la cui intersezione con il contorno è nulla o inferiore ad una soglia δ determinata a priori, vengono classificate come esterne. In questo lavoro di tesi il valore di δ è stato impostato ad un quinto dell'area della cella che si sta valutando. Ciò significa che, se l'area frutto dell'intersezione tra la cella f_i ed il contorno risulta essere maggiore di un quinto dell'area della cella f_i , la cella f_i viene considerata come una cella interna. La Figura 4.15 mostra un esempio. L'immagine a sinistra rappresenta il contorno esterno sovrapposto alle celle che non sono state ancora classificate come interne o esterne. L'immagine a destra mostra il risultato della classificazione in celle interne ed esterne, nella quale le celle interne vengono rappresentate con il colore arancione.

4.2.5 Clustering celle

Una volta ottenuta la divisione tra celle interne ed esterne, l'obiettivo principale è quello di raggruppare tali celle in cluster che ne rappresentano la divisione in stanze. Anche in questo caso, come è stato precedentemente fatto per ottenere i cluster muri, si è optato per l'utilizzo dell'algoritmo DBSCAN. A questo scopo è necessario stabilire una misura di affinità tra coppie di celle interne f_i ed f_j . Per far ciò, prendendo spunto da [62], vengono usati i pesi degli edge delle celle per ottenere una matrice di affinità L nella quale

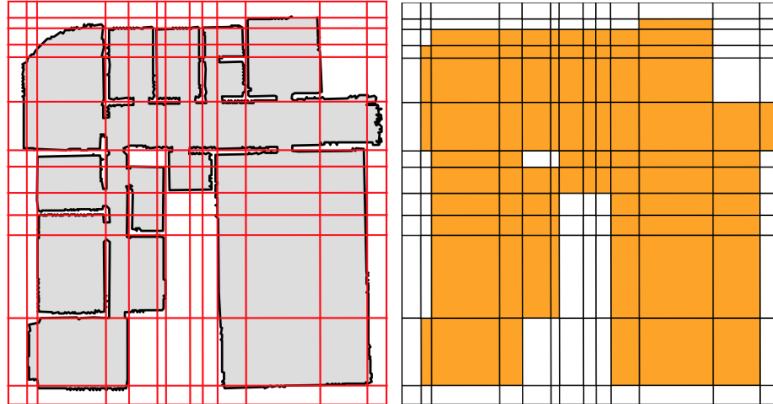


Figura 4.15: Esempio di classificazione delle celle in interne ed esterne. A sinistra si può vedere la sovrapposizione tra le celle ed il contorno esterno. A destra si può vedere il risultato della classificazione in celle interne ed esterne.

ogni valore è definito come:

$$L_{ij} = \begin{cases} e^{-w_{ij}/\sigma} & \text{se } i \neq j \wedge f_i, f_j \text{ sono adiacenti} \\ 1 & i = j \\ 0 & \text{altrimenti} \end{cases} \quad (4.6)$$

Per mezzo della matrice L viene definita una matrice di transizione $M = D^{-1}L$ in cui la matrice D rappresenta una matrice diagonale della forma $D = diag(\sum_{j=1}^{j=n} L_{ij})$. Essa corrisponde ad una matrice quadrata in cui solo i valori sulla diagonale principale possono essere diversi da zero e rappresentano la somma di tutti i valori all'interno della stessa riga di L . Ogni elemento M_{ij} può essere visto come il valore di affinità locale tra le celle f_i ed f_j , considerando solamente le coppie di celle direttamente connesse. Grazie alla matrice M è così possibile raggruppare nello stesso cluster celle che hanno un'alta affinità.

Per ogni coppia di celle f_i ed f_j , DBSCAN analizza il valore $d_{ij} = 1 - M_{ij}$, che indica la distanza tra le celle f_i ed f_j in termini di affinità. Questo viene fatto per valutare se quelle due celle appartengono alla stessa stanza. Una volta completato il processo, i cluster trovati rappresentano le stanze. Il modello così ottenuto rappresenta l'output del modulo che crea il layout. In Figura 4.16 viene mostrato un esempio.

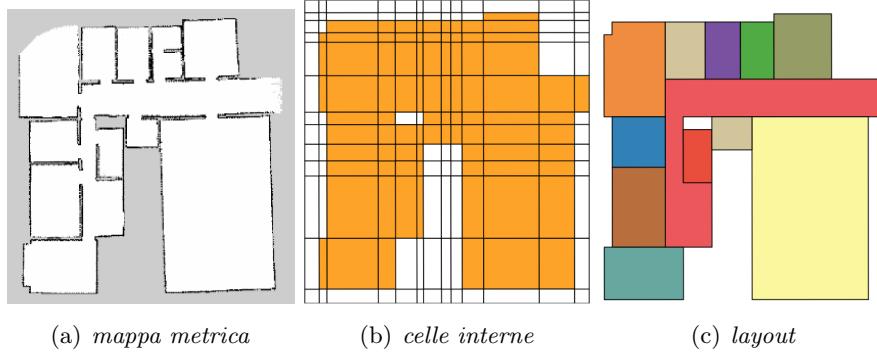


Figura 4.16: Esempio di processo che porta alla costruzione del layout.

4.3 La predizione

Come si può notare nella Figura 4.17, la quale fornisce una visione generale di tutti i componenti che formano il processo, il modulo di predizione prevede in input due informazioni precedentemente calcolate: la mappa metrica ed il layout. Queste informazioni sono necessarie al fine di fornire un’ipotesi di completamento di una stanza parziale. Le ipotesi di completamento di una stanza parziale vengono elaborate prendendo in considerazione tutte le informazioni che derivano dalla fase precedente in cui si calcola il layout. Il layout però non fornisce alcuna conoscenza su quali siano le stanze parziali. È grazie alla mappa metrica che è possibile riconoscerle.

Come già affermato in precedenza, quando si parla di mappe metriche parziali ci si riferisce al risultato ottenuto da una esplorazione robotica non ancora terminata. Una tecnica d’esplorazione di grande rilevanza è quella della frontier-based exploration [99]. In essa è necessario riconoscere le frontiere oltre le quali il robot mobile non è ancora stato ed assegnargli un valore, così da indirizzare il robot verso quella che risulta essere la più promettente. Quando si vuole fornire una predizione di un ambiente, è utile capire in quale porzione della mappa quest’operazione debba essere svolta. Bisogna capire, dunque, quali tra le stanze riconosciute nel layout siano complete e quali invece non lo siano. Ragionevolmente una stanza nella quale si presenta una frontiera sarà parziale. Questo ragionamento risiede infatti alla base del metodo di riconoscimento delle stanze parziali, sulle quali si tenterà in seguito di fornire un’ipotesi di completamento.

L’idea che sta alla base del modulo di predizione delle stanze parziali consiste, quindi, nel riconoscere le frontiere utilizzando la mappa metrica, nel riconoscere le stanze parziali del layout ed infine nel compiere il processo

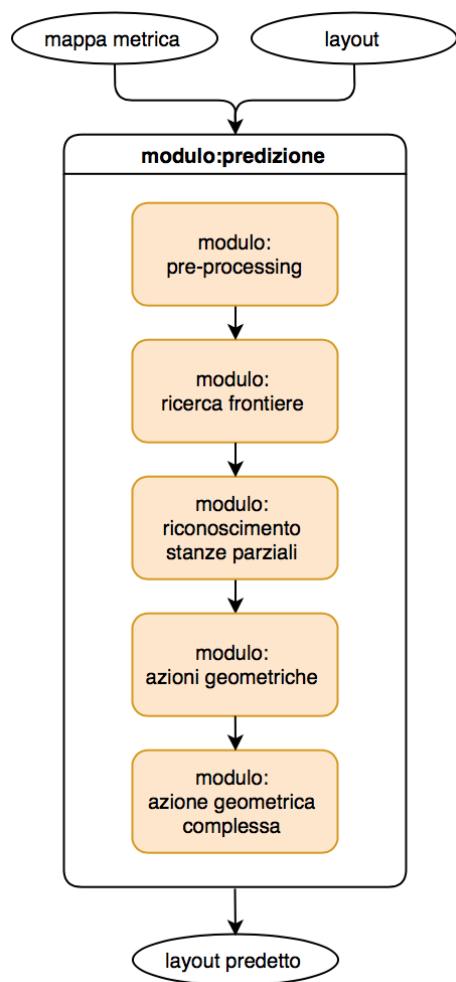


Figura 4.17: Modulo predizione. L'immagine mostra il diagramma dei processi necessari al fine di ottenere una predizione, partendo da una mappa metrica parziale in input e dal layout ottenuto con il modulo precedente.

di predizione fornendo un’ipotesi di completamento per ognuna di esse.

Nella restante parte del capitolo vengono descritti tutti gli elementi che compongono il modulo predizione.

4.3.1 Pre-processing

La fase di pre-processing viene eseguita prima di compiere qualsiasi operazione sul layout. Nonostante l’algoritmo di clustering, che unisce le celle in stanze, funzioni abbastanza bene nella maggior parte dei casi, a volte potrebbe sbagliare. Questo accade ad esempio quando, durante il processo di costruzione del layout, si riscontrano due celle adiacenti di dimensione molto piccola che dovrebbero appartenere a due stanze diverse, il cui edge in comune però corrisponde ad una porta. La presenza di quell’edge debole, al quale non corrispondono segmenti in grado di fornire un peso adeguato, non permette all’algoritmo DBSCAN di identificare una corretta separazione delle due stanze, che vengono unite. Al contrario, invece, può accadere che nella mappa metrica in ingresso vi sono lunghi corridoi che potrebbero essere separati erroneamente in differenti cluster.

Questi due problemi prendono il nome di:

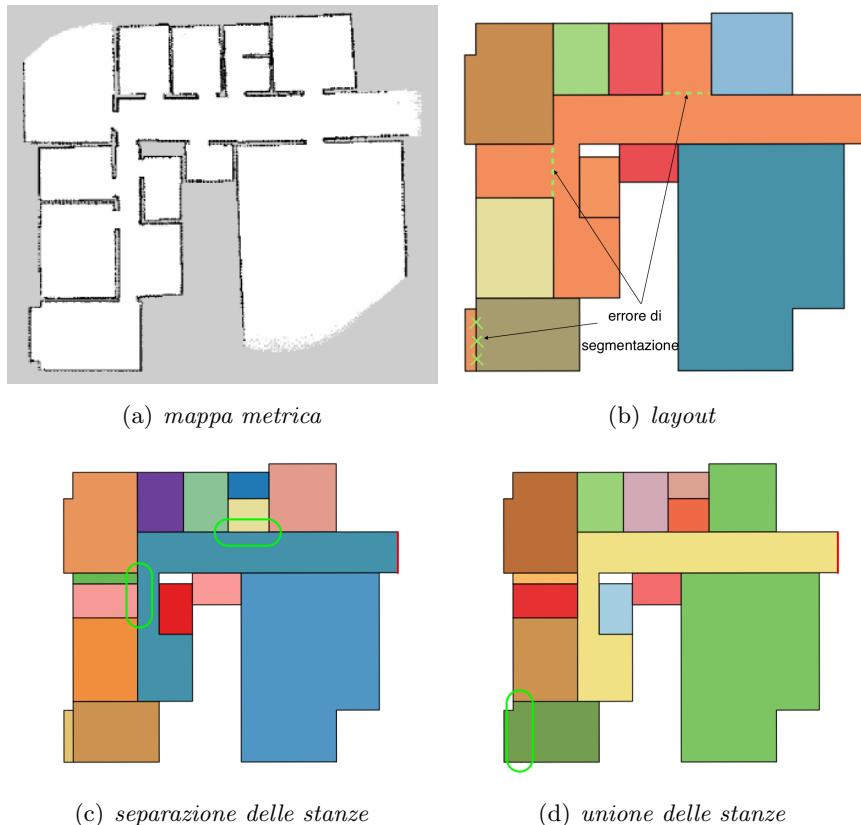
- **Under-segmentazione:** avviene quando il numero di stanze, che si ottengono con il processo che ricostruisce il layout, è inferiore al numero di stanze reali. In particolare, esiste almeno una stanza del layout che corrisponde a due o più stanze reali.
- **Over-segmentazione:** avviene quando il numero di stanze ottenute dal processo che ricostruisce il layout è maggiore del numero delle stanze reali. Ovvero, quando ad una stanza reale vengono associate due o più stanze del layout.

Nella Figura 4.18 (b) si può notare un layout di esempio in cui sono presenti entrambi i problemi sopracitati.

La fase di pre-processing tenta di fornire una metodo che sia in grado di correggere alcuni degli errori sopracitati. In particolare tenta di separare stanze che devono essere unite ed unire, successivamente, quelle che invece sono state separate erroneamente.

Separazione delle stanze

Come accennato in precedenza il problema dell’under-segmentazione è dipendente da un edge di peso debole, probabilmente causato dal fatto che in quella zona esiste una porta aperta. La separazione delle stanze, che si



*Figura 4.18: L'immagine (b) rappresenta un esempio nel quale è possibile trovare sia il problema dell'*under-segmentazione* che quello dell'*over-segmentazione*. Le linee tratteggiate verdi rappresentano una parete nascosta che non è stata considerata per separare due stanze. Al contrario le x verdi rappresentano una parete che è stata creata erroneamente. Le immagini (c) e (d) forniscono un esempio di come il modulo di pre-processing agisca, tentando di fornire una soluzione ai due problemi. Le parti evidenziate in verde nell'immagine (c) rappresentano delle pareti aggiunte, mentre la parte evidenziata in verde nell'immagine (d) mette in mostra come sia sparita una parete erroneamente considerata in precedenza.*

propone in questo lavoro di tesi, consiste nel ricercare le porte dell'ambiente per poterle chiudere virtualmente. In particolare si ricercano tutti gli edge di una cella che corrispondono ad una porta e gli si assegna forzatamente un peso $w(e_{ij}) = 1$.

La ricerca delle porte usata in questo lavoro di tesi, si basa sul *medial axis* [20]. Esso, analogamente al Voronoi graph (Sezione 2.4.1) viene definito come l'insieme di punti aventi più di un ostacolo alla stessa distanza minima. Il medial axis di una figura, però, viene definito quando la figura stessa viene incorporata in uno spazio euclideo ed è dotato di una funzione di distanza.

Per riconoscere le porte è dunque necessario applicare alla mappa metrica la *Euclidean distance transform*. La Euclidean distance transform consiste in una trasformazione in cui il valore di ogni cella dello spazio libero, all'interno della mappa metrica, viene sostituito con il valore della distanza euclidea dalla cella occupata più vicina ad essa. Un esempio del risultato ottenuto con questo processo viene mostrato in Figura 4.19 (a). Alla mappa, risultato della distance transform, viene applicato il medial axis. Esso si ricava analizzando il valore delle distanze d dagli ostacoli più vicini associate ad ogni cella della mappa metrica. In particolare appartengono al medial axis le celle c che possiedono un valore $d > 0$ per cui esiste più di una cella con valore diverso da 0 e la cui distanza da c è pari a d .

A questo punto è necessario ricercare i punti critici, definiti nella Sezione 2.4.1. Essi rappresentano, in questo caso, tutti i punti del medial axis che sono più vicini agli ostacoli rispetto ad altri punti. Il risultato viene mostrato in Figura 4.19 (c).

Una volta ottenuti i punti critici, è necessario stabilire quali si riferiscono a passaggi stretti e quali invece no. La soluzione proposta consiste nel ricercare quali punti critici ricadano esattamente su uno degli edge delle celle che formano una stanza. Essi con molta probabilità si riferiscono ad un passaggio stretto, dunque ad una porta. Prima di poter ricercare le porte, è però necessario riconoscere le stanze under-segmentate. Queste sono tutte le stanze al cui interno è stata cancellata una parete presente nella mappa metrica. Dunque per ogni stanza e per ogni coppia di celle che compongono la stanza, si controlla che i punti critici non ricadano sull'edge in comune tra le due celle. Se questo accade, significa che in quell'edge con molta probabilità esiste una porta. La stanza dovrebbe, perciò, essere separata esattamente da quell'edge. Il peso $w(e_{ij})$ di quell'edge viene quindi impostato ad 1, che equivale a chiudere artificialmente la porta. Una volta chiuse tutte le porte all'interno della stanza, viene rieseguito l'algoritmo DBSCAN sulle celle della stanza, permettendo di classificare ulteriormente la stanza under-segmentata, dividendola esattamente lungo gli edge individuati dalle

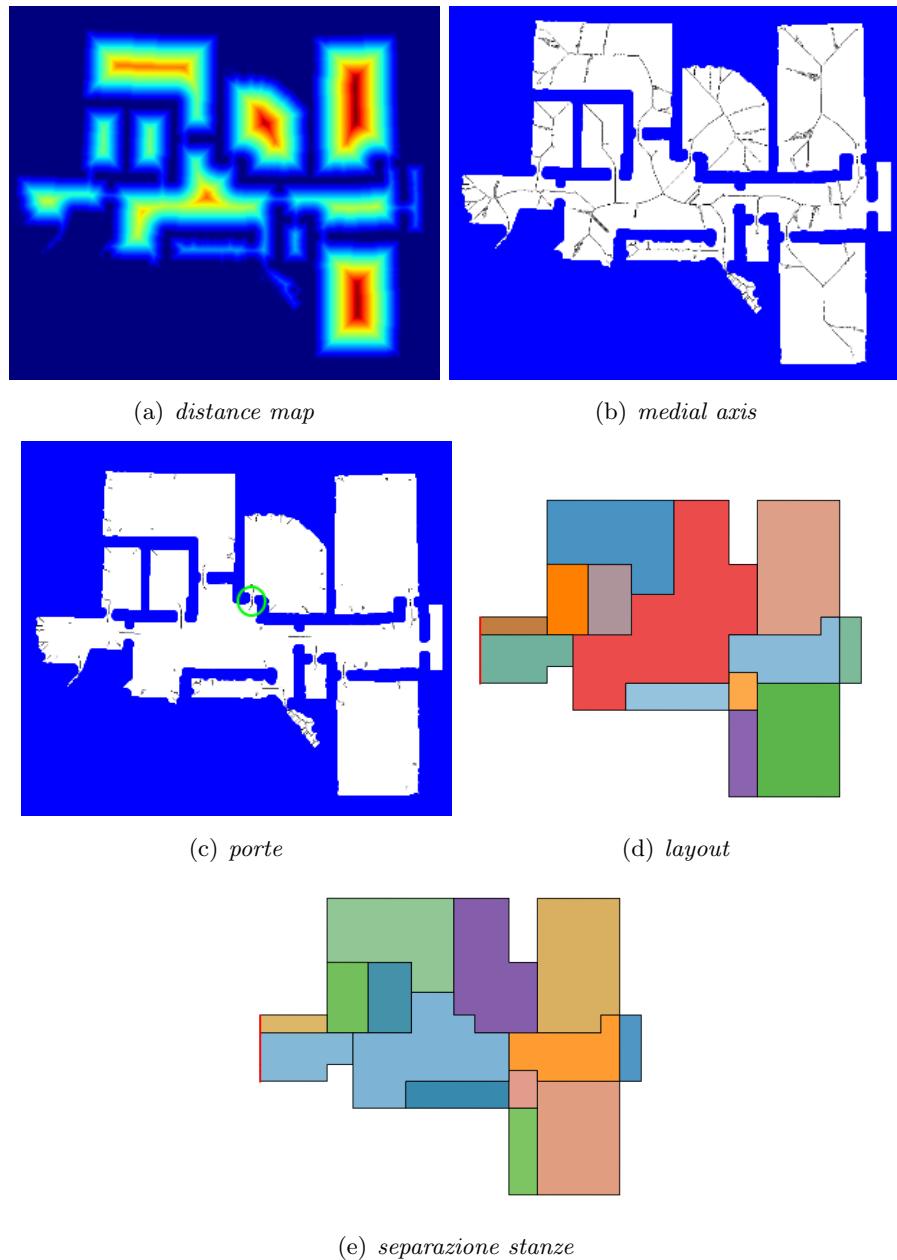


Figura 4.19: Esempio di ricerca delle porte. L'immagine (a) rappresenta la *distance transform* applicata alla mappa metrica. L'immagine (b) rappresenta il *medial axis*. L'immagine (c) rappresenta i punti critici in prossimità delle porte. La porta cerchiata è quella che il processo chiuderà al fine di riconoscere correttamente la divisione delle stanze. Le immagini (d) ed (e) rappresentano il *layout*, rispettivamente prima e dopo il processo che separa le stanze, nel quale l'unica porta chiusa corrisponde a quella cerchiata in verde nell'immagine (c).

porte. Finito questo processo vengono riassegnati, agli edge che indicano un passaggio stretto, i valori di peso originali. Il processo continua fino a quando le stanze under-segmentate non terminano. La Figura 4.19 (e) mostra un esempio, nel quale viene separata l'unica stanza under-segmentata (stanza rossa di Figura 4.19 (d))

Unione delle stanze

L'unione delle stanze che si propone in questo lavoro di tesi, analogamente al lavoro in [62], consiste nel controllare l'effettiva importanza del bordo in comune che separa due stanze adiacenti. Questo si basa sul fatto che un'incorrecta separazione (prodotta dal clustering delle celle) può essere facilmente riconosciuta dal momento che su quel bordo non vi è la proiezione di alcun segmento. Il processo deve in primis ricercare adiacenze tra stanze. Per far ciò vengono ricercate quelle coppie di stanze i cui contorni si toccano in almeno un bordo. Per ogni coppia individuata viene estratto l'insieme dei bordi comuni \mathcal{B} che viene successivamente valutato attraverso un'apposita funzione $Q_{split}(\mathcal{B})$ definita come segue:

$$Q_{split}(\mathcal{B}) = \frac{\sum_{i=1}^n w(e_i) * length(e_i))}{\sum_{i=1}^n length(e_i)} \quad (4.7)$$

dove $\mathcal{B} = \{e_1, \dots, e_n\}$ rappresenta l'insieme degli edge che separano due stanze adiacenti, $w(e)$ denota la copertura dell'edge e da parte dei segmenti, la quale segue la stessa definizione espressa in (4.5) e $length(e)$ indica la lunghezza dell'edge e . Questa funzione di qualità misura la solidità dell'insieme \mathcal{B} nel range $[0\dots1]$, dove 1 rappresenta un separazione lungo un muro perfettamente solido.

Due stanze vengono unite se il valore della funzione $Q_{split}(\mathcal{B})$ risulta essere più basso di una determinata soglia τ impostata in questo lavoro di tesi a $\tau = 0.2$.

4.3.2 Ricerca frontiere

Una frontiera è un bordo tra spazio esplorato e spazio inesplorato. Per riconoscerla è, in primo luogo, necessario analizzare la mappa metrica proveniente dalla fase di esplorazione. Ogni pixel della mappa viene classificato in base all'intensità di colore. I pixel dell'immagine possono essere visti come celle di una griglia e la loro intensità di colore come la probabilità che la cella sia stata vista o meno in fase di esplorazione.

Ogni cella può essere classificata nel seguente modo:

- **Libera:** una cella nella quale è stato riscontrato uno spazio libero e identificata dal colore bianco, visibile in Figura 4.20 (a).
- **Occupata:** una cella nella quale è stato riscontrato un ostacolo e caratterizzata dal colore nero, come in Figura 4.20 (a).
- **Sconosciuta:** una cella non ancora esplorata e caratterizzata dal colore grigio, come in Figura 4.20 (a).

L’algoritmo proposto in questa tesi nel modulo di ricerca delle frontiere è in grado di analizzare l’immagine e classificare ogni cella libera, adiacente ad una cella sconosciuta, come cella facente parte della frontiera. Questo permette di riconoscere tutti i pixel dell’immagine che fanno parte di una frontiera. In questa fase, però, non si è ancora in grado di distinguere frontiere diverse, intese come insiemi connessi di pixel, ma si ha a disposizione un insieme di celle della mappa metrica, i cui elementi non differiscono in alcun modo, se non semplicemente per la loro posizione nello spazio. Analogamente al clustering dei segmenti in cluster muri e al clustering delle celle in stanze del layout, anche in questa fase si è deciso di raggruppare i pixel in frontiere diverse con l’uso dell’algoritmo DBSCAN. Per far ciò, il dataset su cui viene eseguito DBSCAN rappresenta l’insieme delle celle classificate come frontiera F . La misura di affinità utilizzata in questo caso risulta essere una misura di distanza. Infatti, per ogni cella nell’insieme F viene calcolata la distanza rispetto alle altre, $\forall c_1, c_2 \in F : DistanzaEuclidea(c_1, c_2)$ dove $DistanzaEuclidea(c_1, c_2) = \sqrt{(x_{c1} - x_{c2})^2 + (y_{c1} - y_{c2})^2}$.

Grazie a questa misura DBSCAN è in grado di raggruppare celle molto vicine. Dato che le celle in F sono sparse nello spazio, il numero di cluster e la loro dimensione dipendono dalla densità di celle che si trovano in una determinata regione di spazio. Per eliminare cluster poco significativi, i quali rappresentano per lo più rumore, è stato scelto come in [99] di considerare solamente cluster di dimensione maggiore a una determinata soglia, impostata in questo lavoro di tesi a 100 pixel. La Figura 4.20 mostra due esempi del processo di riconoscimento delle frontiere.

4.3.3 Riconoscimento stanze parziali

Le frontiere trovate nella fase precedente sono necessarie al fine di poter riconoscere le stanze parziali del layout. Questo perché, ragionevolmente, se esiste una frontiera all’interno di una stanza, questa è parziale. Essendo una stanza del layout un oggetto composto da un numero finito di celle, dunque, perché una stanza sia parziale deve essere formata da almeno una cella che contiene al suo interno un certo numero di pixel corrispondenti



(a) *mappa in scala di grigi*

(b) *frontiere*



(c) *mappa in scala di grigi*

(d) *frontiere*

Figura 4.20: (a) e (c) mostrano le mappe originali, ottenute dalla fase di esplorazione. (b) e (d) mostrano il processo di riconoscimento delle frontiere.



Figura 4.21: Processo di riconoscimento delle stanze parziali.

ad una frontiera. Per far ciò, quindi, si mappano i pixel classificati come frontiera all'interno delle celle osservate $co \in \mathcal{C}_o$ che costituiscono una stanza del layout. Ciò permette di riconoscere tutte le celle parziali $cp \in \mathcal{C}_p \subseteq \mathcal{C}_o$, dove \mathcal{C}_p rappresenta l'insieme delle celle in \mathcal{C}_o , nelle quali è presente una frontiera. Perché un stanza s del layout sia parziale, si deve verificare che abbia all'interno almeno una cella parziale.

L'intero processo permette di identificare tutte le stanze parziali del layout, dato che ognuna di esse possiede almeno una cella parziale. La Figura 4.21 mostra un esempio del processo, nel quale partendo dall'immagine della mappa metrica, Figura 4.21 (a), vengono riconosciute le frontiere, Figura 4.21 (b), che vengono in seguito utilizzate per ricercare le stanze parziali (stanze in grigio) dell'immagine in Figura 4.21 (c).

4.3.4 Azioni geometriche

A questo punto, una volta trovate le stanze parziali del layout, è possibile fornire un'ipotesi di completamento per ognuna di esse. Questo processo viene svolto dai moduli che eseguono azioni geometriche che permettono di elaborare una prima predizione sulle stanze parziali, basandosi, oltre che sulle informazioni derivanti dalla frontiera, anche sulla geometria dell'ambiente.

Le azioni geometriche proposte in questo lavoro di tesi sono due.

Azione geometrica 1

La prima azione geometrica proposta permette di unire celle esterne $c \in \mathcal{C}_e$ ad una stanza parziale $s \in \mathcal{S}$. Questa azione infatti unisce alla stanze parziali $s \in \mathcal{S}$ tutte le celle esterne $c \in \mathcal{C}_e$ adiacenti ad una cella interna alla stanza parziale s nella quale è stata riscontrata una frontiera e che possiedono pixel facenti parte della frontiera stessa.

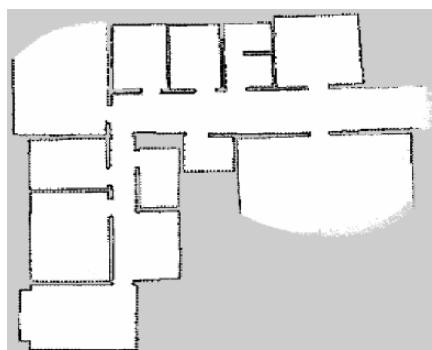
L'azione geometrica 1 è resa necessaria dal fatto che in fase di riconoscimento delle celle interne al layout, alcune vengono considerate esterne in quanto non soddisfano il criterio di minima copertura¹. Una volta ottenute le frontiere, però, si è in grado di riconoscere quali siano le celle esterne, nelle quali è presente del rumore inevitabile, e quali invece siano quelle che, con alta probabilità, non sono state completamente esplorate.

Questo permette di aggiungere a stanze parziali $s \in \mathcal{S}$ una serie di celle effettivamente viste in precedenza, fornendo una prima approssimazione del layout predetto. La Figura 4.22 (c) mostra un esempio.

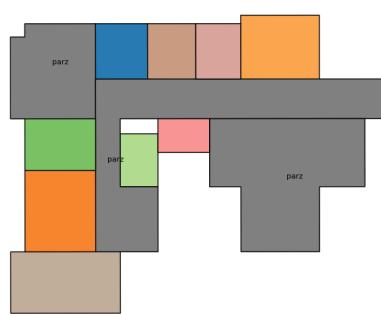
Azione geometrica 2

La seconda azione geometrica proposta permette di aggiungere celle $c \in \mathcal{C}_e$ ad una stanza parziale $s \in \mathcal{S}$, solo se queste sono supportate da pareti di altre stanze. Questa azione permette, infatti, di fare previsioni a corto raggio, andando ad aggiungere solo quelle celle $c \in \mathcal{C}_e$ adiacenti alle celle che rappresentano una frontiera $cp \in \mathcal{C}_p$ di una stanza parziale $s \in \mathcal{S}$. Queste, se unite alle celle della stanza $s \in \mathcal{S}$, aggiungono una parete che è allineata ad un muro già noto. La Figura 4.22 (d) mostra un esempio.

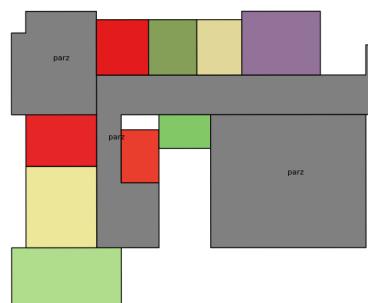
¹Criterio di minima copertura: ogni cella viene considerata interna o esterna in base alla percentuale in comune che la cella stessa possiede con il contorno dell'intera mappa. Come già espresso in Sezione 4.2.4 la percentuale è settata al 20% dell'area della cella stessa.



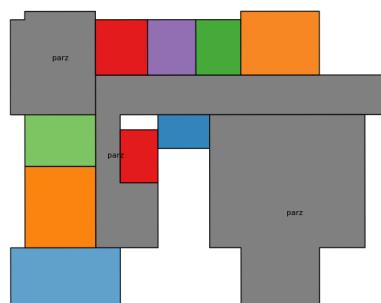
(a) mappa



(b) layout originale



(c) azione geometrica 1



(d) azione geometrica 2

Figura 4.22: Risultato delle azioni geometriche applicate al layout originario.

4.3.5 Azione geometrica complessa

Le azioni geometriche semplici hanno la sola capacità di aggiungere ad una stanza parziale $s \in \mathcal{S}$ celle esterne in \mathcal{C}_e , confinanti con la frontiera della stanza stessa, rappresentata dall'insieme delle celle \mathcal{C}_p . Le azioni semplici non sono tuttavia in grado di fornire una misura globale del valore che le suddette celle apportano al layout.

Partendo da questo presupposto, è stato necessario creare una nuova azione che fosse in grado di scegliere la migliore tra le diverse combinazioni di celle in \mathcal{C}_e potenzialmente unibili alla stanza parziale $s \in \mathcal{S}$.

L'azione complessa proposta prevede 4 fasi d'esecuzione:

- **Azione geometrica 1.** La prima fase permette di aggiungere celle esterne $c \in \mathcal{C}_e$ alla stanza parziale s presa in considerazione. In particolare aggiunge tutte quelle celle $c \in \mathcal{C}_e$ confinanti con le celle parziali $cp \in \mathcal{C}_p$ di una stanza s , quando queste possiedono all'interno pixel facenti parte della stessa frontiera.
- **Celle confinanti C .** La seconda fase colleziona tutte le celle esterne $c \in \mathcal{C}_e$ confinanti con le celle $cp \in \mathcal{C}_p$ che rappresentano la frontiera di una stanza, fino ad un determinato livello di profondità. La profondità è dettata dal minimo numero di salti (misurati in celle confinanti) necessari per raggiungere una cella esterna $c \in \mathcal{C}_e$, partendo dalla stanza parziale. Le celle colorate in verde nelle immagini identificate dal numero 1) di Figura 4.23 rappresentano l'insieme C con profondità 2.
- **Calcolo delle disposizioni di celle \mathcal{D} .** Ottenuto l'insieme delle celle confinanti C , ne vengono calcolate tutte le disposizioni di k elementi con $k = 1 \dots |C|$, fino ad ottenere per $k = |C|$ le permutazioni dell'insieme C . Ogni elemento in \mathcal{D} rappresenta un ordinamento diverso di celle che possono essere aggiunte ad una stanza parziale. La Figura 4.24 ne mostra un esempio.
- **Valutazione del layout.** Dopo aver ottenuto l'insieme delle disposizioni \mathcal{D} , per ogni disposizione $d \in \mathcal{D}$ si calcola il valore della stanza parziale s' ottenuta da $\mathcal{C}_o \cup d$ attraverso una score function $f(s')$.

Celle confinanti C

La creazione dell'insieme C serve a fornire una prima selezione delle celle esterne che possono essere aggiunte ad una stanza parziale. Le celle colorate

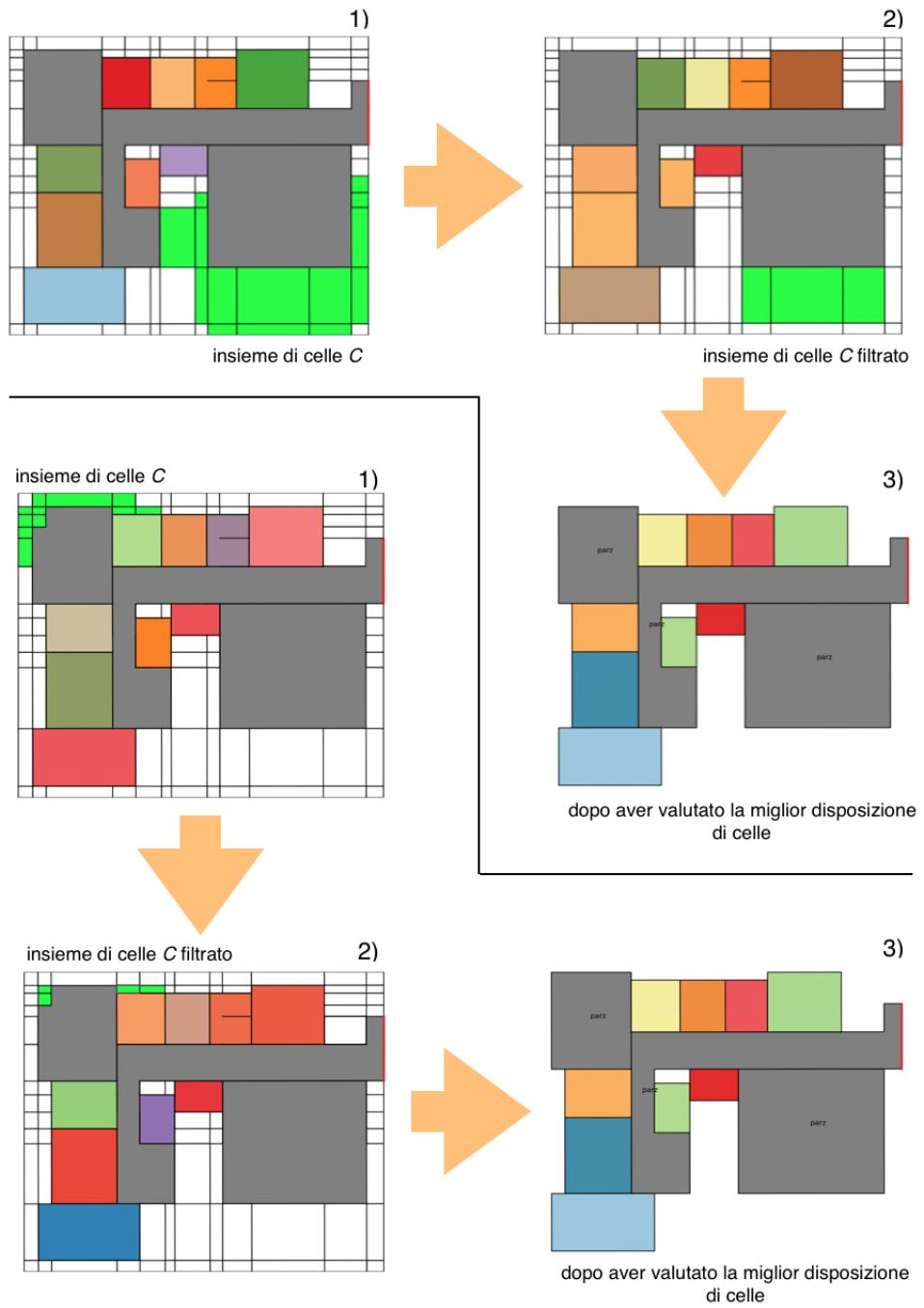


Figura 4.23: Processo dell'azione complessa. L'immagine identificata dal numero 1) mostra le celle in C (in verde chiaro). L'insieme di celle in C viene successivamente filtrato, eliminando le celle che se unite ad un stanza parziale si troverebbero oltre una parete già (anche parzialmente) osservata, come mostrano le immagini identificate dal numero 2). Le immagini identificate al numero 3) rappresentano il layout dopo aver valutato la miglior disposizione d.

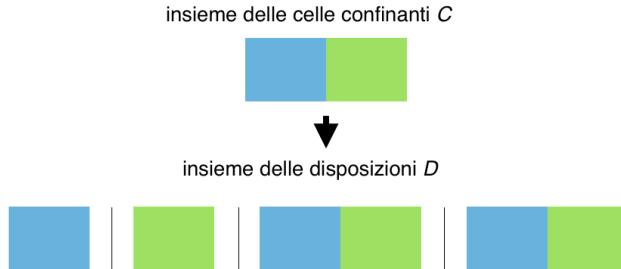


Figura 4.24: Esempio di costruzione dell’insieme delle disposizioni \mathcal{D} ottenuto dall’insieme C . Colori diversi si riferiscono a celle diverse. Nel caso in questione l’insieme delle celle confinanti con le celle di frontiera è composto da due celle diverse (verde e azzurro).

in verde chiaro nelle immagini identificate dal numero 1) di Figura 4.23 forniscono un esempio di come sia costituito l’insieme C . Una cella esterna $c \in \mathcal{C}_e$, perché possa far parte dell’insieme delle celle confinanti C , deve soddisfare alcuni vincoli. In particolare non devono far parte dell’insieme C tutte le celle che, se aggiunte alla stanza $s \in \mathcal{S}$, si troverebbero oltre una parete già (anche parzialmente) osservata e che quindi procurerebbero stanze con all’interno un muro noto. Non vengono considerate come celle facenti parte di C anche quelle che portano la stanza a confinare con il bordo della mappa. Questo poiché introdurrebbero informazioni sull’impossibilità di completamento della stanza, determinando conseguentemente la necessaria esplorazione dell’intera area da parte del robot. Le immagini identificate dal numero 2) in Figura 4.23 mostrano un esempio di come l’insieme C viene modificato se si filtrano le celle che non soddisfano i vincoli.

Calcolo delle disposizioni

Un passaggio fondamentale perché le celle possano essere aggiunte ad una stanza parziale consiste nel ricercare tutti i possibili ordinamenti di celle dell’insieme C . Questo passaggio è necessario al fine di raggiungere l’obiettivo finale, ossia selezionare la combinazione di celle che apporta maggior contributo al layout, basandosi sul valore di una score function. Ottenuto l’insieme delle celle confinanti C , è necessario calcolarne tutte le disposizioni di k elementi, con $k = 1 \dots |C|$. Questo processo fornisce una collezione di insiemi $\mathcal{D} := \{d_1, d_2, \dots, d_m\}$ ² nella quale ogni elemento rappresenta una particolare disposizione di celle. L’assetto delle celle in d rappresenta l’ordine

²Il pedice m indica la cardinalità dell’insieme \mathcal{D} . $m = |\mathcal{D}| = \sum_{k=1}^{|C|} D_{n,k}$, dove $n = |C|$ e $D_{n,k} = n(n-1)(n-2)\dots(n-k+1) = \frac{n!}{(n-k)!}$.

con il quale queste vengono aggregate alla stanza parziale $s \in \mathcal{S}$ considerata. Molte di queste disposizioni $d \in \mathcal{D}$ dunque rappresentano differenti vie per raggiungere lo stesso risultato. L'algoritmo proposto in Algoritmo 1 permette di eliminare, ogni qualvolta viene rilevata una disposizione legale di elementi, tutte le disposizioni equivalenti successive. Una disposizione $d \in \mathcal{D}$ viene considerata legale se l'aggiunta di ogni cella $c \in d$ alla stanza parziale $s \in \mathcal{S}$, presa secondo l'ordine di apparizione in d , è possibile. Non si deve tentare, cioè, di aggiungere una cella non confinante con la stanza o con le celle aggiunte fino a quel momento. Nel caso in cui una cella $c \in d$ non fosse confinante con la stanza parziale s , l'intera disposizione $d \in \mathcal{D}$ non sarebbe legale.

Valutazione del layout

Data una stanza $s \in \mathcal{S}$ composta dalle celle osservate \mathcal{C}_o , si valuta la disposizione di celle in $\mathcal{D} = \{d_i | \forall i = 1 \dots m\}$ tale per cui la stanza s' ottenuta da $\mathcal{C}_o \cup d_i$ massimizza una score function.

$$\arg \max_{d \in \mathcal{D}} f(s' = \mathcal{C}_o \cup d) \quad (4.8)$$

La score function fornisce un guadagno, quando una disposizione $d \in \mathcal{D}$ determina un layout con una migliore rappresentazione geometrica di un ambiente e penalità ogni qualvolta si presentino violazioni della struttura geometrica che in seguito andremo a chiarire.

Più precisamente la score function è definita come segue:

$$f(s') := k_w * extendedWeight(s') - (k_c * convex(s') + k_e * nExtended(s')) \quad (4.9)$$

k_w, k_c, k_e sono costanti che specificano l'importanza di ogni termine.

Di seguito vengono elencati i vari componenti che costituiscono l'equazione (4.9).

extendedWeight(s'). Il primo score proposto fornisce una misura globale. Ad esso viene attribuito un valore positivo relativo alla somma del valore di ogni singolo segmento esteso, sul quale giacciono le pareti di una stanza s' ottenuta dalla disposizione $d \in \mathcal{D}$. I segmenti estesi vengono rappresentati con linee caratterizzate da un punto iniziale ed uno finale che risiedono sui bordi esterni dell'immagine. Il valore w_g di un segmento esteso g è definito come segue:

$$w_g = \frac{cov(g)}{length(g)} \quad (4.10)$$

dove $cov(g)$ denota la porzione del segmento esteso coperto dalle proiezioni dei segmenti (rappresentanti i muri) su di essa e $length(g)$ la sua lunghezza. Ogni disposizione legale aggiunge un determinato numero di celle alla stanza parziale $s \in \mathcal{S}$ considerata. L'aggiunta di queste celle produce una nuova stanza s' , che introdurrà alcune pareti, di cui solo alcune in comune con la stanza originale s . Lo score dunque favorisce disposizioni che generano pareti che giacciono su rette rappresentative molto importanti, sulle quali giacciono pareti di altre stanze del layout. Lo score viene definito come segue:

$$extendedWeight(s') := \sum_{j=0}^{|E|} w_{g_j} \quad (4.11)$$

dove g_j rappresenta il j -esimo segmento esteso su cui giace una parete della stanza parziale s' e $|E|$ rappresenta il numero di pareti della stanza s' .

convex(s'). Basandosi sull'osservazione fatta su diverse mappe, complete e non, si è notato un pattern frequente, nel quale gli ambienti il più delle volte assumono forme convesse. La prima penalità proposta fornisce dunque una misura di preferenza verso le stanze convesse rispetto a quelle concave.

$$convex(s') := \frac{Area(Hull(s')) - Area(s')}{Area(s')} \quad (4.12)$$

dove $Area(s')$ fornisce l'area di s' e $Hull(s')$ fornisce il convex hull³ della stanza $s' = s \cup d$ ottenuta dalla disposizione $d \in \mathcal{D}$.

nExtended(s'). La seconda penalità proposta viene attribuita a quelle configurazioni del layout, nelle quali, aggiungendo tutte le celle di una disposizione d alla stanza parziale s , si introducono ulteriori pareti. In particolare, questa penalità viene pesata in base al numero di segmenti estesi introdotti. Questo favorisce permutazioni che non introducono pareti sconosciute alla stanza.

$$nExtended(s') := |E| \quad (4.13)$$

dove con $|E|$ si intende la cardinalità dell'insieme delle pareti della stanza s' ottenuta con la disposizione $d \in \mathcal{D}$.

Massimizzare il valore della score function $f(s')$ significa, dunque, andare a scegliere la migliore disposizione di celle $d \in \mathcal{D}$ che estendono s . Vale a

³Il convex hull di una forma poligonale A , rappresenta il più piccolo poligono B , che contenga il poligono A al suo interno e che allo stesso tempo abbia una forma convessa.

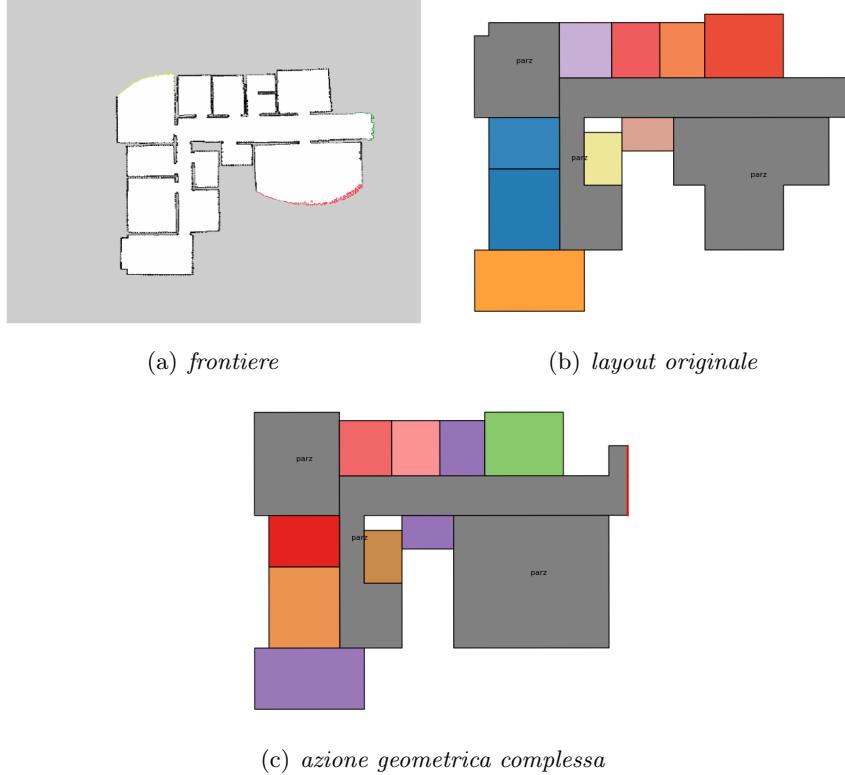


Figura 4.25: Esempio di azione geometrica complessa. L'immagine (a) mostra le frontiere. L'immagine (b) il layout, dopo aver ricercato le stanze parziali. L'immagine (c) mostra il risultato ottenuto dall'azione geometrica complessa.

dire, prediligere la disposizione che produce una stanza s' il più rettangolare possibile, che introduca il minor numero di pareti e che, tra tutte, scelga quelle maggiormente supportate da altre stanze; ossia quelle pareti che giacciono sullo stesso segmento esteso su cui risiedono pareti di altre stanze. La Figura 4.25 mostra un esempio di esecuzione dell'azione geometrica complessa.

Lo pseudocodice dell'algoritmo che implementa le azioni geometriche complesse viene mostrato in Algoritmo 1.

Complessità dell'algoritmo azione geometrica complessa

In questa sezione si mostra la complessità della soluzione proposta per l'azione geometrica complessa. Siano:

- \mathcal{S} l'insieme delle stanze parziali e Γ la cardinalità di \mathcal{S} ($\Gamma = |\mathcal{S}|$).

Algorithm 1 Azione complessa

Require: insieme delle stanze parziali \mathcal{S}
azione_geometrica1(layout)
for all $s \in \mathcal{S}$ **do**
 $C \leftarrow getCelleConfinanti(s)$
 $\mathcal{D} \leftarrow calcolaDisposizioni(C)$
 $d_{ini} \leftarrow \emptyset$
 $initial_score \leftarrow scoreFunction(d_{ini})$
 $SCORE \leftarrow \emptyset$
 $disposizioni_legali \leftarrow \emptyset$
 for all $d \in \mathcal{D}$ **do**
 $s' \leftarrow aggiungiCelleAStanza(s, d)$
 if isLegal(s') **then**
 elimina da \mathcal{D} disposizioni simili a d
 $f(d) \leftarrow scoreFunction(d)$
 $SCORE \leftarrow f(d)$
 $disposizioni_legali \leftarrow d$
 else
 elimina d da \mathcal{D} poiché non legale
 end if
 end for
 $max_score \leftarrow massimo_elemento_in(SCORE)$
 $miglior_disposizione \leftarrow ottieni_disposizione(max_score)$
 if $max_score == initial_score$ **then**
 //la configurazione iniziale è la migliore
 $miglior_disposizione \leftarrow d_{ini}$
 end if
 $s \leftarrow aggiungiCelleAStanza(s, miglior_disposizione)$
end for

- C l'insieme delle celle confinanti associato alla stanza parziale $s \in \mathcal{S}$. La dimensione dell'insieme C è dipendente dal livello di profondità che si raggiunge per la generazione delle celle candidate ad essere aggiunte ad una stanza s .
- \mathcal{H} l'insieme di tutte le possibili combinazioni ottenute dagli elementi in C .
- $C_{n,k} = \frac{n!}{(n-k)!k!} = \binom{n}{k}$ la cardinalità di un insieme di k elementi estratti da un insieme di n oggetti.
- $h = |\mathcal{H}| = \sum_{k=1}^n C_{n,k}$, con $n = |C|$.

Essendo $h < 2^n$, la complessità del problema è $O(\Gamma * 2^n)$, dove n indica il numero di candidati in C per ogni stanza $s \in \mathcal{S}$. Lo studio della soluzione del problema ha portato ad una complessità esponenziale di base 2. Come facilmente intuibile, minore è n (il numero di elementi in C) minore è il tempo necessario al fine di compiere una predizione. Per valori elevati di n , infatti, l'azione complessa rappresenta una fase dell'intero processo particolarmente onerosa.

La trattazione di questo capitolo ha descritto nel suo complesso l'intero sistema dal punto di vista del funzionamento e della relazione tra i vari elementi e processi che lo compongono. Partendo dall'architettura generale si è poi scesi più nel dettaglio spiegando passo per passo come è stata affrontata la predizione del layout dal punto di vista del lavoro oggetto in questa tesi.

Capitolo 5

Architettura del sistema

In questo capitolo viene descritta l'architettura del modulo che permette, a partire da una mappa metrica relativa ad un ambiente non completamente esplorato, di fornire un'ipotesi predittiva sulle porzioni sconosciute dell'ambiente stesso, basandosi sulle informazioni raccolte all'interno della porzione esplorata.

La trattazione di questo capitolo illustra gli aspetti tecnologici ed implementativi affrontati in questo lavoro di tesi. Inizialmente si mostrano le tecnologie scelte per la realizzazione dei moduli descritti nel Capitolo 4, i dati in input e gli strumenti utilizzati per ottenere un output. In seguito si mostra nel dettaglio l'architettura del sistema realizzato, focalizzando maggiormente l'attenzione sulle classi necessarie al modulo di predizione e su come queste interagiscano tra di loro. Vengono inoltre descritte alcune funzionalità significative dei componenti che costituiscono il modulo di predizione.

5.1 La mappa metrica ed il layout

La realizzazione pratica del modulo di predizione prevede di:

- Definire un linguaggio di programmazione da usare nell'implementazione dei moduli descritti nel Capitolo 4. In particolare ogni modulo che compone il sistema è stato implementato in **Python 2.7**. Questo rappresenta un linguaggio ad alto livello orientato agli oggetti ed è proprio per questo motivo che è stato possibile progettare ed implementare il codice sotto forma di moduli e classi che interagiscono tra di loro.

- Avere a disposizione un metodo attraverso cui è possibile ottenere una mappa metrica parziale, che rappresenta l'input primario dell'intero sistema. Ciò non riguarda direttamente l'implementazione del modulo che fornisce una predizione del layout, ma è comunque necessario al fine della verifica del corretto funzionamento dell'intero sistema, nonché del modulo di predizione.
- Avere a disposizione un metodo attraverso cui è possibile ottenere una ricostruzione del layout. Questo metodo risulta essere necessario al fine di individuare le celle e le stanze che compongono il layout e che verranno usate per fornire un'ipotesi di completamento per le stanze parziali. Il layout rappresenta la base di conoscenza da cui è iniziata l'implementazione del progetto presentato in questa tesi, in particolare modificando ed aggiungendo componenti rispetto al sistema presentato in [13].
- Realizzare un apposito modulo in grado di visualizzare sotto forma di un'immagine i dati ottenuti dai processi sviluppati in questo lavoro di tesi e utile a fornire *feedback* visivi immediati sui vari step intermedi dell'intero sistema.

L'intero sistema è stato descritto dal punto di vista teorico nel Capitolo 4. Di seguito si fornisce un'illustrazione sull'implementazione dei moduli mappa metrica e layout.

5.1.1 La mappa metrica

L'obiettivo primario del modulo mappa metrica è quello di fornire una rappresentazione della mappa metrica dell'ambiente sotto forma di matrice, in cui le celle contengono i valori RGB inclusi nell'intervallo [0,255]. La mappa metrica si ottiene dai dati raccolti dai sensori di un robot mobile. In particolare, per ottenere una mappa metrica, come descritta in Sezione 4.1, il modulo implementa un'esplorazione di un ambiente indoor attraverso cui il robot mobile acquisisce i dati provenienti dai sensori al fine di elaborarli e generare così la mappa metrica.

La totalità delle mappe utilizzate in questo lavoro di tesi rappresenta il risultato di un'esplorazione avvenuta in precedenza e quindi il sistema proposto non lavora al momento online, integrato come parte di un sistema di esplorazione. L'esplorazione di un ambiente avviene mediante l'uso di un simulatore, *Stage* [95]. In Stage il robot viene posto in un ambiente bidimensionale che viene simulato a partire da un'immagine, nella quale sono

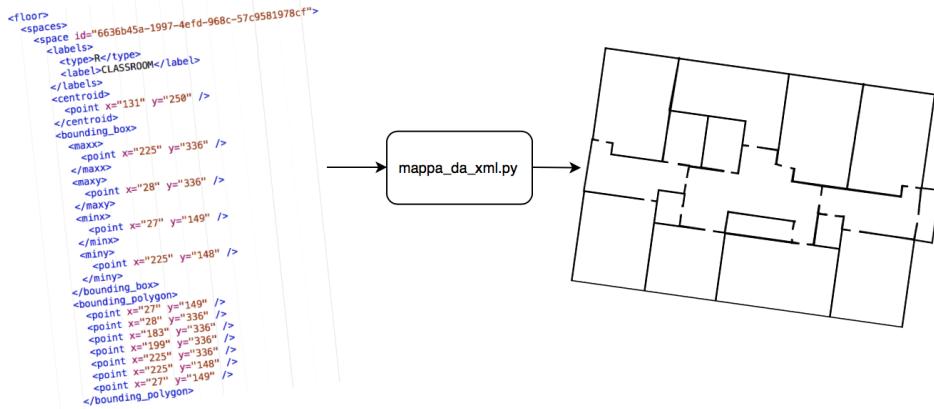


Figura 5.1: Lo script Python `mappa_da_xml.py`, riceve in input un file XML che descrive l’ambiente nel quale si vuole eseguire la simulazione. L’output dello script rappresenta un’immagine in cui le parti bianche rappresentano lo spazio libero e quelle nere lo spazio occupato da un ostacolo.

rappresentati lo spazio libero e gli ostacoli, attraverso i colori bianco e nero. L’immagine che rappresenta l’ambiente viene ottenuta grazie allo script Python `mappa_da_xml.py`. Questo permette di ottenere un’immagine partendo da un file XML (*eXtensible Markup Language*) che descrive l’ambiente. La Figura 5.1 mostra un esempio in cui è possibile distinguere input ed output dello script Python `mappa_da_xml.py`.

Una volta ottenuta l’immagine relativa all’ambiente descritto dal file XML, si esegue il file Python `crea_world.py`, che prevede in ingresso la mappa appena creata dall’XML e genera un file in formato *world*, che verrà poi aperto da Stage. All’interno del file viene inoltre settata la posizione in cui è inizialmente posto il robot.

L’implementazione di Stage utilizzata per ottenere le mappe metriche usate in questo lavoro di tesi è integrata in *ROS (Robot Operating System)* [75]. ROS è un framework che fornisce librerie e strumenti per creare applicazioni robotiche. Esso fornisce astrazione dell’hardware, driver per il controllo dei dispositivi, strumenti di visualizzazione, comunicazione a scambio di messaggi tra processi (message passing), gestione dei pacchetti e molto altro, ed è rilasciato sotto una licenza *open source*.

In questo lavoro di tesi, per creare la mappa metrica, viene utilizzato il modulo ROS, *SLAM GMapping* incluso nel package `gmapping` [74], se-

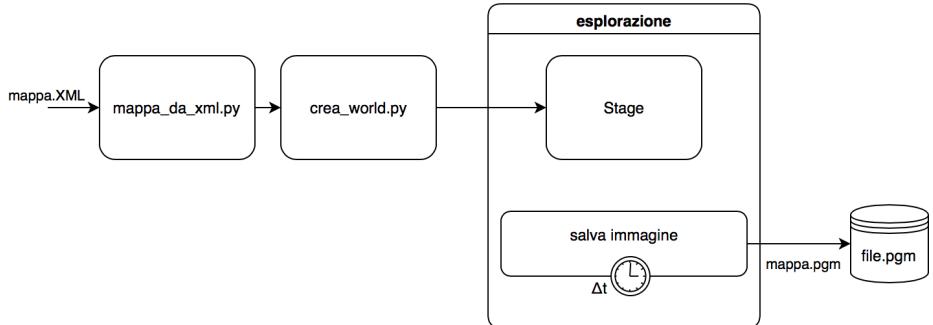


Figura 5.2: Processo con il quale, partendo da un XML che descrive un ambiente, è possibile ottenere un insieme di mappe metriche parziali in formato pgm relative all’ambiente, rappresentato ad intervalli temporali differenti.

guendo un approccio di esplorazione Nearest frontier, discusso in [40]. Lo SLAM GMmapping integra in modo incrementale le scansioni laser acquisite durante l’esplorazione, allo scopo di costruire una occupancy grid map dell’ambiente e al contempo stabilire la posizione del robot all’interno di essa. La costruzione della mappa richiede che l’ambiente venga esplorato da parte di un robot dotato di sensori. A tale scopo è stata utilizzata la tecnica di esplorazione Nearest Frontier, nella quale il robot calcola le distanze che lo separano da tutte le frontiere individuate e sceglie quella più vicina alla sua posizione corrente come destinazione successiva. La navigazione verso tale frontiera viene eseguita pianificando un percorso, garantito essere libero da ostacoli.

Durante il processo d’esplorazione è possibile eseguire lo script Python `salva_mappa.py`. Questo permette di salvare l’immagine dell’ambiente esplorato ad intervalli temporali di Δt secondi, modificabile a piacere e settato di default a $\Delta t = 5$ s. Le immagini dell’ambiente estratte ogni Δt vengono salvate nel formato *pgm* (*Portable Graymap Bitmap*), in cui le informazioni interne fanno riferimento alle diverse sfumature di nero, grigio e bianco della mappa metrica.

L’intero processo descritto fino ad ora viene mostrato in Figura 5.2, nella quale partendo dal file XML, che rappresenta l’ambiente in cui si vuole simulare l’esplorazione, si arriva ad avere un insieme di file che rappresenta l’ambiente esplorato ad intervalli di tempo differenti.

Prima di poter effettivamente usare i file generati come mappe metri-

che parziali in input al sistema, è necessario che ogni file venga convertito nel formato *png* (*Portable Network Graphic*) ed adattato ad una dimensione $A \times B$. In essa A risulta essere fissa e settata in questo lavoro di tesi al valore 1000, mentre B viene proporzionata ogni volta in base alla grandezza originale dell’immagine, in modo da non perderne le proporzioni. La dimensione dell’immagine viene modificata per far fronte al problema riscontrato in cui le dimensioni di alcune immagini risultavano essere molto piccole rispetto ad altre. Il compito di adattare la dimensione dell’immagine e di convertire il formato in *png* viene eseguito dallo script Python `converti_parziali.py`.

5.1.2 Il layout

In questa sezione si vuole descrivere l’architettura generale del modulo layout, descritto a livello teorico in Sezione 4.2. L’intero processo, che parte dalla lettura di una mappa metrica parziale e che termina restituendo una lista di oggetti che rappresenta il layout, come ad esempio la lista delle stanze, può essere visto come una serie di funzioni eseguite in sequenza. Il modulo è stato progettato al fine di essere utilizzato offline, ciò significa che il momento di esecuzione del modulo è totalmente indipendente dal momento d’esplorazione dell’ambiente. Il modulo layout è composto principalmente da cinque componenti, precedentemente descritti in Sezione 4.2. In questa sezione non si vuole dare una spiegazione sul funzionamento dei moduli, ma si vuole fornire una documentazione sui dettagli implementativi del software sviluppato. Si evidenzia inoltre come il lavoro relativo alla ricostruzione del layout in [13] sia stato modificato in alcuni dei suoi elementi ed adattato al sistema sviluppato in questa tesi, al fine di svolgere correttamente il compito di ricostruzione di un layout parziale e successivamente di predizione.

La Figura 5.3 mostra il flusso d’esecuzione dell’intero modulo layout, in cui, per ogni fase principale, vengono indicati i passi più significativi. Alcune di queste fasi, identificate dal colore arancione, rappresentano le attività sviluppate o pesantemente modificate in questo lavoro di tesi, che permettono una migliore gestione delle mappe parziali rispetto a [13]. La Figura 5.3 mostra inoltre il design con il quale è stato progettato il lavoro in questa tesi, suddividendo logicamente le operazioni in cinque fasi diverse. Le fasi del modulo sono:

- *Riconoscimento segmenti*: permette di estrarre una lista di segmenti corrispondenti ai muri della mappa metrica.
- *Segmenti estesi*: permette di creare una lista di segmenti estesi, raggruppando in cluster diversi i segmenti estratti nella fase precedente.

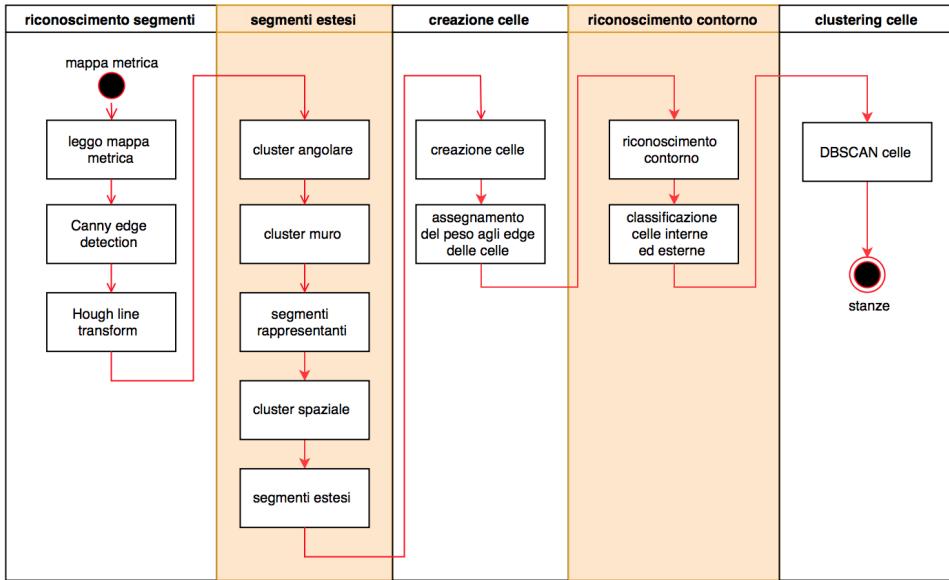


Figura 5.3: Diagramma di attività ad alto livello del modulo layout. Ogni colonna corrisponde alla divisione logica con la quale il modulo è stato diviso. Le colonne arancioni corrispondono ad attività modificate o aggiunte al lavoro [13], per avere una migliore gestione delle mappe parziali in ingresso all'intero sistema.

- *Creazione celle*: permette di creare una lista di celle derivanti dalle intersezioni tra i segmenti estesi.
- *Riconoscimento contorno*: permette di riconoscere le mura perimetrali della mappa metrica e di classificare le celle estratte nella fase precedente come interne o esterne all'edificio.
- *Clustering celle*: permette di creare una lista di stanze, raggruppando le celle interne identificate in precedenza.

Di seguito si elencano gli input e gli output di ogni componente del modulo layout dal punto di vista implementativo.

Riconoscimento segmenti

Input: mappa metrica parziale `mappaMetrica`. Parametri algoritmo di Canny edge detection: `minVal` e `maxVal`. Parametri algoritmo Hough line transform: `rho`, `theta`, `thresholdHough`, `minLineLength` e `maxLineGap`.

Output: lista `walls` contenente oggetti di tipo `Segmento`.

Questo modulo, precedentemente descritto in Sezione 4.2.1, prevede in ingresso la mappa metrica parziale, su cui si esegue l'algoritmo Canny edge

detection offerto dalla libreria Python **OpenCV** [66], invocando la funzione `cv2.Canny`. Questa funzione associa un valore ad ogni cella della mappa metrica. Successivamente confronta questo valore con i parametri `minVal` e `maxVal`, impostati ai valori `minVal=90` e `maxVal = 100`. Se il valore risulta essere inferiore a `minVal`, viene scartato. Se il valore risulta essere superiore a `maxVal` viene selezionato come parte del bordo. Se il valore risulta essere nell'intervallo tra `minVal` e `maxVal` esso viene accettato solamente se adiacente ad un valore precedentemente selezionato come parte del bordo.

Sul risultato della funzione `cv2.Canny` viene in seguito invocata la funzione `cv2.HoughLinesP` offerta dalla medesima libreria. I parametri `rho=1`, `theta=π/180`, `thresholdHough=20`, `minLineLength=7` e `maxLineGap=3` vengono forniti in ingresso al componente che riconosce i segmenti, in modo da poter essere regolati dall'esterno. Il risultato della funzione `cv2.HoughLinesP` viene infine utilizzato per ottenere una lista `walls` di oggetti di tipo `Segmento`, che rappresentano i segmenti con i quali si approssimano le mura della mappa metrica.

Segmenti estesi

Input: lista `walls`. Parametri di mean shift: `h` e `minOffset`. Parametri DBSCAN cluster muri: `eps` e `min_samples`. Parametro per il cluster spaziale: `minLateralSeparation`.

Output: lista `extended_segments`. Ogni elemento della lista viene rappresentato con un oggetto di tipo `Extended_segment`¹.

Questo modulo, precedentemente descritto in Sezione 4.2.2, ha come obiettivo quello di raggruppare gli elementi della lista `walls` in diversi cluster, che verranno in seguito rappresentati attraverso i segmenti estesi, utilizzando oggetti di tipo `Extended_segment`.

A tale scopo vengono invocate in cascata una serie di funzioni in grado di classificare i segmenti della lista `walls`.

La prima operazione raggruppa i segmenti in base alla loro direzione ottenendo così i cluster angolari. Ciò avviene invocando la funzione `mean_shift` i cui i parametri `h=0.023` e `minOffset=0.00001` rappresentano rispettivamente una costante arbitraria e la soglia che gestisce la condizione di terminazione dell'algoritmo `mean_shift`.

Ogni cluster angolare viene ulteriormente suddiviso ottenendo i cluster muri attraverso l'algoritmo DBSCAN, descritto in Sezione 4.2.2 e offerto dalla libreria Python **Scikit-learn** [78]. I parametri `eps=7` e `min_samples=2`

¹Gli oggetti `Extended_segment` si riferiscono ai segmenti estesi espressi in Sezione 4.2.2.

rappresentano rispettivamente il raggio di vicinanza e il minimo numero di punti nell' ϵ -vicinato.

Dai cluster muri vengono selezionati dei segmenti rappresentanti invocando la funzione `get_rappresentanti`, che vengono poi uniti in cluster spaziali invocando la funzione `spatialClustering`. Ogni coppia di segmenti rappresentanti viene considerata appartenente allo stesso cluster spaziale se la distanza laterale tra i due segmenti non supera il parametro `sogliaLateraleClusterMura`, il cui valore è stato impostato a 10.

Viene in seguito invocata la funzione `nuovo_cluster_spaziale`, che permette di associare ad ogni segmento della lista `walls` il cluster spaziale a cui appartiene il rappresentante del proprio cluster muro.

Infine, tramite la funzione `extend_line` viene creata la lista dei segmenti estesi `extended_segments` e contenente oggetti di tipo `Extended_segment`, ognuno dei quali viene attribuito ad un unico cluster spaziale.

Creazione celle

Input: lista `walls`, lista `extended_segments`.

Output: lista `celle`, contenente oggetti di tipo `Superficie`.

Il modulo che crea le celle è illustrato dal punto di vista teorico nella Sezione 4.2.3. Fornendo in ingresso alla funzione `crea_edges` la lista dei segmenti estesi, `extended_segments`, è possibile ottenere la lista `edge` costituita da oggetti di tipo `Segmento`. Gli `edge` rappresentano i segmenti createsi dalle intersezioni tra gli oggetti della lista `extended_segments`. La lista `celle` viene creata tramite la funzione `crea_celle`. Ogni cella possiede come attributo la lista di `edge` che la delimitano. Invocando la funzione `setPeso`, infine, viene assegnato un peso ad ognuno degli `edge` che delimitano una cella. Questo peso stabilisce quanto l'`edge` sia coperto dagli elementi della lista `walls`.

Riconoscimento contorno

Input: mappa metrica `mappaMetrica` e lista `celle`.

Output: `contorno`, oggetto di tipo `Polygon` offerto dalla libreria `Shapely` [80] di Python.

Il modulo che estrae il contorno delle mura perimetrali di un edificio è spiegato dal punto di vista teorico in Sezione 4.2.4. Tramite la funzione `contorno_esterno` è possibile ottenere un poligono i cui lati rappresentano le mura perimetrali dell'edificio. In particolare, applicando la funzione

`cv2.findContours` offerta dalla libreria openCV sulla mappa metrica parziale, si ottiene una lista di contorni riconosciuti nell’immagine. La lista dei contorni viene in seguito ordinata in ordine crescente in base all’area interna. Si elimina il primo elemento della lista, poiché rappresenta il contorno dell’immagine stessa e si seleziona il secondo, poiché risulta essere quello che avvolge le mura perimetrali dell’edificio. Il contorno così selezionato viene in seguito utilizzato per classificare le celle della lista `celle` come interne o esterne al contorno, grazie alla funzione `classificazione_superfici`.

Clustering celle

Input: lista `celle`. Parametri DBSCAN: `eps` e `minPts`

Output: lista `stanze`, lista contenente oggetti di tipo `Polygon`.

La prima operazione che compie questo modulo, illustrato in Sezione 4.2.5, è quella di creare la matrice di affinità tra le celle nella lista `celle`. La matrice di affinità viene utilizzata dalla funzione DBSCAN, la quale raggruppa le celle in cluster diversi, ognuno rappresentante una stanza. I parametri della funzione DBSCAN sono stati impostati ai valori `eps=0.85` e `minPts=1`. Tutte le celle facenti parte dello stesso cluster vengono così unite creando un unico poligono con il quale viene rappresentata la stanza. L’insieme delle stanze viene rappresentato dalla lista `stanze`, terminando così il processo da cui è possibile ottenere il layout.

5.1.3 Visualizzazione dei dati

Riportare i dati elaborati in maniera visiva all’interno di un’immagine può essere utile non solo per rappresentare il layout finale, ma anche per ricevere feedback immediati sugli step intermedi del sistema. Nel progetto oggetto di questo lavoro di tesi il compito di visualizzare tali dati viene svolto dal file Python `disegna.py`. Questo raggruppa al suo interno una serie di funzioni che permettono di visualizzare differenti aspetti durante tutto il processo di costruzione del layout e di predizione, quali ad esempio visualizzare le celle e le stanze che compongono il layout o la posizione nello spazio dei segmenti che rappresentano le mura della mappa metrica.

Ogni funzione interna al file `disegna.py` si basa sulla libreria Python `Matplotlib` [56].

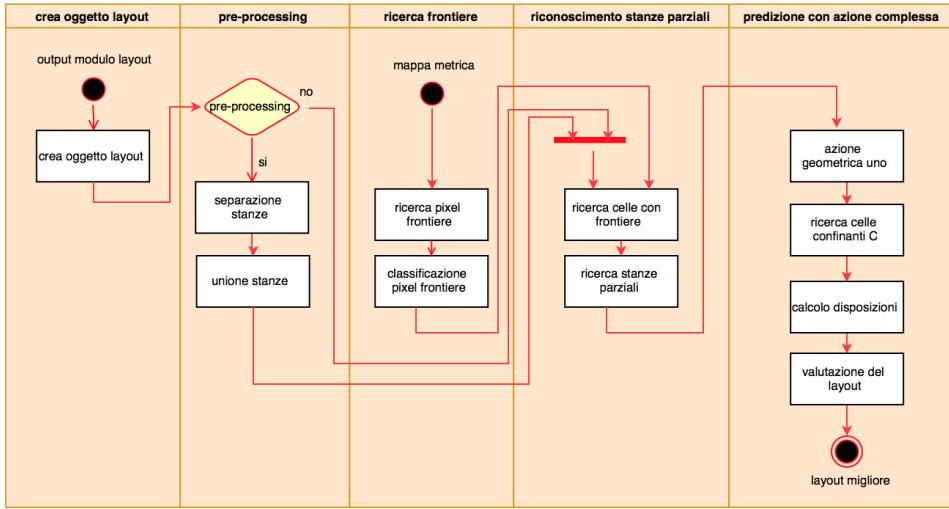


Figura 5.4: Diagramma di attività ad alto livello del modulo predizione. Ogni colonna corrisponde alla divisione logica con la quale il modulo è stato suddiviso.

5.2 Architettura del modulo di predizione

In questa sezione si vuole descrivere l’architettura generale del sistema che implementa la predizione. Questo processo, come per il layout, può essere visto come una sequenza di funzioni sviluppate in Python, anch’esso utilizzato offline. Il design del modulo predizione è composto principalmente da quattro elementi che ne indicano la divisione logica. Questi elementi sono: il pre-processing, la ricerca delle frontiere, il riconoscimento delle stanze parziali ed infine la predizione, che può essere compiuta sia attraverso delle azioni geometriche semplici sia attraverso una più complessa. Tutti questi processi sono stati precedentemente descritti nel Capitolo 4 e per tal motivo in questa sezione si vuole fornire una documentazione sui dettagli implementativi del software sviluppato. La Figura 5.4 mostra il processo d’esecuzione dell’intero modulo di predizione, il cui passo più significativo è l’azione geometrica complessa. Come si può inoltre notare in Figura 5.4, la prima operazione che svolge il modulo è quella di creare l’oggetto layout. Questa operazione è resa necessaria dal fatto che il modulo layout non fornisce un vero e proprio oggetto di tipo layout ma una serie di liste che lo descrivono. Dunque, per semplificare le operazioni all’interno del modulo di predizione si è scelto di creare un oggetto apposito.

Di seguito vengono illustrati input ed output per ogni processo che compone il modulo predizione.

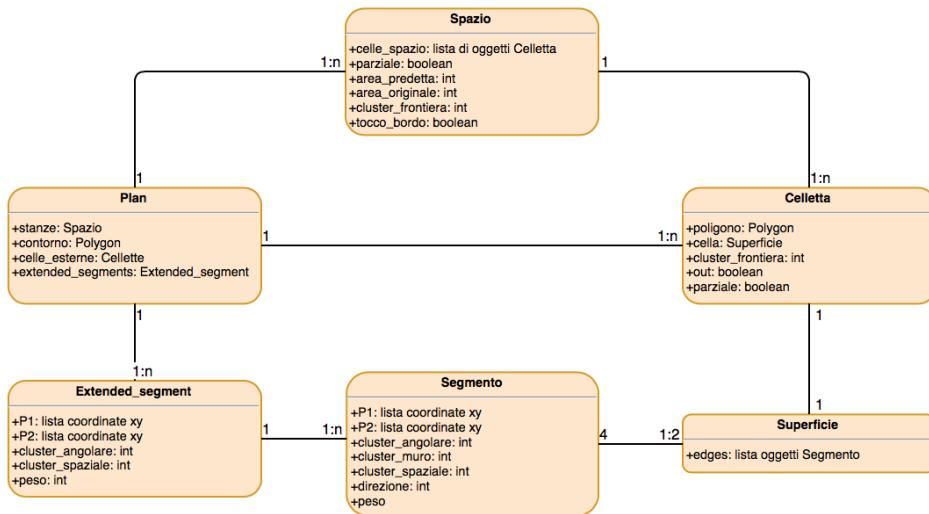


Figura 5.5: Relazione tra gli oggetti estratti dal modulo layout dopo aver creato l'oggetto layout.

5.2.1 Crea oggetto layout

Input: lista stanze, lista celle, lista walls, lista extended_segments, contorno.

Output: plan, oggetto di tipo Plan che descrive il layout.

In questa fase si riscontra la necessità di creare un oggetto che semplifichi la gestione dei dati estratti dal modulo layout. La divisione logica e le interazioni tra gli oggetti fino ad ora ottenuti viene mostrata in Figura 5.5, nella quale vengono inoltre mostrati gli attributi per ogni oggetto.

In particolare un oggetto di tipo Plan viene caratterizzato principalmente da una lista di spazi, che identificano le stanze. Ogni spazio della lista è un oggetto che raggruppa le celle della lista celle, in modo da poter identificare tutte e solo le celle che costituiscono una stanza. Un’ulteriore informazione utile agli oggetti di tipo Spazio viene rappresentata attraverso una variabile booleana che identifica se lo spazio ritrae una stanza parziale o meno, settata di default a False. Ogni elemento della lista celle viene trasformato in un oggetto di tipo Celletta. Questo oggetto oltre che alle informazioni sulla cella, dunque sugli edge che la formano, contiene anche informazioni sulla sua parzialità e posizione interna o esterna rispetto al contorno.

Da ora in poi ogni qualvolta si parlerà di layout, ci si riferirà ad un oggetto di tipo Plan. Lo stesso vale per le stanze e le celle a cui riferiscono

oggetti rispettivamente di tipo **Spazio** e **Celletta**.

5.2.2 Pre-processing

Input: oggetto **plan**, mappa metrica **mappaMetrica**.

Output oggetto **plan**.

Il modulo pre-processing, descritto in Sezione 4.3.1, prevede l'esecuzione in cascata di due operazioni che permettano di alleviare i problemi di under-segmentazione e di over-segmentazione delle stanze del layout.

La prima operazione che compie il modulo pre-processing consiste nel separare le stanze che erano state erroneamente unite ad altre dal modulo layout. Questa operazione viene eseguita identificando le porte. La tecnica utilizzata in questo lavoro di tesi si basa sul *medial axis*. Per individuare le porte si invoca la funzione `distance_transform_edt`, offerta dalla libreria Python **SciPy** [79], sulla mappa metrica. Questa operazione restituisce una matrice bidimensionale, in cui ad ogni cella viene associata la distanza della cella dal suo più vicino ostacolo. A questo punto è possibile ricavare il medial axis applicando la funzione `medial_axis` offerta dalla libreria Python **scikit-image** [77]. Questa operazione restituisce una lista contenente tutti i punti del medial axis. Viene in seguito estratta dalla lista dei punti del medial axis la lista dei punti critici che viene utilizzata dalla funzione `separaUndersegmentazione` per separare le stanze nei punti in cui rileva i punti critici, i quali corrispondono alle porte.

La seconda operazione che compie il modulo pre-processing consiste nell'unire due stanze separate erroneamente dal modulo layout. Ciò avviene invocando la funzione `unisciOversegmentazione` che unisce due stanze, se tra di loro viene riscontrata la presenza di un edge debole.

5.2.3 Ricerca Frontiere

Input: mappa metrica **mappaMetrica**.

Output: lista **pixel_frontiere**, contenente oggetti di tipo **Pixel**, i cui attributi sono: `coordinata_x`, `coordinata_y`, la lista `rgb` contenente le tre componenti RGB del pixel ed il `cluster_frontiera` che rappresenta la frontiera a cui viene associato il pixel.

La prima operazione che compie il modulo che ricerca le frontiere, precedentemente descritto dal punto di vista teorico in Sezione 4.3.2, consiste nell'andare a ricercare nella mappa metrica **mappaMetrica** le celle libere adiacenti ad almeno una cella sconosciuta e classificarle come frontiera, creando

la lista `pixel_frontiere`. Successivamente per raggruppare gli elementi della lista, identificando così le frontiere della mappa metrica, viene utilizzato l'algoritmo DBSCAN, settando i parametri `eps=20` e `min_samples=10`. A tale scopo, prima di poter effettivamente eseguire l'algoritmo DBSCAN, viene calcolata la matrice `dist_mat` le cui celle contengono la distanza euclidea tra ogni coppia di elementi in `pixel_frontiere`.

5.2.4 Riconoscimento stanze parziali

Input: oggetto `plan`, `pixel_frontiere`.

Output: setta gli attributi `parziale` e `cluster_frontiera` degli oggetti di tipo `Spazio` presenti in `plan`.

La prima operazione compiuta dal modulo, descritto in Sezione 4.3.3, consiste nell'andare a ricercare se nelle celle che formano una stanza in `plan` vi siano contenuti elementi in `pixel_frontiere`, questo equivale a ricercare all'interno di ogni cella se vi è una frontiera. Quando all'interno di una cella si riscontra un numero maggiore di zero di elementi appartenenti alla lista `pixel_frontiere`, alla cella viene settato l'attributo `cluster_frontiera` con l'id relativo alla frontiera trovata.

Una volta identificate tutte le celle con all'interno una frontiera è possibile identificare le stanze parziali. Questo perché ogni stanza risulta essere parziale se e soltanto se al suo interno esiste almeno una cella con una frontiera. Per ricercare le stanze parziali viene invocata la funzione `trova_spazi_parziali` la quale setta per ogni stanza l'attributo `cluster_frontiera` con l'id della frontiera riscontrata in una delle celle che la compongono.

5.2.5 Predizione

Input: oggetto `plan`.

Output: oggetto `plan`.

L'obiettivo di questo modulo, precedentemente descritto in Sezione 4.3.5, è quello di aggiungere la miglior combinazione di celle esterne alle stanze parziali, per ampliarle, ottimizzando una determinata score function.

Il primo passo consiste nell'andare ad aggiungere tutte e solo le celle esterne ad una stanza parziale in `plan` che siano state effettivamente viste con il sensore laser. Questo significa andare ad aggiungere quelle celle che hanno all'interno punti visti in fase d'esplorazione, ma che per via della loro bassa percentuale di ambiente esplorato non erano state classificate come

celle interne dal processo *riconoscimento contorno* del modulo layout. Per aggiungere queste celle vengono invocate in cascata le seguenti funzioni:

- **estrai_celle_confinanti_alle_parziali**: permette di estrarre tutte le celle esterne che confinano con una cella interna che contiene una frontiera.
- **trova_celle_toccate_dal_laser_beam**: dalla lista di celle estratte con la funzione precedente, si selezionano soltanto quelle che hanno all'interno porzioni esplorate dell'ambiente. La porzione minima esplorata settata in questo lavoro di tesi corrisponde a dieci pixel della mappa metrica.

Il risultato di questa operazione consiste in una lista di celle denominata **celle_confinanti**, dalla quale vengono selezionate ed aggiunte alla stanza parziale solamente le celle che non eliminano pareti già viste nella stanza.

Successivamente per tutte le celle in **celle_confinanti** che avrebbero eliminato una parete, si creano nuove stanze. Questo perché plausibilmente la parte vista non si riferisce alla stanza parziale corrente, ma ad un'altra non ancora presente nel layout.

Al termine di questa operazione, per ogni stanza parziale in **plan**, vengono estratte le celle esterne confinanti alla propria frontiera attraverso la funzione **estrai_spazio_delle_celle**. Questa funzione prevede in ingresso il parametro **level** che indica il livello di profondità con il quale si desidera selezionare le celle esterne e che è settato in questo lavoro di tesi al valore 2. La profondità equivale al numero minimo di celle che si dovrebbero attraversare per raggiungere una cella esterna, partendo dalla stanza parziale e non oltrepassando alcun muro. Sulla lista così estratta vengono calcolati tutti i possibili ordinamenti sulle celle, partendo da sequenze con un solo elemento della lista per finire con tutti i possibili ordinamenti considerando tutti gli elementi della lista.

Ottenuta la lista degli ordinamenti, è ora possibile valutarli per poi selezionare quello migliore. Tutti gli ordinamenti, che se uniti ad una stanza parziale formerebbero una stanza con “buchi” all'interno, vengono scartati poiché considerati non legali.

Il compito di valutare l'unione delle celle di un ordinamento ad una stanza parziale viene svolto dalla funzione **score_function**, la quale assegna un valore ad ognuno di essi sulla base delle metriche definite in Sezione 4.3.5.

Dopo aver assegnato un valore ad ogni ordinamento, viene selezionato quello il cui valore risulta essere massimo. La predizione termina aggiungendo alla stanza parziale sotto esame tutte le celle relative al miglior ordinamento.

Capitolo 6

Realizzazioni sperimentali e valutazione

In questo capitolo si mostrano le prove sperimentali effettuate per valutare il lavoro sviluppato in questa tesi. La trattazione del capitolo è divisa in tre parti.

- Una prima parte in cui si illustra l'impostazione generale delle prove sperimentali.
- Una seconda parte in cui si fornisce una spiegazione teorica delle metriche utilizzate per la valutazione del sistema.
- Una terza parte in cui si mostrano i risultati ottenuti dall'applicazione complessiva del sistema. La valutazione viene eseguita sia dal punto di vista qualitativo (mostrando esempi visuali) sia dal punto di vista quantitativo (utilizzando le metriche introdotte).

Tutti i risultati mostrati e discussi in questo capitolo sono stati ottenuti eseguendo il sistema su un Intel Core i5@2,4GHz CPU con 8 GB DDR3 a 1600 MHz. Il sistema è stato sperimentato su 49 mappe metriche parziali relative ad edifici scolastici reali ottenuti per mezzo dei lavori presentati in [51], [52] e [53].

6.1 Il setting

In questa sezione si fornisce una spiegazione su come sia stato impostato il lavoro per la valutazione del sistema. Per poter valutare il sistema si necessita di una mappa che fornisca il metro di paragone con cui confrontare quella ottenuta dall'applicazione del metodo proposto. La mappa a cui ci

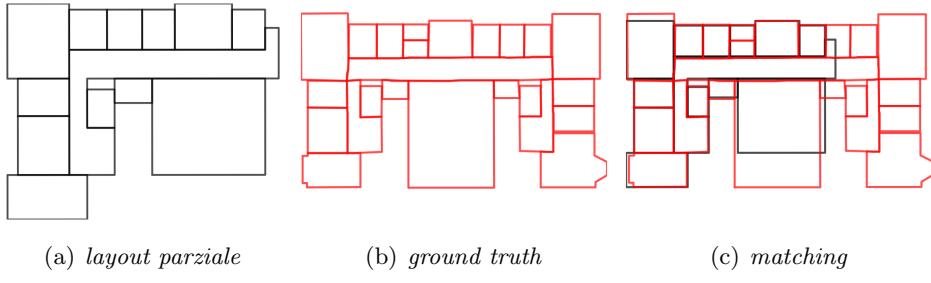


Figura 6.1: Esempio di matching tra layout parziale e ground truth. L’immagine (a) mostra il layout ottenuto dal lavoro svolto in questa tesi, l’immagine (b) mostra la mappa ground truth dell’edificio e l’immagine (c) la sovrapposizione tra il layout e la mappa ground truth.

si riferisce prende il nome di *ground truth* (*GT*). All’interno della mappa ground truth ci sono i riferimenti alla vera divisione in stanze dell’ambiente. Le informazioni sulla mappa GT, in questo lavoro di tesi, si presentano sotto forma di file XML, che viene elaborato al fine di creare un insieme di stanze GT. I file XML utilizzati per la valutazione descrivono planimetrie di edifici reali di tipologia scuola, già utilizzati nei lavori presentati in [51], [52], [53] e [13].

La difficoltà principale riscontrata durante il processo di valutazione consiste nel creare automaticamente un matching tra le stanze del layout ricostruito e quelle della mappa GT, poiché non si ha a disposizione alcuna mappa ground truth parziale. La mappa GT, infatti, rappresenta l’intero edificio e rende difficile la sovrapposizione tra le stanze del layout e quelle della mappa ground truth. La Figura 6.1 mostra un esempio in cui l’immagine (a) rappresenta le stanze estratte con il metodo proposto in questa tesi, l’immagine (b) rappresenta la mappa ground truth dell’intero edificio e l’immagine (c) la sovrapposizione tra la mappa parziale e quella ground truth.

Il meccanismo di matching, proposto in questa tesi per la valutazione del sistema, sovrappone le stanze del layout con le corrispondenti stanze della mappa ground truth (Figura 6.1 (c)) e crea un numero finito di sovrapposizioni \mathcal{M} per poi selezionare quella che meglio rispecchia la realtà sulla base di due metriche che di seguito andremo a definire.

Ogni sovrapposizione dell’insieme \mathcal{M} viene creata come segue: siano \mathcal{S} e \mathcal{G} gli insiemi delle stanze rispettivamente del layout e della mappa ground truth. Per ogni stanza $s \in \mathcal{S}$ non parziale, si crea una sovrapposizione $m(s, g) \in \mathcal{M}$ accoppiando la stanza s ad ogni stanza $g \in \mathcal{G}$. L’accoppiamen-

to viene fatto ridimensionando e traslando la stanza ground truth g considerata, e di conseguenza anche le altre, per farla corrispondere (allinearla) alla stanza del layout s .

Ogni sovrapposizione $m(s, g)$ viene valutata sulla base di due metriche che indicano la corrispondenza in termini di copertura tra le stanze del layout e quelle della mappa ground truth. La copertura è intesa come l'area sovrapposta tra due stanze. In particolare, date due stanze $s \in \mathcal{S}$ e $g \in \mathcal{G}$ di una sovrapposizione $m(s, g) \in \mathcal{M}$, la stanza s è tanto più coperta dalla stanza g quanto più la superficie di s è sovrapposta a quella di g .

Le metriche utilizzate in questo lavoro di tesi traggono ispirazione dal lavoro in [10] e [13] ed utilizzano il concetto di *forward coverage* e *backward coverage*.

Forward coverage

La forward coverage è una funzione che definisce la corrispondenza di una stanza del layout rispetto ad una della mappa ground truth. Una stanza $s \in \mathcal{S}$ si dice *forward covered* da una stanza $g \in \mathcal{G}$ della mappa ground truth se g , tra tutte le stanze in \mathcal{G} , è quella che massimizza la copertura della stanza s e viene definita dalla funzione indicatrice $\mathbf{1}_{FC}(s, g)$:

$$\mathbf{1}_{FC}(s, g) := \begin{cases} 1 & \text{se } g \text{ massimizza la copertura di } s \\ 0 & \text{altrimenti} \end{cases} \quad (6.1)$$

In relazione con la definizione di forward coverage viene anche definita la *accuracy_{FC}*. Essa indica l'accuratezza della corrispondenza tra le stanze del layout \mathcal{S} e le stanze della mappa ground truth \mathcal{G} da cui sono forward covered ed è definita come segue:

$$accuracy_{FC} := \frac{\sum_{s \in \mathcal{S}} \frac{area_{FC}(s)}{area(s)}}{|\mathcal{S}|} \quad (6.2)$$

dove $area(s)$ è la superficie di una stanza del layout, $area_{FC}(s)$ indica l'area dell'intersezione tra la stanza del layout $s \in \mathcal{S}$ e la stanza ground truth $g \in \mathcal{G}$ da cui s è forward covered. $|\mathcal{S}|$ è il numero totale di stanze del layout.

Più il valore della *accuracy_{FC}* è alto più le stanze del layout \mathcal{S} sono interamente contenute all'interno delle stanze ground truth \mathcal{G} .

Backward coverage

La backward coverage è una funzione che definisce la corrispondenza di una stanza della mappa ground truth rispetto ad una del layout. Una stanza

$g \in \mathcal{G}$ della mappa ground truth si dice *backward covered* da una stanza $s \in \mathcal{S}$ del layout se s , tra tutte le stanze in \mathcal{S} , è quella che massimizza la copertura della stanza g e viene definita dalla funzione indicatrice $\mathbf{1}_{BC}(s, g)$:

$$\mathbf{1}_{BC}(s, g) := \begin{cases} 1 & \text{se } s \text{ massimizza la copertura di } g \\ 0 & \text{altrimenti} \end{cases} \quad (6.3)$$

In relazione con la definizione di backward coverage viene anche definita la *accuracy_{BC}*. Essa indica l'accuratezza della corrispondenza tra le stanze ground truth \mathcal{G} e le stanze del layout \mathcal{S} da cui sono backward covered ed è definita come segue:

$$accuracy_{BC} := \frac{\sum_{g \in \mathcal{G}} \frac{area_{BC}(g)}{area(g)}}{|\mathcal{G}|} \quad (6.4)$$

dove $area(g)$ è la superficie di una stanza ground truth, $area_{BC}(s)$ indica l'area dell'intersezione tra la stanza ground truth $g \in \mathcal{G}$ e la stanza del layout $s \in \mathcal{S}$ da cui g è backward covered. $|\mathcal{G}|$ è il numero totale di stanze ground truth che sono backward covered da una stanza in \mathcal{S} .

Più il valore della *accuracy_{BC}* è alto più le stanze ground truth \mathcal{G} sono interamente contenute all'interno delle stanze del layout \mathcal{S} .

Una volta calcolate la *accuracy_{FC}* e la *accuracy_{BC}* per ogni sovrapposizione in \mathcal{M} tra layout e mappa ground truth è possibile selezionare quella il cui valore medio tra *accuracy_{FC}* e la *accuracy_{BC}* risulta essere massimo, poiché in generale più le due accuracy sono alte più il matching risulta migliore.

$$m^* = \arg \max_{m \in \mathcal{M}} mean(m) \quad (6.5)$$

dove m^* rappresenta la miglior sovrapposizione e $mean(m)$ segue la formula:

$$mean(m) := \frac{accuracy_{FC} + accuracy_{BC}}{2} \quad (6.6)$$

Il risultato di tale operazione viene mostrato in Figura 6.1 (c) in cui si possono notare il layout (in nero) e la mappa ground truth (in rosso) sovrapposti.

6.2 Le metriche

In questa sezione si introducono le misure utilizzate per la valutazione quantitativa delle performance dell'intero sistema. Il sistema è stato valutato

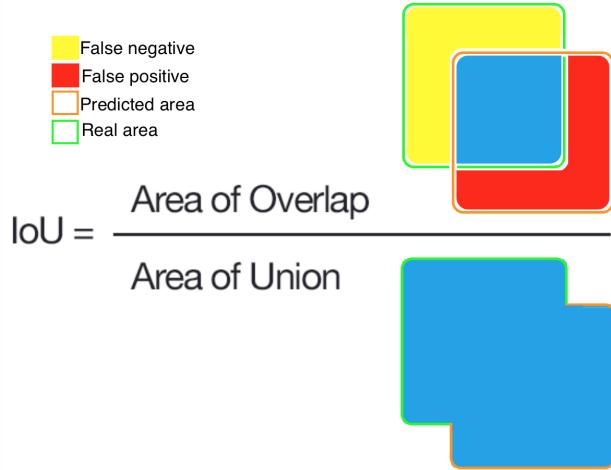


Figura 6.2: Intersection over Union. Le parti colorate in rosso rappresentano le parti erroneamente predette in una stanza parziale, mentre le parti in giallo rappresentano porzioni reali di una stanza ground truth che non sono state predette all'interno di una stanza parziale applicando il metodo di predizione.

secondo due aspetti. Il primo aspetto riguarda il task di predizione, in particolare si fornisce una misura in grado di valutare la forma della stanza predetta, usando il concetto di *Intersection over Union*(IoU). Il secondo aspetto riguarda le frontiere interne alla mappa metrica parziale, in particolare si fornisce una misura che valuti, data una frontiera, l'accuratezza dell'area predetta con il metodo di predizione introdotto in questo elaborato, che viene confrontato con altri metodi forniti dallo stato dell'arte i quali sono usati per la scelta della frontiera più promettente in fase d'esplorazione.

Di seguito vengono fornite tutte le misure utili al fine di valutare le performance del sistema.

6.2.1 Intersection over Union

Per valutare le performance del sistema, come in [2], viene usata la Intersection over Union (IoU), la cui formula per semplicità e chiarezza viene riportata in Figura 6.2. Le parti colorate in azzurro di Figura 6.2 rappresentano la regione di spazio su cui viene calcolata la IoU.

La IoU fornisce una metrica per la valutazione della correttezza della forma e della superficie della stanza su cui è stata fatta una predizione. In particolare, come emerge in [2], la IoU è definita come il rapporto dell'intersezione tra una stanza del layout e la stanza della mappa ground truth a

cui essa corrisponde, secondo il metodo di matching descritto in precedenza, con l'unione delle stesse e viene calcolata, in questo lavoro di tesi, dalla miglior sovrapposizione m^* . Le regioni di spazio erroneamente predette (false positive) vengono rappresentate in Figura 6.2 con il colore rosso e le regioni di spazio che non sono state predette nonostante esistano realmente nella mappa ground truth (false negative) vengono rappresentate con il colore giallo.

Grazie alla IoU è possibile definire quanto sia accurata la forma della predizione di una stanza parziale.

6.2.2 Le metriche sulle frontiere

Le performance del sistema vengono ulteriormente valutate utilizzando il metodo proposto in questa tesi per la misura delle informazioni potenzialmente ottenibili dalle frontiere (*information gain*) confrontandolo con altri metodi simili dello stato dell'arte, [33] e [90]. Ciò permette di confrontare la predizione dell'area non ancora esplorata attraverso il metodo introdotto in questo elaborato con l'area predetta da altri metodi, usati per ottenere una stima dell'information gain. Questa scelta risiede nella volontà di valutare il metodo di predizione proposto in questo elaborato all'interno di un'applicazione di utilizzo. In particolare si vuole valutare quanto la predizione del layout aiuti la scelta della miglior frontiera da raggiungere al fine di esplorare più velocemente l'ambiente.

In questo lavoro di tesi vengono usate due misure diverse per valutare una frontiera f .

- La prima misura si riferisce all'area predetta della stanza nella quale si trova la frontiera f e viene indicata con il termine $A_p(f)$. $A_p(f)$ equivale all'area aggiunta, applicando il metodo di predizione proposto in questa tesi, alla mappa metrica parziale in ingresso oltre una determinata frontiera f . La Figura 6.3 (a) mostra un esempio in cui si evidenziano in rosso le aree predette.
- Ogni qualvolta una stanza parziale, nella quale si presenta la frontiera f , tocca il bordo dell'edificio conosciuto fino a quel momento (cioè quando esiste una parete della stanza parziale che giace su uno dei segmenti estesi che rappresentano la bounding box della mappa metrica), la misura assegnata alla frontiera si riferisce alla lunghezza del segmento che tocca il bordo e viene indicata con il termine l_b . La Figura 6.3 (b) mostra un esempio in cui si indica con la freccia rossa il

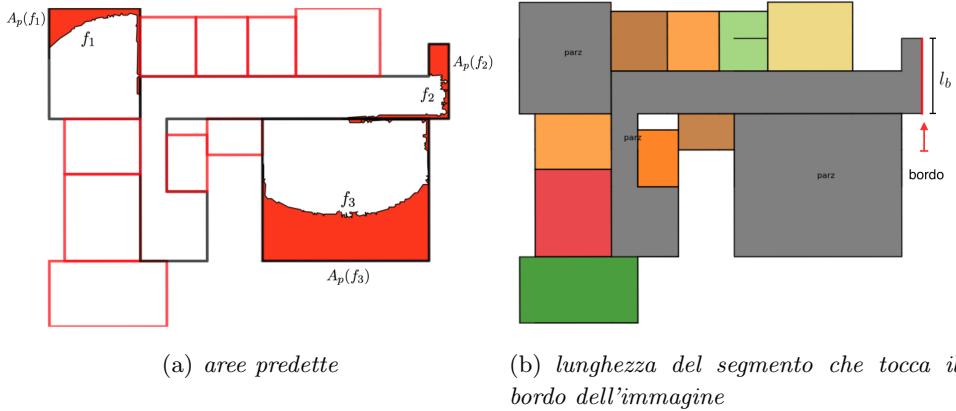


Figura 6.3: L’immagine (a) mostra in rosso le aree predette per una determinata frontiera. L’immagine (b) mostra il segmento che tocca il bordo dell’immagine.

segmento che tocca il bordo dell’immagine. La scelta di assegnare un peso particolare alle frontiere che si trovano all’interno di stanze parziali che toccano il bordo risiede nel fatto che quelle stanze potrebbero proseguire verso porzioni di ambiente non ancora visitate.

In particolare è possibile definire l’information gain relativo ad una frontiera f di una stanza parziale s come segue:

$$\text{info_gain}(f) = \{A_p(f), l_b\} \quad (6.7)$$

dove, nel caso in cui la stanza s non dovesse toccare il bordo dell’immagine, il valore di l_b risulterebbe essere *Null*.

Le misure relative all’information gain, che fornisce lo stato dell’arte e che sono state utilizzate in questo lavoro di tesi per confrontare le performance del metodo, sono due:

- **Sensore virtuale.** La prima misura, $A_c(f)$, è relativa al valore dell’area stimata in termini di area vista da un sensore virtuale a 360° posto nel punto medio di una frontiera. In particolare, prendendo spunto da [33], $A_c(f)$ viene calcolata emanando un numero finito di raggi, con centro nel punto medio della frontiera e lunghezza pari all’estensione massima raggiungibile dal sensore virtuale del robot, che in questo lavoro di tesi è stata assunta pari a 10 m . La Figura 6.4 (c) mostra un esempio, dove i raggi virtuali emanati dal sensore virtuale vengono rappresentati in azzurro. Per ogni raggio si considera il segmento di lunghezza r_{max} partendo dal centro della frontiera. Se il segmento interseca un muro precedentemente osservato, viene eliminata la

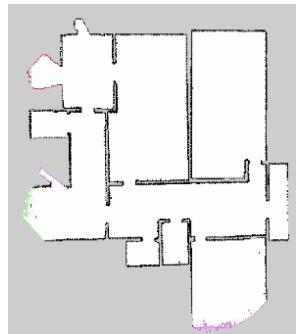
porzione di segmento successiva all’intersezione. Si calcola in seguito la lunghezza della porzione di raggio rimanente che risiede al di fuori dello spazio esplorato, cioè che sta al di fuori dalle mura perimetrali dell’edificio. $A_c(f)$ viene stimata sommando tutte le lunghezze ricavate in questo modo. Il risultato di tale operazione, eseguito su una frontiera, viene mostrato in Figura 6.4 (c), il punto rosso in Figura 6.4 (b) rappresenta il centro della frontiera.

- **Lunghezza della frontiera.** Prendendo spunto da [90] la seconda misura, $L(f)$, è relativa alla lunghezza della frontiera. Essa viene calcolata come lunghezza del segmento che congiunge i punti alle estremità di una frontiera. Il risultato di tale operazione viene mostrato in Figura 6.4 (e), i punti rossi nell’immagine in Figura 6.4 (d) rappresentano i punti della frontiera presa in considerazione.

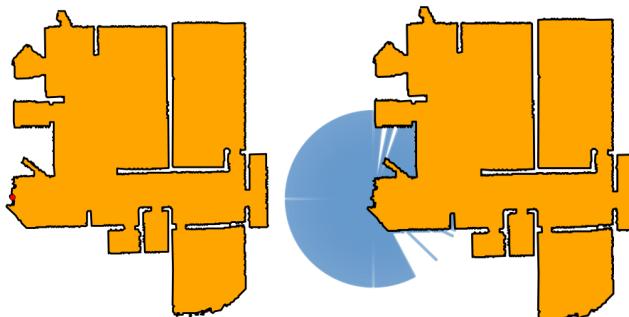
Definite queste tre misure, ad ogni frontiera di una mappa metrica parziale vengono associati i tre valori relativi alle metriche descritte. Queste misure verranno in seguito utilizzate per testare le performance del sistema, come metodo per la selezione della migliore frontiera che viene scelta durante un’esplorazione.

6.3 I risultati sperimentali

In questa sezione si presentano i risultati ottenuti dai test con i quali sono state valutate quantitativamente le performance del sistema. La migliore sovrapposizione m^* , individuata come illustrato in Sezione 6.1, permette di associare una stanza del layout ad una stanza della mappa ground truth e dunque fornisce la base di partenza per i test. La valutazione delle performance del sistema si divide principalmente in due categorie. Nella prima categoria si vuole fornire una misura che valuti le performance del task di predizione attuato sul layout. In particolare si fornisce una misura di quanto sia accurata l’area predetta di una stanza parziale; prima confrontando l’area predetta attraverso il metodo introdotto in questo elaborato con il metodo del sensore virtuale (precedentemente descritto in Sezione 6.2.2), poi si confronta la forma della stanza su cui è stata fornita una predizione con la stanza reale corrispondente della ground truth, utilizzando il concetto di Intersection over Union (precedentemente descritto in Sezione 6.2.1). La seconda categoria, invece, consiste in una serie di test, eseguiti sulle frontiere, che si basano sulle metriche introdotte precedentemente (Sezione 6.2.2) e che hanno come obiettivo quello di selezionare la frontiera più promettente per l’esplorazione, fornendo inoltre un’idea di applicazione in cui è possibile

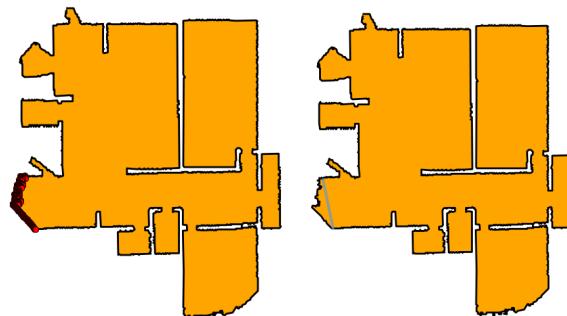


(a) *mappa metrica*



(b) *centro frontiera*

(c) *area*



(d) *punti frontiera*

(e) *linea*

Figura 6.4: Nell'immagine (a) si mostra la mappa metrica parziale su cui sono state evidenziate le frontiere con colori diversi. L'immagine (b) rappresenta il contorno della mappa ed il punto rosso il punto medio della frontiera, rappresentata dal colore verde nell'immagine (a). Nell'immagine (c) si mostra il comportamento del sensore virtuale a 360° posto sul punto medio della frontiera. L'immagine (d) rappresenta il contorno della mappa ed i punti rossi la frontiera. Nell'immagine (e) viene rappresentata la linea con la quale si approssima la frontiera verde.

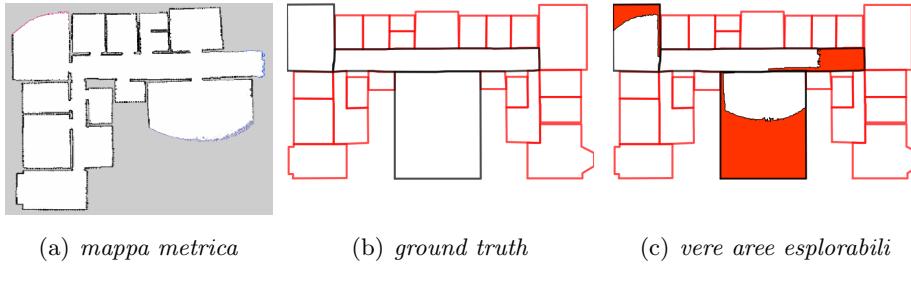


Figura 6.5: Nell’immagine (b) le pareti delle stanze ground truth al cui interno sono stante riscontrate delle frontiere vengono identificate dal colore nero, mentre tutte le altre con il colore rosso. Le parti colorate in rosso nell’immagine (c) rappresentano le vere aree potenzialmente esplorabili dalla frontiera all’interno della stanza considerata.

utilizzare il metodo.

Prima di mostrare i risultati dei test è però utile spiegare il criterio di selezione della frontiera GT con la quale si confrontano le frontiere selezionate con le tre metriche sulle frontiere. Si mappano le frontiere all’interno della mappa ground truth, permettendo di riconoscere le stanze ground truth nelle quali è presente almeno una frontiera. Se all’interno di una stanza dovessero presentarsi più frontiere, viene scelta quella che possiede, all’interno della stanza stessa, il maggior numero di pixel. A queste stanze viene sottratta l’area della parte relativa alla porzione di ambiente già esplorata. La Figura 6.5 (b) mostra le stanze ground truth in cui sono state riscontrate delle frontiere. Le parti colorate in rosso di Figura 6.5 (c) rappresentano le aree reali, $\mathcal{A}(f)$, che si possono esplorare visitando le stanze in cui si presenta una frontiera f . La migliore frontiera f_{GT} viene dunque selezionata come quella presente nella stanza la cui area reale esplorabile risulta essere maggiore rispetto alle altre.

$$f_{GT} = \arg \max_{f \in F} \mathcal{A}(f) \quad (6.8)$$

dove F rappresenta l’insieme delle frontiere di una mappa metrica parziale.

Di seguito vengono descritti i test svolti sul sistema.

6.3.1 Test sulle aree

Setting: Sia l’insieme delle frontiere F composto da $F = F_b \cup F_m$, dove F_b rappresenta l’insieme delle frontiere che si trovano in stanze che toccano,

con almeno una delle proprie pareti, un segmento esteso che si riferisce alla bounding box della mappa metrica e F_m l'insieme delle frontiere che non si trovano in stanze che toccano la bounding box. Data una mappa metrica parziale si considerano solo le frontiere dell'insieme F_m . Questo perché si vogliono valutare solamente le stanze parziali per cui il sistema può fornire una buona predizione.

Per ogni stanza in cui è contenuta una frontiera $f \in F_m$, viene stimata l'area ancora inesplorata della stanza stessa, attraverso due metriche. La prima, $A_p(f)$, rappresenta l'area stimata utilizzando il metodo sviluppato in questo lavoro di tesi. La seconda, $A_c(f)$, rappresenta l'area stimata utilizzando il sensore virtuale introdotto in Sezione 6.2.2. La vera area, non esplorata, della stanza è ottenuta dalla mappa GT e viene indicata con $\mathcal{A}(f)$. Il test sulle aree considera il valore assoluto dello scarto in termini di area dato dalla differenza tra l'area reale esplorabile $\mathcal{A}(f)$ della stanza in cui si presenta una frontiera f e l'area estratta sia con la metrika che considera l'area predetta $A_p(f)$ che con quella del sensore virtuale $A_c(f)$, pesato in base all'area reale esplorabile $\mathcal{A}(f)$:

$$\Omega(f) = \begin{cases} \Omega_1(f) = \frac{|\mathcal{A}(f) - A_p(f)|}{\mathcal{A}(f)} \\ \Omega_2(f) = \frac{|\mathcal{A}(f) - A_c(f)|}{\mathcal{A}(f)} \end{cases}, \forall f \in F_m \quad (6.9)$$

Una volta calcolato lo scarto pesato dell'area per ogni frontiera, viene calcolata la media di tutti i valori con due metriche diverse:

$$\theta = \begin{cases} \bar{\theta}_1 = \frac{\sum_{f \in F_m} \Omega_1(f)}{|F_m|} \\ \bar{\theta}_2 = \frac{\sum_{f \in F_m} \Omega_2(f)}{|F_m|} \end{cases} \quad (6.10)$$

$$\psi = \begin{cases} \bar{\psi}_1 = \frac{\sum_{f \in F_m} \Omega_1(f) \mathcal{A}(f)}{\sum_{f \in F_m} \mathcal{A}(f)} \\ \bar{\psi}_2 = \frac{\sum_{f \in F_m} \Omega_2(f) \mathcal{A}(f)}{\sum_{f \in F_m} \mathcal{A}(f)} \end{cases} \quad (6.11)$$

dove con $|F_m|$ si intende la cardinalità dell'insieme F_m .

Il risultato del test viene mostrato in Tabella 6.1. I valori di θ esposti nella prima riga della Tabella 6.1 dimostrano come l'errore sull'area stimata con la metrika del sensore virtuale sia superiore di oltre dodici volte rispetto

all'errore sull'area stimata con la metodologia di predizione introdotta in questo lavoro di tesi. Come si può inoltre notare dai valori delle deviazioni standard, il metodo introdotto in questo elaborato tende ad essere più stabile rispetto a quello del sensore virtuale in quanto i valori di $\Omega_1(f)$ discostano di poco dai valori medi θ_1 e ψ_1 . Al contrario, molti dei valori di $\Omega_2(f)$ risultano essere molto distanti dai valori medi θ_2 e ψ_2 .

In Figura 6.6 vengono riportati due grafici che mostrano sull'asse delle ordinate i valori di Ω_1 e di Ω_2 ottenuti da ogni frontiera considerata nel test e sull'asse delle ascisse i valori dell'area realmente esplorabile $\mathcal{A}(f)$ della stanza contenente f . Come si può notare nel grafico in alto della figura, per la maggior parte delle frontiere l'errore di predizione utilizzando il nostro metodo Ω_1 risulta essere molto basso, tranne che per un numero ridotto di frontiere per cui si ha un errore considerevole. Al contrario il grafico in basso di Figura 6.6 mostra che minore è l'area realmente esplorabile $\mathcal{A}(f)$, maggiore risulta essere l'errore di predizione utilizzando il metodo del sensore virtuale. Basti pensare ad una stanza quasi completamente esplorata. In questa stanza l'area realmente esplorabile dalla frontiera interna risulterà essere particolarmente ridotta, mentre quella fornita dalla metrica che considera il sensore virtuale $A_c(f)$ risulterà essere nettamente maggiore. Il metodo che considera il sensore virtuale tende a stimare sempre la stessa area da esplorare, questo spiega perché la predizione risulta particolarmente errata se $\mathcal{A}(f)$ è piccola e più precisa se $\mathcal{A}(f)$ è grande. Il metodo che considera il sensore virtuale, tende inoltre a sovrastimare l'area esplorabile. A supporto di tale affermazione si consideri inoltre la metrica:

$$E(f) = \begin{cases} E_1(f) = \mathcal{A}(f) - A_p(f) \\ E_2(f) = \mathcal{A}(f) - A_c(f) \end{cases}, \forall f \in F_m \quad (6.12)$$

Il valore medio calcolato attraverso la formula

$$\bar{E} = \begin{cases} \bar{E}_1 = \frac{\sum_{f \in F_m} E_1(f)}{|F_m|} \\ \bar{E}_2 = \frac{\sum_{f \in F_m} E_2(f)}{|F_m|} \end{cases} \quad (6.13)$$

viene riportato nella terza riga di Tabella 6.1. Il valore di \bar{E} ottenuto dal metodo che considera il sensore virtuale risulta essere negativo perché l'area stimata con questo metodo è una sovrastima dell'area realmente esplorabile per la maggior parte delle frontiere.

	nostro metodo	sensore virtuale	numero frontiere
θ	0.3767 (0.4899)	4.6413 (5.9131)	68
ψ	0.3394 (0.4913)	1.6644 (6.6202)	68
\bar{E}	3856.88 (10754.64)	-23466.38 (27068.82)	68

Tabella 6.1: Test sulle aree applicato su un totale di 68 frontiere.

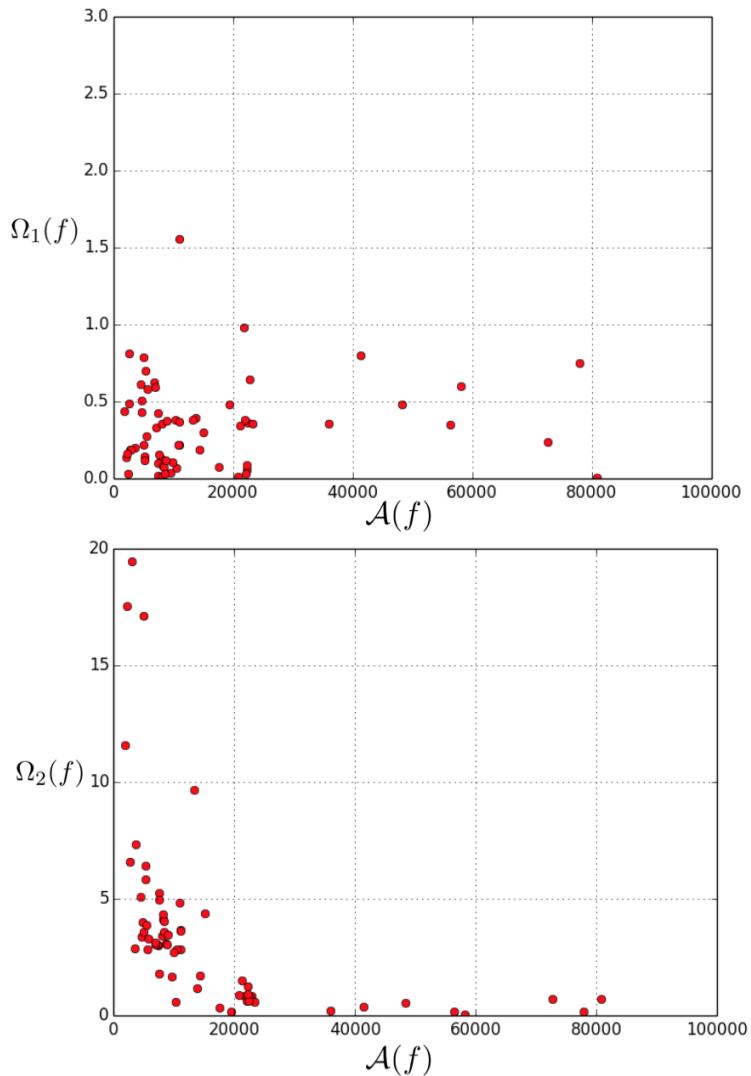


Figura 6.6: Grafici con i valori degli errori Ω_1 e Ω_2 . Sull'asse delle ascisse i valori delle aree realmente esplorabili $\mathcal{A}(f)$ da una frontiera e sulle ordinate i valori degli errori di predizione Ω_1 e Ω_2 . Si noti la differenza in scala dell'asse delle ordinate nei due grafici.

	numero mappe	numero stanze parziali	IoU medio
IoU	42	96	81.63%

Tabella 6.2: Intersection over Union medio ottenuto da 96 stanze parziali che non toccano il bordo dell’immagine, estratte da 42 mappe metriche parziali.

6.3.2 Intersection over Union

In questo lavoro di tesi la IoU è definita come il rapporto fra l’area dell’intersezione di una stanza parziale del layout che non tocca il bordo dell’immagine e la stanza della mappa ground truth a cui essa corrisponde e l’area dell’unione delle stesse. La IoU viene calcolata a partire dalla miglior sovrapposizione m^* . L’obiettivo principale del calcolo della IoU non risiede nella volontà di valutare il sistema come metodo per la ricostruzione del layout, come in [2], ma si vuole fornire una misura che sia in grado di valutare quantitativamente quanto effettivamente la forma della parte di mappa predetta corrisponda alla realtà. Per evitare dunque di considerare all’interno del valore della IoU errori dovuti all’under-segmentazione, precedentemente descritta in Sezione 4.3.1, le stanze parziali considerate per il calcolo della IoU sono state filtrate eliminando quelle porzioni di stanza che sono state under-segmentate, ossia vengono eliminate quelle porzioni di una stanza parziale del layout sovrapposte a stanze GT che non corrispondono alla stanza parziale stessa.

In Tabella 6.2 viene riportato il valore medio della IoU considerando 96 stanze parziali ottenute da 42 mappe metriche parziali. In particolare la percentuale media della IoU ottenuta permette di affermare che il metodo sviluppato in questo elaborato fornisce una buona approssimazione di completamento di ambienti non del tutto esplorati. Il valore della IoU risulta essere significativo nonostante venga valutato a partire dal miglior matching m^* che non risulta essere sempre perfetto e che quindi introduce un errore. La Figura 6.7 mostra un esempio di miglior sovrapposizione m^* in cui il layout riconoscibile dai contorni neri rappresenta il layout parziale e quello dai contorni rossi rappresenta la mappa GT. Come si può notare, nonostante la sovrapposizione tra il layout parziale e la mappa GT risulti essere accettabile esistono comunque porzioni non perfettamente sovrapposte. Questi piccoli errori influenzano il calcolo della IoU.

In Figura 6.8 vengono riportati due grafici. Il grafico in alto mostra sull’asse delle ordinate la IoU estratta da ogni stanza parziale e sull’asse delle ascisse la percentuale realmente esplorata della stanza parziale considerata. Analizzando l’immagine, e come ci si potrebbe aspettare, si può notare che

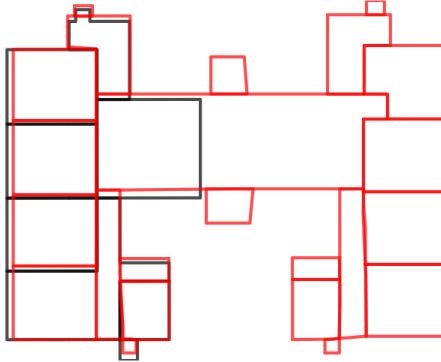


Figura 6.7: La miglior sovrapposizione m^ dell'esempio mostra come, nonostante il layout parziale (nero) e la mappa GT (rosso) abbiano all'incirca la stessa forma, esse non siano perfettamente allineate.*

la concentrazione maggiore dei valori della IoU si ottiene per percentuali alte d'esplorazione della stanza parziale (maggiori del 60%). Nonostante questo però, contrariamente a come ci si potrebbe aspettare, esistono casi in cui si hanno stanze la cui IoU risulta essere alta (superiore al 60%) avendola esplorata poco (percentuale d'esplorazione della stanza inferiore al 20%).

Il grafico in basso mostra sull'asse delle ordinate la IoU media delle stanze che compongono ogni mappa metrica parziale considerata e sull'asse delle ascisse la percentuale della mappa realmente esplorata. Una piccola analisi sul grafico mostra come la percentuale d'esplorazione dell'ambiente influisca poco sulla IoU media di una mappa, in quanto la concentrazione maggiore di valori della IoU (maggiori dell'80%) viene ugualmente distribuita non appena si raggiunge una percentuale d'esplorazione dell'intero ambiente superiore al 40%. Questo accade perché, quando si raggiunge un numero sufficiente di rette rappresentative dei muri, la percentuale di edificio nota risulta essere meno rilevante.

6.3.3 Primo test sulle frontiere

Setting: data una mappa metrica parziale si considerano solo le frontiere dell'insieme F_m , ossia l'insieme delle frontiere in F che non si trovano in stanze che toccano, con almeno una delle proprie pareti, un segmento esteso che si riferisce alla bounding box della mappa metrica. Questo perché si vogliono valutare solamente le frontiere che si trovano in stanze per cui il sistema può fornire una buona predizione.

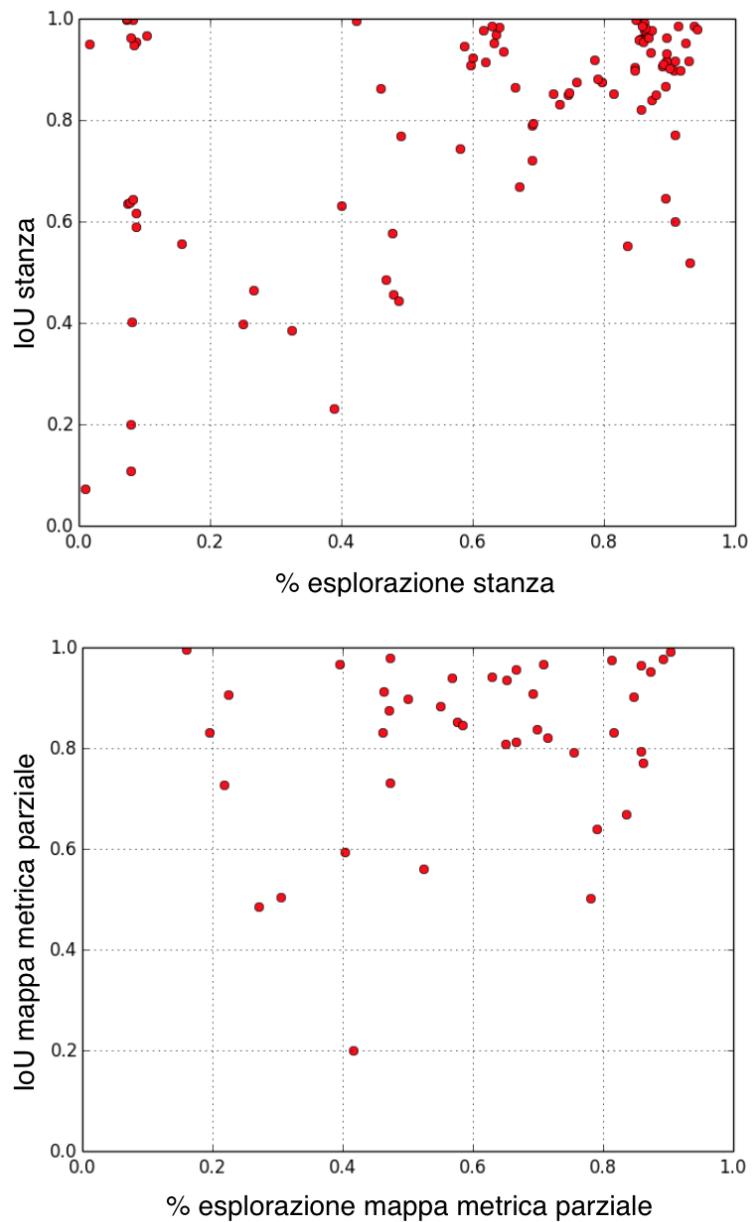


Figura 6.8: Intersection over Union. Nell'immagine in alto vengono riportati i valori della IoU stanza per stanza. Nell'immagine in basso vengono riportati i valori della IoU di ogni mappa.

	nostro metodo ($f_{GT} = f_t$)	sensore virtuale ($f_{GT} = f_c$)	metodo lunghezza ($f_{GT} = f_l$)	numero frontiere	numero mappe
Test 1 a	35 (92%)	29 (76%)	26 (68%)	68	38
Test 1 b	17 (85%)	11 (55%)	8 (40%)	50	20
Test 2	31 (63 %)	25 (51%)	16 (32%)	121	49

Tabella 6.3: Il valore numerico associato ad ogni colonna della tabella indica il numero di volte con cui i tre metodi selezionano la frontiera migliore f_{GT} e tra parentesi la percentuale. Nella seconda, terza e quarta colonna viene riportato il numero di volte con cui la frontiera f_{GT} viene selezionata rispettivamente con il metodo proposto in questa tesi, con il metodo che considera il sensore virtuale e con il metodo che considera la lunghezza della frontiera. La quinta e sesta colonna indicano rispettivamente il numero di mappe ed il numero di frontiere su cui sono stati effettuati i test.

Lo scopo del primo test sulle frontiere è quello di calcolare quante volte la migliore frontiera f_{GT} è selezionata rispetto ai metodi per la scelta delle frontiere introdotti in precedenza nella Sezione 6.2.2. Si noti che, escludendo le frontiere che si trovano in stanze che toccano la bounding box, la miglior frontiera selezionata con la metrica relativa alla metodologia sviluppata in questo lavoro di tesi f_t consiste solamente nel ricercare la frontiera che fornisce la maggior area predetta:

$$f_t = \arg \max A_p(f) \quad (6.14)$$

Se si considerano le metriche fornite dallo stato dell'arte con le quali si confronta la metodologia introdotta in questo lavoro di tesi, la miglior frontiera f_c selezionata con il metodo che considera il sensore virtuale viene selezionata come segue:

$$f_c = \arg \max A_c(f) \quad (6.15)$$

La miglior frontiera f_l selezionata considerando la metrica sulla lunghezza della frontiera viene selezionata come segue:

$$f_l = \arg \max L(f) \quad (6.16)$$

Una volta selezionate le migliori frontiere f_t , f_c e f_l con le tre metodologie, il primo test sulle frontiere consiste nel conteggiare quante volte la miglior frontiera f_{GT} sia uguale a f_t , f_c e f_l prendendo in considerazione 38 mappe metriche parziali con un totale di 68 frontiere. Il risultato del test viene mostrato nella prima riga (Test 1 a) della Tabella 6.3.

Potrebbe capitare che all'interno di una mappa esista solamente una frontiera. La scelta della miglior frontiera in quei casi risulta essere obbligata. Al fine di fornire una misura che escluda le scelte obbligate considerando

solamente i casi nei quali si deve realmente prendere una decisione, vale a dire quando all'interno di una mappa parziale esistono almeno due frontiere, è stato deciso di fornire nella seconda riga (Test 1 b) di Tabella 6.3 il risultato del medesimo test escludendo però le mappe nelle quali non si deve attuare alcuna scelta tra frontiere.

Il risultato ottenuto da questo test mostra che la metodologia sviluppata in questo lavoro di tesi permette di scegliere la miglior frontiera per l'esplorazione un numero di volte maggiore rispetto alle altre metodologie.

6.3.4 Secondo test sulle frontiere

Il secondo test proposto per la valutazione delle performance consiste, analogamente al primo, nel contare quante volte la migliore frontiera f_{GT} è selezionata come tale in base alle stime dei tre metodi. La differenza sostanziale con il primo test risiede nel fatto che in questo caso vengono considerate tutte le frontiere, dunque anche quelle che si trovano in stanze che toccano con almeno una delle proprie pareti la bounding box della mappa parziale. Considerare tutte le frontiere porta un cambiamento della funzione con la quale si sceglie la miglior frontiera f_t , selezionata con la metrica relativa alla metodologia sviluppata in questo lavoro di tesi. In particolare la miglior frontiera f_t viene selezionata in base all'area predetta più ampia, se non esistono stanze del layout che toccano il bordo. Altrimenti, se esiste almeno una stanza che tocca il bordo, f_t viene selezionata in base alla lunghezza della parete più grande che tocca il bordo tra le stanze del layout:

$$f_t = \begin{cases} \arg \max A_p(f) & \text{se } \forall s \in \mathcal{S}, s \text{ non tocca il bordo} \\ \arg \max l_b & \text{se } \exists s \in \mathcal{S} \text{ che tocca il bordo} \end{cases} \quad (6.17)$$

dove con \mathcal{S} si intende l'insieme di tutte le stanze del layout.

Anche in questo caso, il test consiste nel conteggiare quante volte la miglior frontiera f_{GT} è uguale a f_t , f_c e f_l considerando un determinato numero di mappe, che in questo test ammonta a 49 con un totale di 121 frontiere. Il risultato del test viene mostrato nella terza riga (Test 2) della Tabella 6.3.

Il risultato del test anche in questo caso mostra che la metodologia sviluppata in questo lavoro di tesi permette di selezionare la miglior frontiera un numero di volte maggiore rispetto alle altre. Questo test dimostra inoltre che impiegando la metodologia di predizione proposta in questo elaborato si potrebbe semplificare significativamente il task d'esplorazione, in quanto si potrebbe considerare di inviare il robot solo verso quelle frontiere che toccano la bounding box della mappa metrica e di predire invece le porzioni

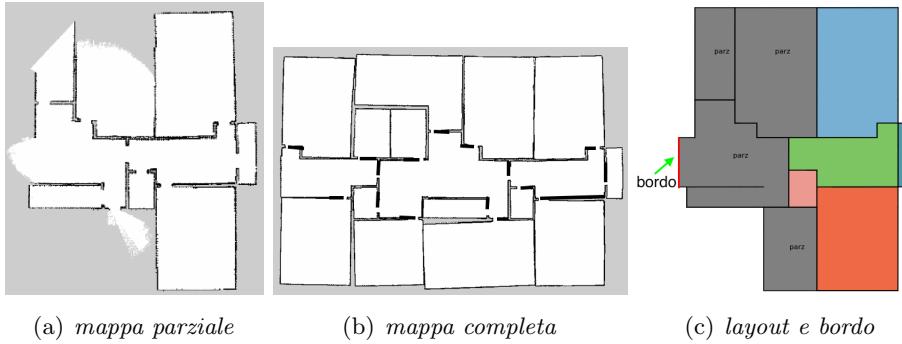


Figura 6.9: L’immagine (a) rappresenta la mappa metrica parziale e l’immagine (b) la mappa completa dell’ambiente. Nell’immagine (c) si mostra il layout e si indica con la freccia verde la parete (in rosso) della stanza che tocca il bordo della mappa metrica.

dell’ambiente su cui è possibile fare una predizione esaustiva, riducendo di conseguenza i punti dell’ambiente verso cui il robot deve dirigersi durante l’intero task d’esplorazione. Inoltre, inviare il robot verso le frontiere che toccano il bordo potrebbe portare verso porzioni molto grandi dell’ambiente che non sono ancora state esplorate. In Figura 6.9 viene riportato un esempio di quanto appena affermato in cui l’immagine (a) rappresenta la mappa metrica parziale, l’immagine (b) la mappa completa e l’immagine (c) il layout nel quale si indica, grazie alla freccia verde, il segmento che approssima la frontiera della stanza che tocca il bordo dell’immagine. Come si può notare, se un robot si dirigesse verso quella frontiera sarebbe in grado di scoprire una porzione di ambiente molto più grande (poiché sarebbe in grado di visitare nuove stanze) rispetto a quanto accadrebbe visitando le altre stanze parziali del layout (che porterebbero semplicemente al completamento della stanza stessa).

6.3.5 Complessità temporale dell’azione complessa

Nella Sezione 4.3.5 è stata spiegata l’azione complessa e ne è stata introdotta la sua complessità. Quest’azione risulta essere la fase computazionale più onerosa del sistema proposto in questa tesi soprattutto dal punto di vista temporale, perché per poter scegliere la miglior ipotesi di completamento di una stanza parziale tra le tante candidate si richiede comunque che vengano valutate tutte. Il numero di ipotesi da valutare dipende dal numero di celle esterne e confinanti ad una stanza parziale, l’insieme C , e maggiore è il numero di elementi in questo insieme maggiore è il numero di ipotesi da valutare. Il numero di ipotesi da valutare è nel numero di elementi dell’insieme \mathcal{D} . Ogni elemento in \mathcal{D} rappresenta un ordinamento diverso di celle in C .

	numero occorrenze	\mathcal{D}	\mathcal{D}_{leg} medio	tempo [s]
$C = 1$	11	1	0.7	0.0506
$C = 2$	17	4	1.8	0.0279
$C = 3$	6	15	5.0	0.0464
$C = 4$	5	64	11.2	0.1737
$C = 5$	1	325	27.0	0.4952
$C = 7$	1	13699	137.0	6.6270

Tabella 6.4: Tempi d'esecuzione per la valutazione delle ipotesi di completamento di una stanza parziale, in base al numero di elementi dell'insieme C .

che possono essere aggiunte ad una stanza parziale. Molti degli elementi in \mathcal{D} risultano essere non legali poiché rappresentano un ordinamento di celle che potrebbe portare ad una stanza con all'interno un buco. A tale scopo si definisce con il simbolo \mathcal{D}_{leg} l'insieme di tutti e soli gli elementi legali in \mathcal{D} , dove $\mathcal{D}_{leg} \subseteq \mathcal{D}$.

In questa sezione si vuole fornire una misura quantitativa del tempo d'esecuzione della valutazione delle ipotesi di completamento di una stanza parziale con riferimento al numero di elementi dell'insieme C . Dal conteggio del tempo viene esclusa la prima parte dell'azione complessa che si riferisce all'azione geometrica 1. In Tabella 6.4 vengono riportati i tempi necessari a valutare le ipotesi di completamento, a selezionare la migliore ipotesi e ad aggiungerla alla stanza parziale. Come si può notare, in Tabella 6.4 più è alto il numero degli elementi in C maggiore risulta essere il tempo necessario al fine di compiere il task. Il tempo, infatti, presenta una crescita esponenziale.

6.4 Esempi del metodo

In questa sezione si mostrano esempi visuali sul comportamento del sistema e sulla scelta della miglior frontiera.

Come si può notare in Figura 6.10 l'esecuzione prevede i tre stadi precedentemente descritti nel Capitolo 4: la mappa metrica, il layout e la predizione.

Nel primo stadio si fornisce al sistema la mappa metrica sotto forma di occupancy grid map. La mappa metrica viene successivamente elaborata nel secondo stadio creando un modello di rappresentazione che ne mostra la suddivisione in stanze. Nel terzo stadio vengono riconosciute le frontiere che permettono di riconoscere le stanze parziali che in seguito verranno completate. Nell'esempio di Figura 6.10 le frontiere sono f_1 , f_2 e f_3 . La

frontiera f_2 risiede all'interno di una stanza la cui forma ricorda quella di un corridoio. A livello logico, quando si parla di esplorazione di un ambiente, ci si aspetterebbe di incrementare maggiormente la conoscenza sulla parte dell'ambiente sconosciuta esplorando un corridoio. Questo perché generalmente su un corridoio si affacciano molte stanze. Quest'idea risiede alla base dell'esempio in Figura 6.10. Le predizioni fatte sulle stanze, al cui interno è possibile riconoscere le frontiere f_1 e f_3 , forniscono approssimativamente l'esatta dimensione e forma delle stanze reali corrispondenti, cosa che non accade invece per la stanza al cui interno è possibile riscontrare la frontiera f_2 . Una delle pareti della stanza in cui si riscontra f_2 , però, tocca il bordo dell'immagine. Questo permette di selezionare f_2 come miglior frontiera secondo la formula (6.17).

Senza il metodo di predizione proposto, una possibile strategia per la selezione della miglior frontiera sarebbe potuta essere quella di selezionare la frontiera più ampia, f_3 , il cui contributo informativo atteso risulta essere nettamente inferiore a quello della frontiera f_2 . Al fine di dimostrare la veridicità di quanto appena affermato si rimanda il lettore all'immagine in alto a destra della mappa completa di Figura 6.10, da cui è possibile osservare la reale porzione di ambiente potenzialmente esplorabile dalla frontiera f_2 .

Un ulteriore esempio viene mostrato in Figura 6.11. All'interno della mappa metrica si possono distinguere quattro frontiere diverse, f_1 , f_2 , f_3 e f_4 . Come si può notare la predizione del layout permette di fornire una buona approssimazione sulla struttura di alcune delle stanze parziali, in quanto le stanze in cui si trovano le frontiere f_1 ed f_3 vengono approssimate quasi all'esatta forma e dimensione delle loro corrispondenti GT (si noti il confronto con la mappa completa in alto a destra). Nell'esempio, inoltre, il numero di frontiere da esplorare viene significativamente ridotto poiché solo le frontiere f_2 e f_4 vengono considerate promettenti. In particolare, e come si può notare dalla mappa metrica completa in alto a destra, la frontiera f_2 permette di raggiungere la gran parte del territorio non ancora esplorato dell'ambiente.

I due esempi introdotti mostrano il processo che permette di fornire un'ipotesi di completamento di una mappa parziale e il suo uso per selezionare la frontiera più promettente, ossia quella che, se esplorata, apporta con maggior probabilità il più ampio contributo informativo all'esplorazione stessa.

Infine, in questa sezione si vuole presentare un esempio che mostri l'e-

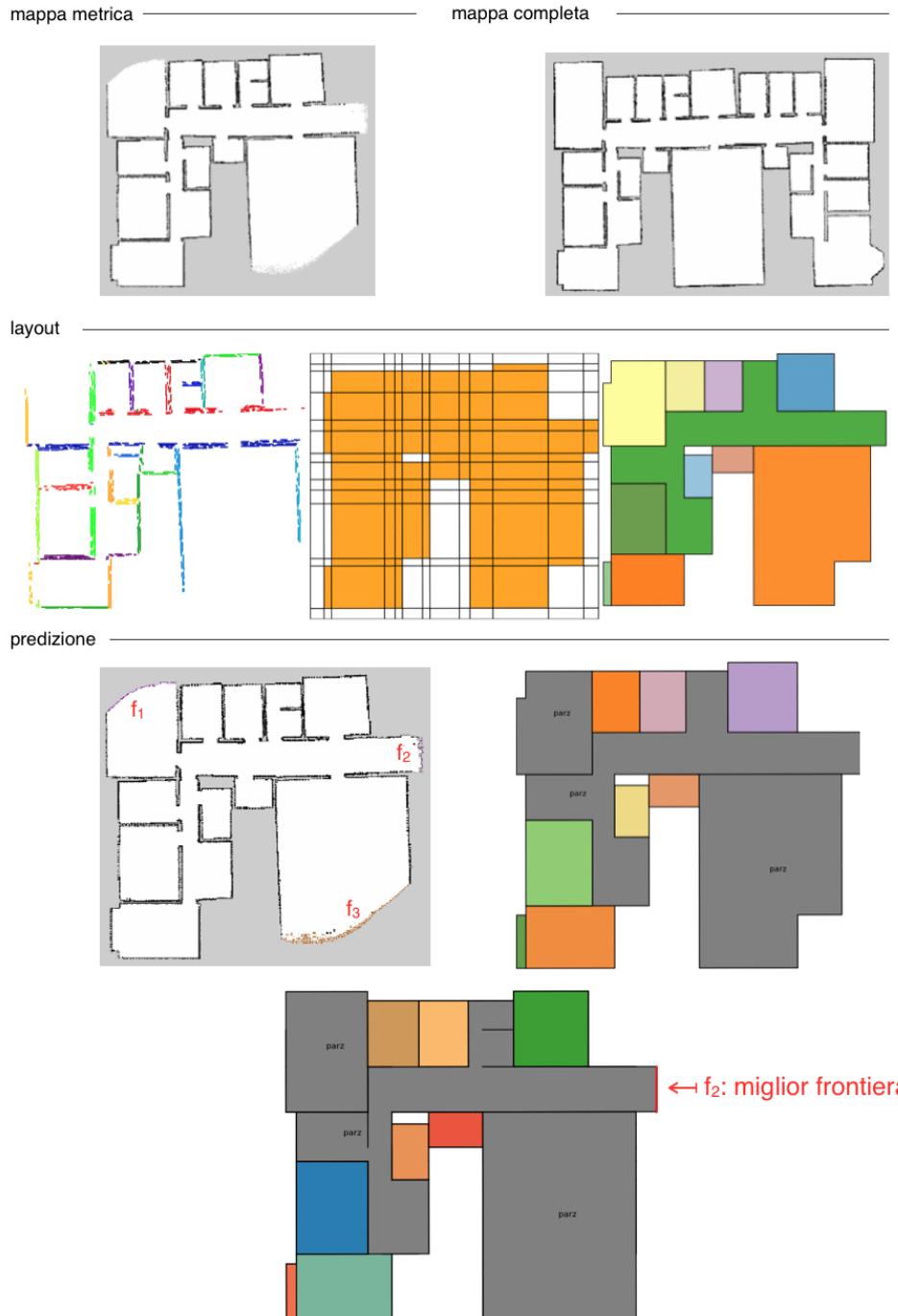


Figura 6.10: Esempio di esecuzione del sistema, con scelta della miglior frontiera da esplorare. L'esecuzione viene divisa in tre stadi: mappa metrica, layout e predizione. Nell'ultimo stadio, quello dedicato alla predizione, si associa alle frontiere f_1 , f_2 e f_3 il valore dell'information gain fornito dalla formula (6.7). I valori degli information gain vengono successivamente valutati selezionando la frontiera f_2 come miglior frontiera sulla base della formula (6.17).

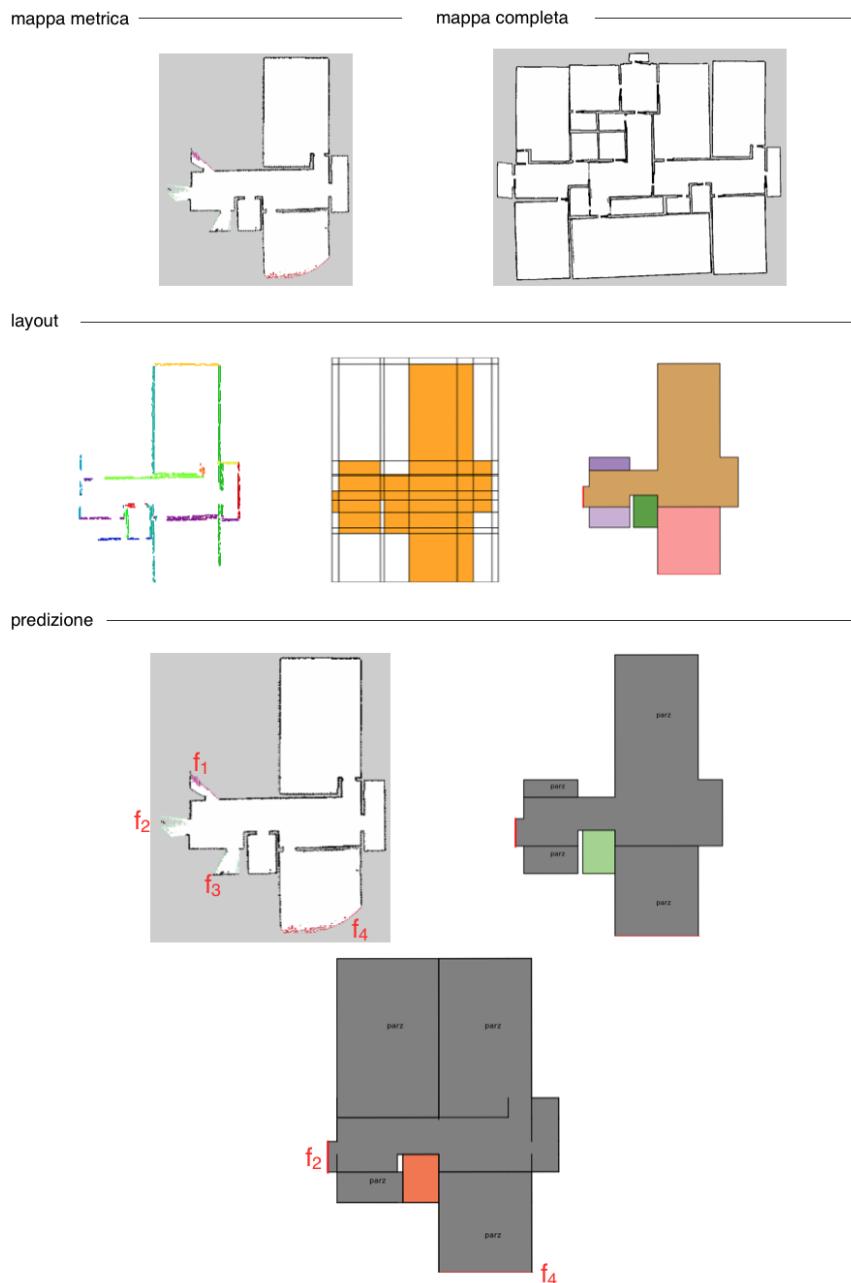


Figura 6.11: Esempio di esecuzione del sistema. L'esecuzione viene divisa in tre stadi: mappa metrica, layout e predizione. L'esempio mostra come una predizione effettuata sul layout permetta di approssimare la forma e la dimensione di alcune delle stanze parziali e come sia possibile diminuire il numero di frontiere candidate ad essere esplorate, passando da quattro frontiere (f_1 , f_2 , f_3 e f_4) a due (f_2 e f_4).

voluzione della predizione nel tempo (riportato in Figura 6.12). L'esempio comprende una sequenza di cinque mappe parziali, inerenti allo stesso ambiente, acquisite in momenti temporalmente successivi. Per ogni mappa parziale si mostrano: la percentuale esplorata dell'ambiente, la percentuale della IoU media sulla mappa e gli errori medi sulle aree θ e ψ . L'esempio mostra chiaramente come per percentuali basse d'esplorazione la predizione del layout sia comunque significativa e come, nonostante l'esplorazione non sia ancora terminata, si riesca comunque a fornire con considerevole accuratezza l'intero layout. Nell'esempio di Figura 6.12, infatti, si mostra che per ottenere una ricostruzione perfetta dell'intero layout basta esplorare l'ambiente all'80%. In Figura 6.12 vengono inoltre riportati i grafici della IoU e gli errori sulle aree (θ e ψ), mostrando come a seguito dell'aumento della percentuale di area esplorata dell'ambiente si abbia un miglior valore della IoU e una diminuzione degli errori sulle aree.

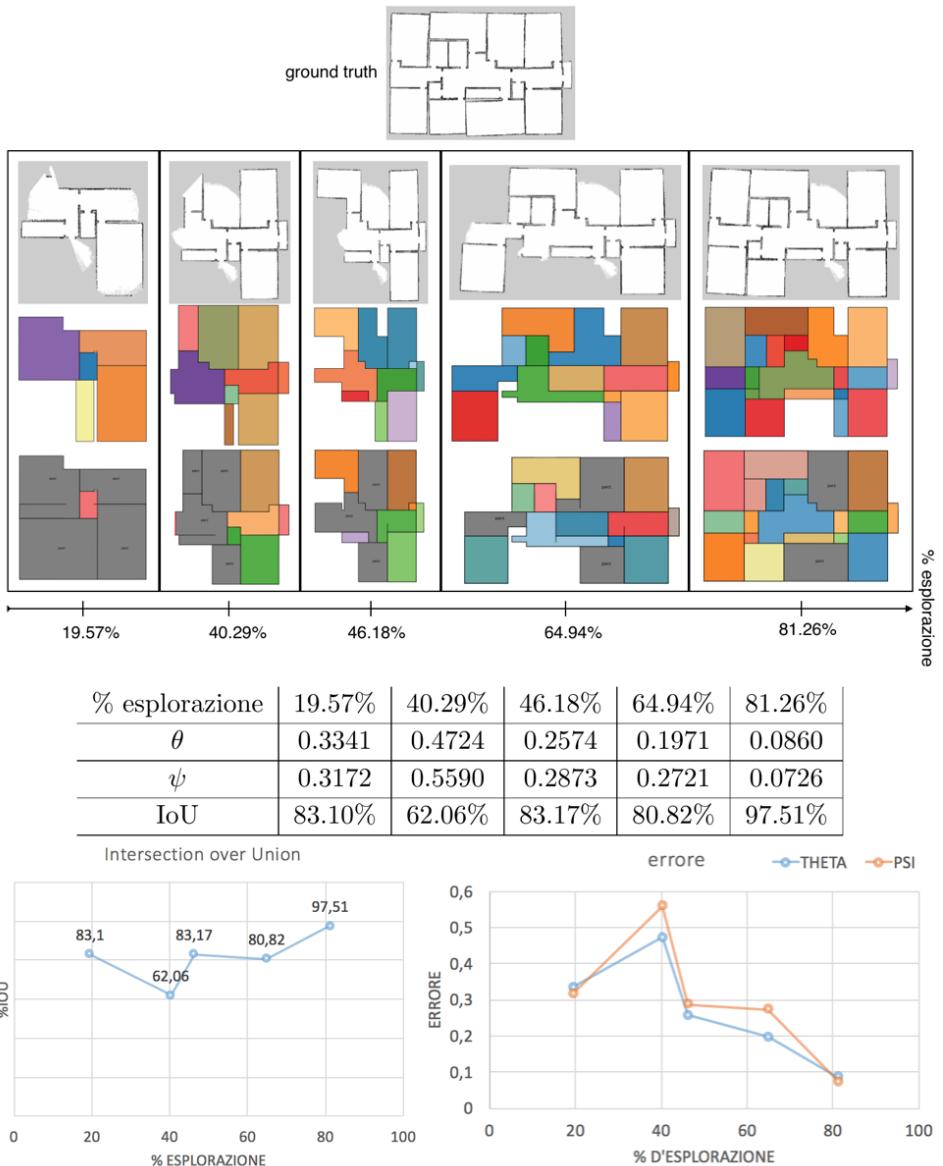


Figura 6.12: Esempio di esecuzione. Si confrontano i risultati ottenuti dall'esecuzione di cinque mappe inerenti allo stesso ambiente e temporalmente successive. In tabella vengono riportati i valori della IoU e degli errori medi sulle aree (θ e ψ). I grafici mostrano il comportamento della IoU e degli errori sulle aree (θ e ψ) a seguito dell'aumento della percentuale di area esplorata dell'ambiente.

Capitolo 7

Direzioni future di ricerca e conclusioni

In questo lavoro di tesi è stato sviluppato un sistema per la predizione di porzioni di ambiente non ancora esplorate, nell'ambito della robotica mobile autonoma. In particolare il sistema proposto è in grado, a partire da una mappa metrica parziale, di ricostruire il layout completo ad essa associato, fornendone una divisione in stanze, e successivamente di offrire un'ipotesi di completamento sulle stanze parziali. Il sistema proposto nella tesi si divide in tre fasi fondamentali: la costruzione della *mappa metrica*, la creazione del *layout* e la *predizione*.

Il punto di partenza consiste nella costruzione della *mappa metrica*. Essa fornisce informazioni di carattere metrico sulla struttura e sull'occupazione dell'ambiente. Le mappe metriche parziali utilizzate in questo lavoro di tesi vengono raccolte durante un processo simulato di esplorazione di un ambiente. Esse rappresentano infatti la conoscenza su un ambiente osservato dal robot ad istanti di tempo successivi.

Le informazioni metriche raccolte dalla mappa metrica parziale vengono raffinate al fine di creare una rappresentazione dell'ambiente che sia in grado di fornirne la disposizione e divisione in stanze. Questo processo prende il nome di creazione del *layout* ed è stato precedentemente descritto nel lavoro [13]. In questa tesi sono stati modificati ed aggiunti processi per migliorare le prestazioni e la gestione delle mappe parziali. Dalla mappa metrica vengono inoltre raccolte le informazioni sulle frontiere, permettendo di riconoscere le stanze parziali del layout, ossia quelle stanze al cui interno è presente una frontiera.

Dal layout si estraggono tutte le informazioni geometriche necessarie alla fase di *predizione*, per offrire un’ipotesi di completamento sulle stanze parziali. L’approccio proposto di predizione consiste nel generare un insieme di possibili ipotesi di completamento per ogni stanza parziale. Ogni ipotesi creata viene valutata sulla base di una *score function*, che permette di selezionare l’ipotesi che maggiormente rispecchia le informazioni geometriche estratte dalla struttura geometrica del layout e relative alle porzioni di ambiente già note.

L’intero sistema è stato valutato dal punto di vista quantitativo attraverso la definizione di alcune metriche. Le metriche introdotte si dividono in due categorie: nella prima si valutano le reali performance di predizione e nella seconda si valuta il sistema come metodo per la ricerca della prossima migliore frontiera da esplorare. La valutazione delle performance del sistema ha dimostrato che la predizione che esso fornisce rappresenta una buona approssimazione per il completamento di ambienti non del tutto esplorati e che è inoltre in grado di selezionare, nella maggioranza dei casi studiati, la miglior prossima frontiera per l’esplorazione.

In conclusione, vengono presentati alcuni dei possibili sviluppi futuri del lavoro di tesi proposto.

Eliminazione del vincolo *Manhattan world*. Uno dei limiti principali del sistema consiste nell’assunzione che gli ambienti da cui sono state estratte le mappe possiedono soltanto muri verticali ed orizzontali. Il primo e più importante sviluppo futuro consiste nel raffinare la metodologia di predizione delle stanze parziali perché sia in grado di gestire efficientemente muri non perpendicolari tra loro o curvi.

Utilizzo di dati reali. Un altro limite fondamentale sotto cui lavora il sistema sviluppato in questo elaborato consiste nell’utilizzare mappe provenienti da un ambiente simulato privo di mobilia interna. Seppur tali mappe derivino da un’esplorazione simulata di ambienti realistici, non sono mai state usate mappe provenienti da un ambiente esplorato da un robot reale per la valutazione del sistema. Uno sviluppo futuro molto importante consiste, appunto, nel raffinare il metodo affinché sia in grado di utilizzare in input mappe provenienti da esplorazioni di robot reali e che possiedano dunque imperfezioni dovute agli oggetti interni. In Figura 7.1 è mostrata,

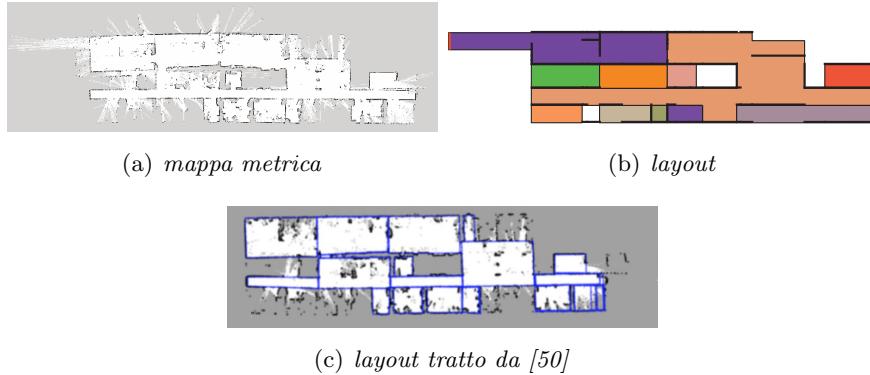


Figura 7.1: Esempio di ricostruzione del layout applicato su una mappa reale tratta da [50]. L'immagine (a) rappresenta la mappa reale in input tratta da [50]. L'immagine (b) rappresenta la ricostruzione del layout estratta in questo lavoro di tesi e (c) rappresenta la ricostruzione del layout estratta col metodo sviluppato in [50].

a titolo di esempio, l'applicazione del metodo qui proposto, senza modifiche rispetto a come descritto, su una mappa ottenuta da un robot reale e presente in un dataset pubblico, [50]. Come unico passaggio aggiuntivo, le celle occupate all'interno della stanza, dovute ad occlusione, sono state automaticamente filtrate dalla mappa stessa. Come si può notare dall'esempio, il metodo qui proposto riesce a ricostruire alcune delle caratteristiche strutturali dell'edificio in maniera efficiente, seppur introducendo numerosi artefatti. Un possibile sviluppo futuro per attenuare queste problematiche può risiedere nel filtrare, online e durante la fase di mapping, quei punti che sono considerati come “occlusione” o “oggetti”, seguendo un metodo come [1], o applicare un metodo che permetta di ottenere la struttura delle pareti in un ambiente con molta occlusione, come [2].

In [50] viene fornito un metodo per la ricostruzione del layout a partire da mappe in cui l'occlusione risulta essere consistente (come ad esempio in Figura 7.1 (c)). Il metodo proposto in [50], come si può vedere, riesce efficacemente a stimare la struttura di stanze quasi interamente esplorate, ma filtra dalla ricostruzione del layout stanze parzialmente esplorate. Il sistema proposto in questo lavoro di tesi, invece, completa le stanze parziali del layout. Siccome l'applicazione di questi due metodi potrebbe essere complementare, un possibile sviluppo futuro, che considera appunto dati reali, potrebbe essere quello di integrare i due metodi per gestire efficientemente l'incertezza e la complessità dei dati provenienti dal mondo reale.

Utilizzo di robot reali in applicazioni che prevedono l'esplorazio-

ne. Una volta testato il sistema con mappe provenienti da esplorazioni reali e non più simulate, lo step successivo prevede l'integrazione del sistema di costruzione del layout, di predizione e selezione della miglior frontiera su un robot reale impegnato ad esplorare un edificio. Difatti solo l'applicazione online da parte di un robot reale è in grado di verificare e valutare l'apporto del sistema alle performance del task di esplorazione.

Integrazione di un metodo topologico e semantico. Un altro sviluppo futuro consiste nell'integrazione del sistema studiato con una metodologia topologica e semantica. Per migliorare il sistema di predizione sviluppato in questo lavoro di tesi si potrebbe pensare di integrarlo con una metodologia che sia in grado di estrarre il grafo che mostra le connessioni tra le stanze del layout ed una metodologia che sia in grado di assegnare etichette semantiche ai nodi che rappresentano i nodi del grafo. Una base di partenza potrebbe essere semplicemente quella di distinguere efficientemente i corridoi dalle stanze. Come già affermato diverse volte durante l'elaborato, una strategia di esplorazione deve prima o poi condurre il robot a percorrere i corridoi di un edificio. Questo perché attraverso un corridoio si possono raggiungere molteplici stanze. Distinguere i corridoi dalle stanze permetterebbe dunque di riconoscere quelle frontiere che hanno una maggior probabilità di dirigere il robot verso porzioni inesplorate dell'ambiente.

Bibliografia

- [1] R. Ambruş, N. Bore, J. Folkesson, and P. Jensfelt. Meta-rooms: Building and maintaining long term spatial models in a dynamic world. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1854–1861, 2014.
- [2] R. Ambruş, S. Claici, and A. Wendt. Automatic room segmentation from unstructured 3-d data of indoor environments. *Robotics and Automation Letters*, 2(2):749–756, 2017.
- [3] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 52–59, 2016.
- [4] A. Aydemir and J. Jensfelt, P. and Folkesson. What can we learn from 38,000 rooms? reasoning about unexplored space in indoor environments. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 4675–4682, 2012.
- [5] P. Beeson, N. K. Jong, and B. Kuipers. Towards autonomous topological place detection using the extended voronoi graph. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4373–4379. IEEE, 2005.
- [6] M. Betke and L. Gurvits. Mobile robot localization using landmarks. *IEEE Transactions on Robotics and Automation*, 13(2):251–263, 1997.
- [7] J.L. Blanco, J. Gonzalez, and J.A. Fernandez-Madrigal. Consistent observation grouping for generating metric-topological maps that improves robot localization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 818–823, 2006.
- [8] I. Bloch and H. Maître. Fuzzy mathematical morphologies: a comparative study. *Pattern Recognition*, 28(9):1341–1387, 1995.

- [9] R. Bormann, J. Hampp, and M. Hägele. New brooms sweep clean—an autonomous robotic cleaning assistant for professional office cleaning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4470–4477, 2015.
- [10] R. Bormann, F. Jordan, W. Li, J.A. Hampp, and M. Hägele. Room segmentation: Survey, implementation, and analysis. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1019–1026, 2016.
- [11] E. Brunskill, T. Kollar, and N Roy. Topological mapping using spectral clustering and classification. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 3491–3496, 2007.
- [12] P. Buschka and A. Saffiotti. A virtual sensor for room detection. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 637–642, 2002.
- [13] M. Calabrese. Costruzione di mappe multilivello per robot mobili. Master’s thesis, Politecnico di Milano, 2016.
- [14] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
- [15] R. Capobianco, G. Gemignani, D. D. Bloisi, D. Nardi, and L. Iocchi. Automatic extraction of structural representations of environments. In *Intelligent Autonomous Systems 13*, pages 721–733. Springer, 2016.
- [16] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.
- [17] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.
- [18] L. Cruz, D. Lucio, and L. Velho. Kinect and rgbd images: Challenges and applications. In *Proceedings of the Conference on Graphics, Patterns and Images Tutorials*, pages 36–49, 2012.
- [19] A. Davids. Urban search and rescue robots: from tragedy to technology. *IEEE Intelligent Systems*, 17(2):81–83, 2002.

- [20] T. K. Dey and W. Zhao. Approximating the medial axis from the voronoi diagram with a convergence guarantee. *Algorithmica*, 38(1):179–200, 2004.
- [21] G. Diosi, A. and Taylor and L. Kleeman. Interactive slam using laser and advanced sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1103–1108, 2005.
- [22] G. Dudek, M. R.M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397, 1996.
- [23] S. Ekvall, D. Kragic, and P. Jensfelt. Object detection and mapping for service robot tasks. *Robotica*, 25(2):175–187, 2007.
- [24] A. Elfes. A sonar-based mapping and navigation system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1151–1156, 1986.
- [25] S. P. Engelson. *Passive Map Learning and Visual Place Recognition*. PhD thesis, Yale University, 1994.
- [26] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.
- [27] E. Fabrizi and A. Saffiotti. Extracting topology-based maps from gridmaps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2972–2978, 2000.
- [28] E. Fabrizi and A. Saffiotti. Augmenting topology-based maps with geometric information. *Robotics and Autonomous Systems*, 40(2):91–97, 2002.
- [29] Steven Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2(1-4):153–174, 1987.
- [30] S. Friedman, H. Pasula, and D. Fox. Voronoi random fields: Extracting the topological structure of indoor environments via place labeling. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2109–2114, 2007.
- [31] A. Furlan, S. D Miller, D. G. Sorrenti, F.F. Li, and S. Savarese. Free your camera: 3d indoor scene understanding from arbitrary camera

- motion. In *Proceedings of the British Machine Vision Conference*, pages 24.1–24.12, 2013.
- [32] R. Gens and P. Domingos. Discriminative learning of sum-product networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3248–3256, 2012.
 - [33] H. H. Gonzalez-Banos and J.C. Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21(10-11):829–848, 2002.
 - [34] F. Han and S.C. Zhu. Bottom-up/top-down image parsing with attribute grammar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(1):59–73, 2009.
 - [35] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *Proceedings of the International Conference on Computer Vision*, pages 1849–1856, 2009.
 - [36] S. Hemachandra, M.R. Walter, S. Tellex, and S. Teller. Learning spatial-semantic representations from natural language descriptions and scene classifications. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2623–2630, 2014.
 - [37] J. Illingworth and J. Kittler. A survey of the hough transform. *Computer Vision, Graphics, and Image Processing*, 44(1):87–116, 1988.
 - [38] A.T. Ismail, A. Sheta, and M. Al-Weshah. A mobile robot path planning using genetic algorithm in static environment. *Journal of Computer Science*, 4(4):341–344, 2008.
 - [39] J.H. Jang and K.S Hong. Fast line segment grouping method for finding globally more favorable line segments. *Pattern Recognition*, 35(10):2235–2247, 2002.
 - [40] M. Juliá, A. Gil, and O. Reinoso. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33(4):427–444, 2012.
 - [41] F. Karimipour and M Ghandehari. A stable voronoi-based algorithm for medial axis extraction through labeling sample points. In *Proceedings of the International Symposium on Voronoi Diagrams in Science and Engineering.*, pages 109–114, 2012.

- [42] N. Kiryati, Y. Eldar, and A. M. Bruckstein. A probabilistic hough transform. *Pattern Recognition*, 24(4):303–316, 1991.
- [43] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 3232–3238, 2003.
- [44] B. Kuipers and Y.T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8(1):47–63, 1991.
- [45] R. Lakaemper. Simultaneous multi-line-segment merging for robot mapping using mean shift clustering. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1654–1660, 2009.
- [46] J.C. Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.
- [47] L. Lin and M. A. Goodrich. Uav intelligent path planning for wilderness search and rescue. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 709–714, 2009.
- [48] Z. Liu and G. von Wichert. Applying rule-based context knowledge to build abstract semantic maps of indoor environments. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 5141–5147, 2013.
- [49] Z. Liu and G. von Wichert. Extracting semantic indoor maps from occupancy grids. *Robotics and Autonomous Systems*, 62(5):663–674, 2014.
- [50] Z. Liu and G. von Wichert. A generalizable knowledge framework for semantic indoor mapping based on Markov logic networks and data driven MCMC. *Future Generation Computer Systems*, 36:42–56, 2014.
- [51] M. Luperto and F. Amigoni. Exploiting structural properties of buildings towards general semantic mapping systems. In *Intelligent Autonomous Systems 13*, pages 375–387. Springer International Publishing, 2016.
- [52] M. Luperto, L. D’Emilio, and F. Amigoni. A generative spectral model for semantic mapping of buildings. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 4451–4458, 2015.

- [53] M. Luperto, A. Quattrini Li, and F. Amigoni. A system for building semantic maps of indoor environments exploiting the concept of building typology. In *Proceedings of the RoboCup Symposium*, pages 504–515, 2013.
- [54] M. C. Maccarone. Fuzzy mathematical morphology: concepts and applications. *Vistas in Astronomy*, 40(4):469–477, 1996.
- [55] M. J. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, 1992.
- [56] Matplotlib. <http://matplotlib.org>, 2017. [Online; accessed 18-March-2017].
- [57] P. Merrell, E. Schkufza, and V. Koltun. Computer-generated residential building layouts. In *Proceedings of the ACM Transactions on Graphics*, pages 181–193, 2010.
- [58] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61, 1988.
- [59] Ò. M. Mozos. *Semantic labeling of places with mobile robots*, volume 61 of *Springer Tracts in Advanced Robotics*. Springer, 2010.
- [60] Ò. M. Mozos, A. Rottmann, R. Triebel, P. Jensfelt, W. Burgard, et al. Semantic labeling of places using information extracted from laser and vision sensor data. *IEEE/RSJ IROS Workshop: From sensors to human spatial concepts*, 2006.
- [61] C. Mura, O. Mattausch, A. J. Villanueva, E. Gobbetti, and R. Pajaronla. Robust reconstruction of interior building structures with multiple rooms under clutter and occlusions. In *Proceedings of the International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics)*, pages 52–59, 2013.
- [62] C. Mura, O. Mattausch, A.J. Villanueva, E. Gobbetti, and R. Pajaronla. Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Computers & Graphics*, 44:20–32, 2014.
- [63] P. Newman, J. Leonard, J. D Tardós, and J. Neira. Explore and return: Experimental validation of real-time concurrent mapping and localization. In *Proceedings fo the IEEE International Conference on Robotics and Automation*, pages 1802–1809, 2002.

- [64] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 2:849–856, 2002.
- [65] J. Oberlander, K. Uhl, J. M. Zollner, and R. Dillmann. A region-based slam algorithm capturing metric, topological, and semantic properties. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1886–1891, 2008.
- [66] OpenCV. <http://opencv.org>, 2017. [Online; accessed 18-March- 2017].
- [67] G. Oriolo, G. Ulivi, and M. Vendittelli. Fuzzy maps: a new tool for mobile robot perception and planning. *Journal of Robotic Systems*, 14(3):179–197, 1997.
- [68] H.S. Park and C.H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2):3336–3341, 2009.
- [69] R. Peherz, S. Tschiatschek, F. Pernkopf, and P. M. Domingos. On theoretical properties of sum-product networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 744–752, 2015.
- [70] H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 689–690, 2011.
- [71] A. Pronobis, P. Jensfelt, K. Sjöö, H. Zender, G. J. Kruijff, Ö.M. Mozos, and W. Burgard. Semantic modelling of space. *Cognitive Systems*, 8:165–221, 2010.
- [72] A. Pronobis, Ö. M. Mozos, B. Caputo, and P. Jensfelt. Multi-modal semantic place classification. *International Journal of Robotics Research*, 29(2-3):298–320, 2010.
- [73] A. Pronobis and Rajesh P.N. Rao. Learning deep generative spatial models for mobile robots. *arXiv preprint arXiv:1610.02627*, 2016.
- [74] GMapping ROS Repository. https://github.com/ros-perception/slam_gmapping, 2017. [Online; accessed 18-March- 2017].
- [75] ROS. <http://www.ros.org>, 2017. [Online; accessed 18-March- 2017].

- [76] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
- [77] Scikit-image. <http://scikit-image.org>, 2017. [Online; accessed 18-March- 2017].
- [78] Scikit-learn. <http://scikit-learn.org/stable/>, 2017. [Online; accessed 18-March- 2017].
- [79] Scipy. <https://scipy.org>, 2017. [Online; accessed 18-March- 2017].
- [80] Shapely. <https://github.com/toblerity/shapely>, 2017. [Online; accessed 18-March- 2017].
- [81] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Springer Science & Business Media, 2008.
- [82] K. Sjöö. Semantic map segmentation using function-based energy maximization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4066–4073, 2012.
- [83] V. Spirin, S. Cameron, and J. de Hoog. Time preference for information in multi-agent exploration with limited communication. In *Proceedings of the Conference Towards Autonomous Robotic Systems*, pages 34–45, 2013.
- [84] C. Stachniss, Ó. M. Mozos, and W. Burgard. Speeding-up multi-robot exploration by considering semantic place information. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1692–1697, 2006.
- [85] D. Perea Ström, F. Nenci, and C. Stachniss. Predictive exploration considering previously mapped environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2761–2766, 2015.
- [86] S. Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.
- [87] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.
- [88] S. Thrun and A. Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the National Conference on Artificial Intelligence*, pages 944–951, 1996.

- [89] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.
- [90] B. Tovar, L. Muñoz-Gómez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, and S. Hutchinson. Planning exploration strategies for simultaneous localization and mapping. *Robotics and Autonomous Systems*, 54(4):314–331, 2006.
- [91] Z. Tu, X. Chen, A. L. Yuille, and S.C. Zhu. Image parsing: Unifying segmentation, detection, and recognition. *International Journal of Computer Vision*, 63(2):113–140, 2005.
- [92] E. Turner and A. Zakhori. Floor plan generation and room labeling of indoor environments from laser range data. In *Proceedings of the International Conference on Computer Graphics Theory and Applications*, pages 1–12, 2014.
- [93] Eric Turner, Peter Cheng, and Avideh Zakhori. Fast, automated, scalable generation of textured 3d models of indoor environments. *IEEE Journal of Selected Topics in Signal Processing*, 9(3):409–421, 2015.
- [94] C. A. Vanegas, D.G. Aliaga, and B. Benes. Automatic extraction of manhattan-world building masses from 3d laser range scans. *Transactions on Visualization and Computer Graphics*, 18(10):1627–1637, 2012.
- [95] R. Vaughan. Massively multi-robot simulation in stage. In: *Swarm intelligence*, 2(2):189–208, 2008.
- [96] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.
- [97] F. Wotawa. Adaptive autonomous systems—from the system’s architecture to testing. In *Leveraging Applications of Formal Methods, Verification, and Validation*, pages 76–90. Springer, 2012.
- [98] K. M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1160–1165, 2008.

- [99] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151, 1997.
- [100] H. Zender, Ò. M. Mozos, P. Jensfelt, G.J. Kruijff, and W. Burgard. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56(6):493–502, 2008.
- [101] Z. Zivkovic, B. Bakker, and B. Kroese. Hierarchical map building and planning based on graph partitioning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 803–809, 2006.

Appendice A

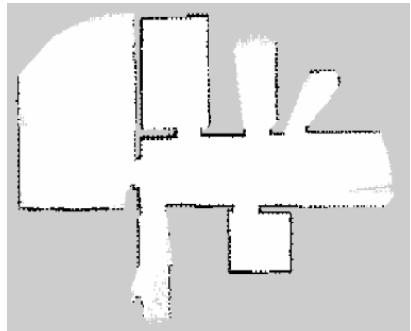
Risultati

In questa appendice vengono mostrati i risultati ottenuti su alcune delle mappe metriche parziali considerate applicando l'approccio di predizione sviluppato in questo lavoro di tesi.

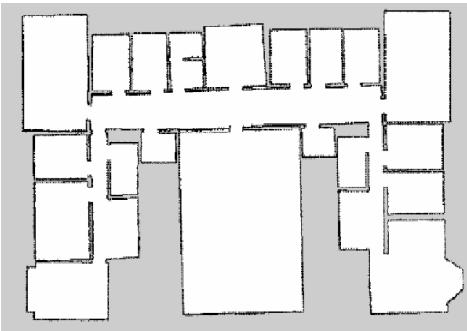
Le Figure che vanno da A.1-A.3, A.4-A.7 e A.8-A.9 rappresentano tre esempi di applicazione del sistema di predizione applicato a mappe metriche parziali acquisite ad istanti di tempo successivi e provenienti da una esplorazione inerente allo stesso ambiente.

Le Figure A.10, A.11 e A.12 rappresentano altri tre esempi di applicazione del metodo di predizione su mappe metriche parziali.

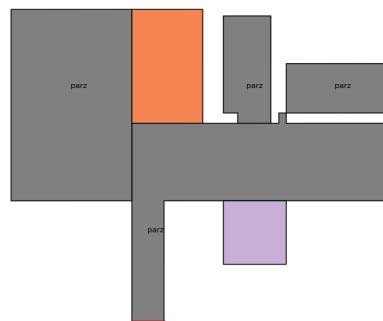
Ciascuna figura è divisa come segue. L'immagine (a) rappresenta la mappa metrica parziale in input al sistema e (b) rappresenta la mappa completa estratta alla fine del processo di esplorazione dell'ambiente ed usata al fine di mostrare visivamente (se confrontata con (a)) la percentuale di completamento dell'esplorazione. L'immagine (c) mostra il layout, facendo distinzione tra le stanze complete (colorate) e stanze parziali (grigie). L'immagine (d) mostra l'azione geometrica 1 ed (e) il risultato finale della predizione al termine dell'azione geometrica complessa. Nell'immagine (f) si mostra il layout predetto sovrapposto alla mappa ground truth.



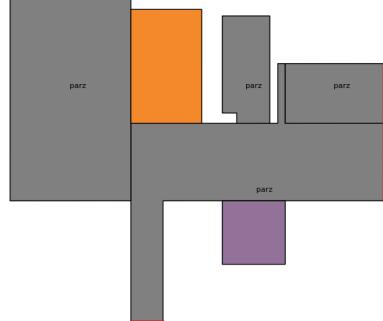
(a) *mappa metrica*



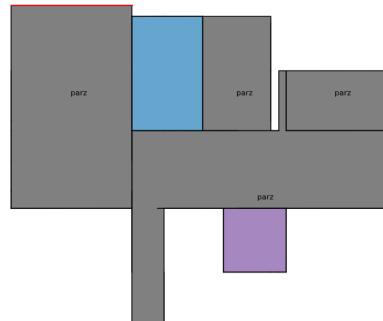
(b) *mappa completa*



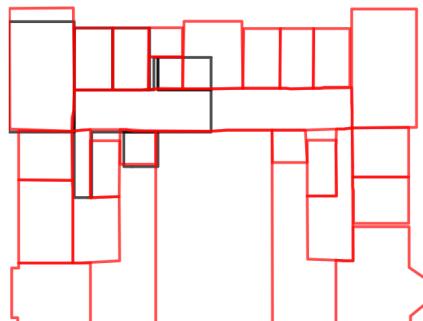
(c) *stanze e stanze parziali*



(d) *azione geometrica 1*

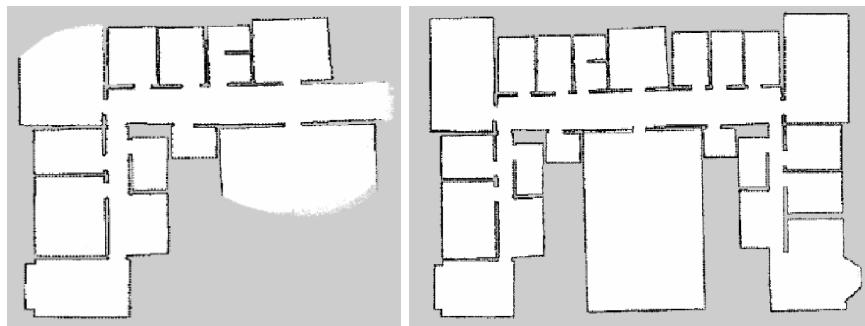


(e) *azione geometrica complessa*



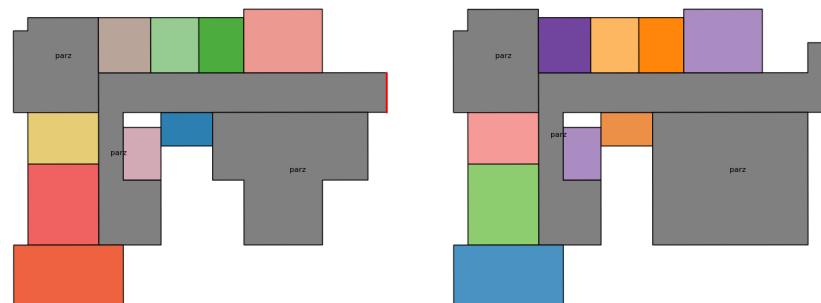
(f) *sovraposizione tra layout e GT*

Figura A.1: Esempio 1. (a) rappresenta la mappa metrica parziale in input. (b) rappresenta la mappa completa dell'ambiente. (c) rappresenta il layout, facendo distinzione tra stanze complete e parziali. (d) rappresenta l'azione geometrica 1 applicata al layout. (e) rappresenta il risultato finale, dopo aver completato l'azione geometrica complessa. (f) rappresenta la sovrapposizione tra il layout predetto e la mappa GT.



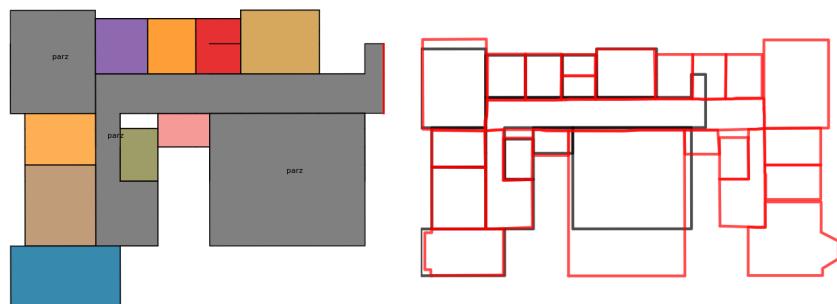
(a) *mappa metrica*

(b) *mappa completa*



(c) *stanze e stanze parziali*

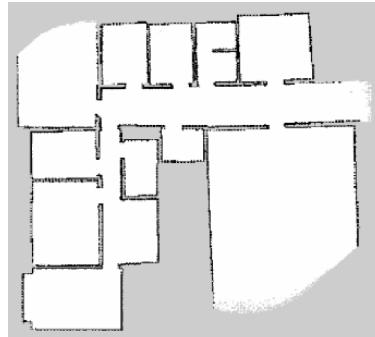
(d) *azione geometrica 1*



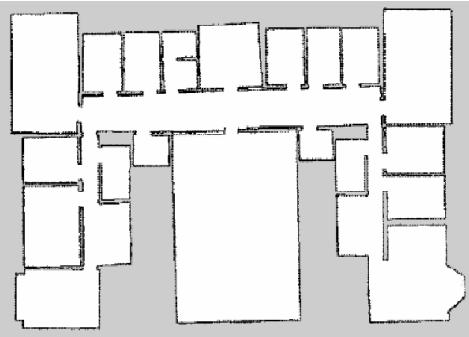
(e) *azione geometrica complessa*

(f) *sovraposizione tra layout e GT*

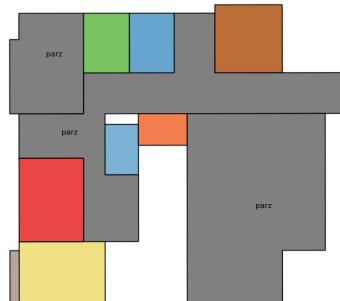
Figura A.2: Esempio 2. (a) rappresenta la mappa metrica parziale in input. (b) rappresenta la mappa completa dell'ambiente. (c) rappresenta il layout, facendo distinzione tra stanze complete e parziali. (d) rappresenta l'azione geometrica 1 applicata al layout. (e) rappresenta il risultato finale, dopo aver completato l'azione geometrica complessa. (f) rappresenta la sovrapposizione tra il layout predetto e la mappa GT.



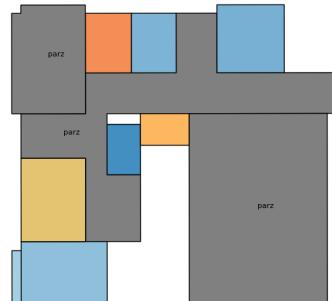
(a) *mappa metrica*



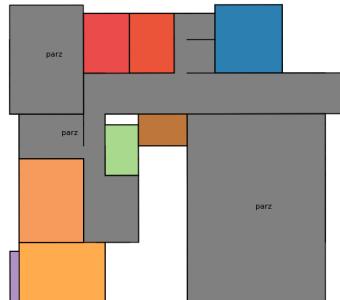
(b) *mappa completa*



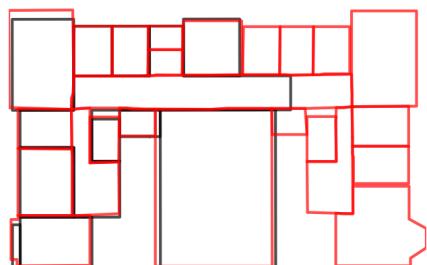
(c) *stanze e stanze parziali*



(d) *azione geometrica 1*

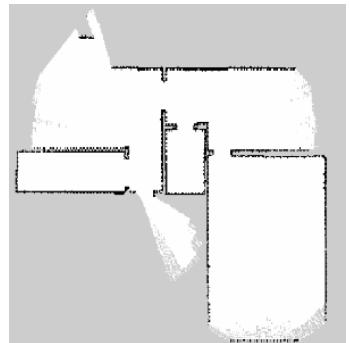


(e) *azione geometrica complessa*

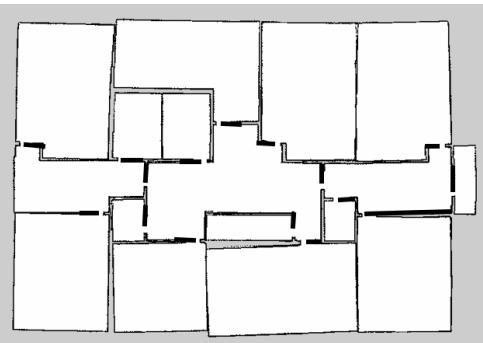


(f) *sovraposizione tra layout e GT*

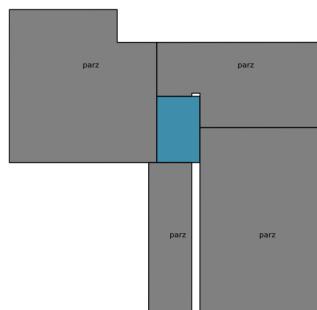
Figura A.3: Esempio 3. (a) rappresenta la mappa metrica parziale in input. (b) rappresenta la mappa completa dell'ambiente. (c) rappresenta il layout, facendo distinzione tra stanze complete e parziali. (d) rappresenta l'azione geometrica 1 applicata al layout. (e) rappresenta il risultato finale, dopo aver completato l'azione geometrica complessa. (f) rappresenta la sovrapposizione tra il layout predetto e la mappa GT.



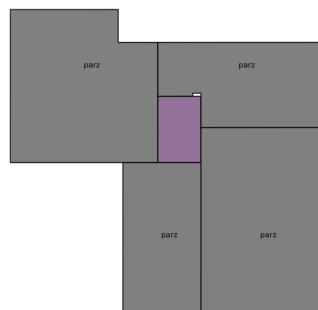
(a) *mappa metrica*



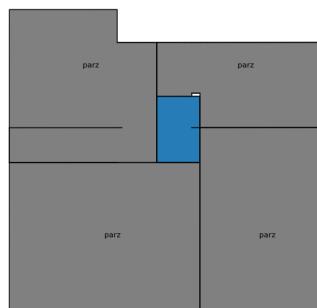
(b) *mappa completa*



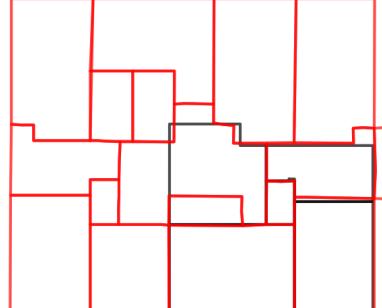
(c) *stanze e stanze parziali*



(d) *azione geometrica 1*

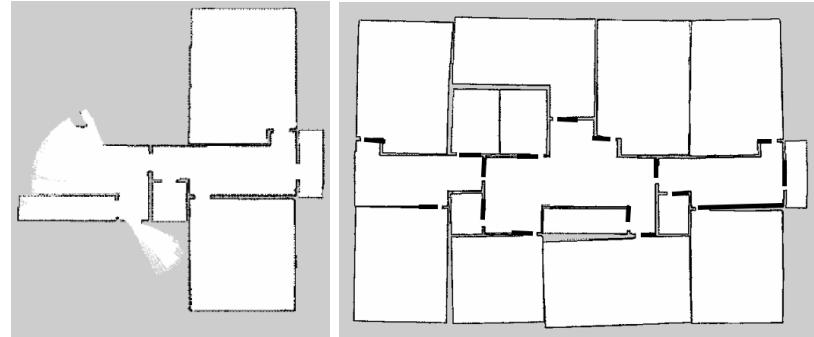


(e) *azione geometrica complessa*



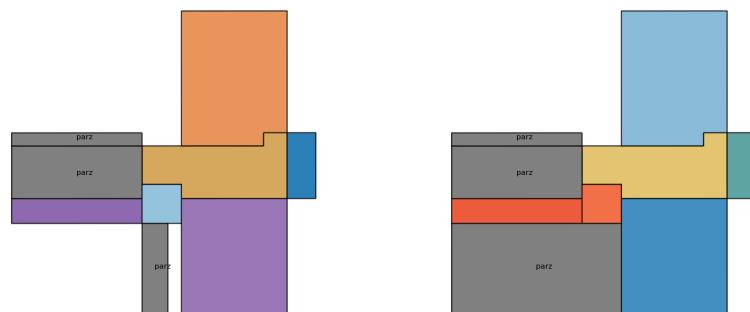
(f) *sovraposizione tra layout e GT*

Figura A.4: Esempio 4. (a) rappresenta la mappa metrica parziale in input. (b) rappresenta la mappa completa dell'ambiente. (c) rappresenta il layout, facendo distinzione tra stanze complete e parziali. (d) rappresenta l'azione geometrica 1 applicata al layout. (e) rappresenta il risultato finale, dopo aver completato l'azione geometrica complessa. (f) rappresenta la sovrapposizione tra il layout predetto e la mappa GT.



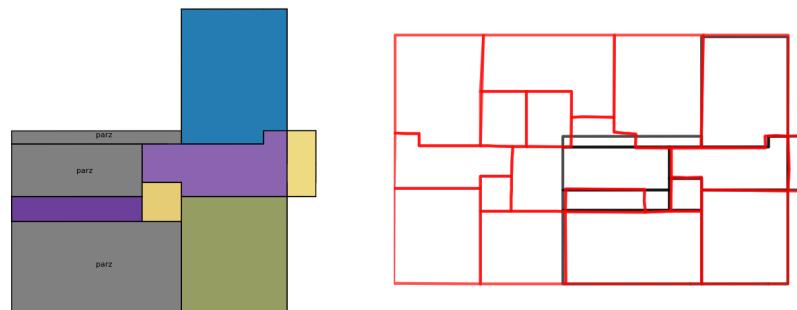
(a) *mappa metrica*

(b) *mappa completa*



(c) *stanze e stanze parziali*

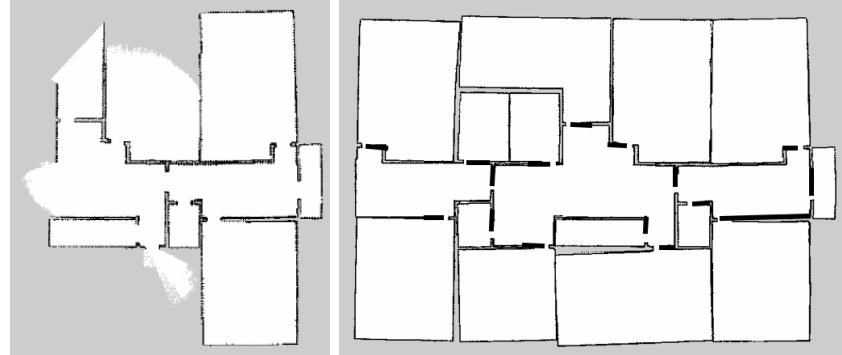
(d) *azione geometrica 1*



(e) *azione geometrica complessa*

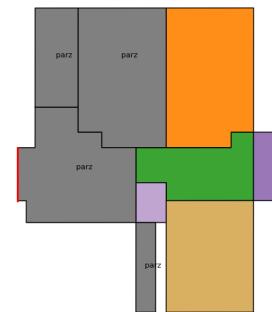
(f) *sovraposizione tra layout e GT*

Figura A.5: Esempio 5. (a) rappresenta la mappa metrica parziale in input. (b) rappresenta la mappa completa dell'ambiente. (c) rappresenta il layout, facendo distinzione tra stanze complete e parziali. (d) rappresenta l'azione geometrica 1 applicata al layout. (e) rappresenta il risultato finale, dopo aver completato l'azione geometrica complessa. (f) rappresenta la sovrapposizione tra il layout predetto e la mappa GT.

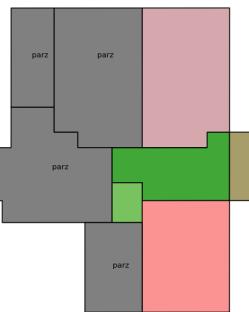


(a) *mappa metrica*

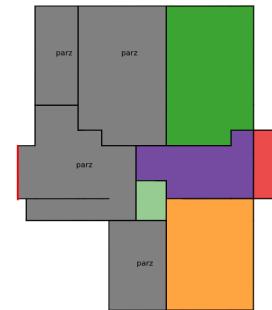
(b) *mappa completa*



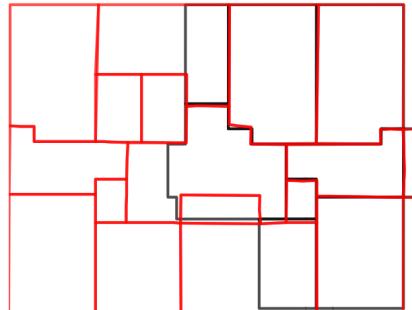
(c) *stanze e stanze parziali*



(d) *azione geometrica 1*

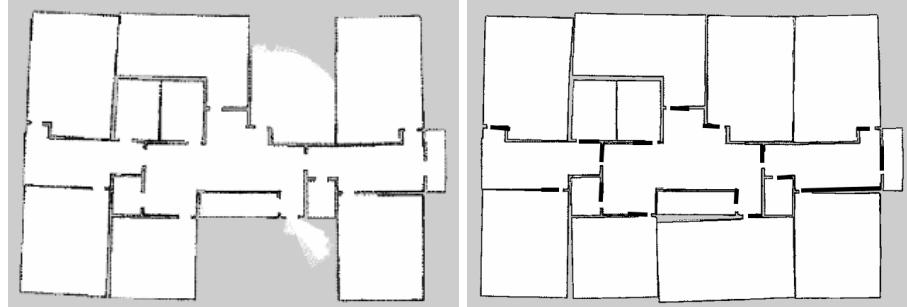


(e) *azione geometrica complessa*



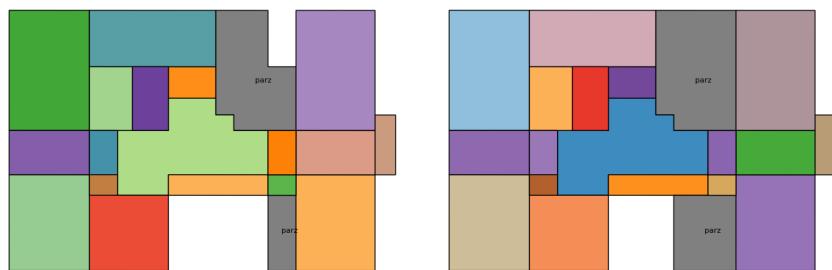
(f) *sovrapposizione tra layout e GT*

Figura A.6: Esempio 6. (a) rappresenta la mappa metrica parziale in input. (b) rappresenta la mappa completa dell'ambiente. (c) rappresenta il layout, facendo distinzione tra stanze complete e parziali. (d) rappresenta l'azione geometrica 1 applicata al layout. (e) rappresenta il risultato finale, dopo aver completato l'azione geometrica complessa. (f) rappresenta la sovrapposizione tra il layout predetto e la mappa GT.



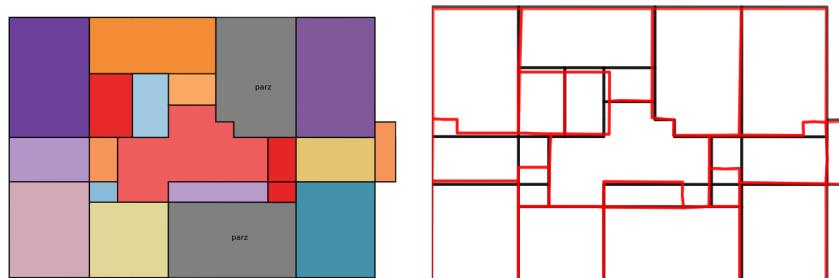
(a) *mappa metrica*

(b) *mappa completa*



(c) *stanze e stanze parziali*

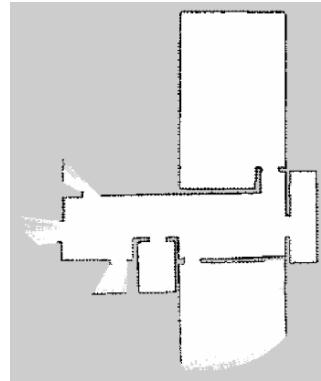
(d) *azione geometrica 1*



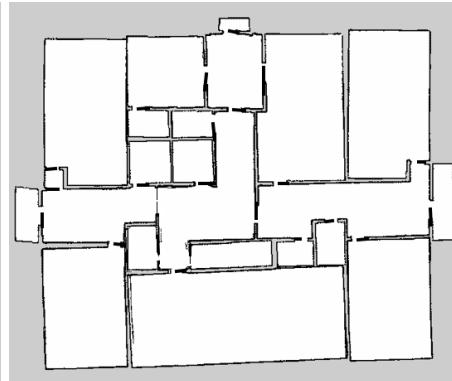
(e) *azione geometrica complessa*

(f) *sovraposizione tra layout e GT*

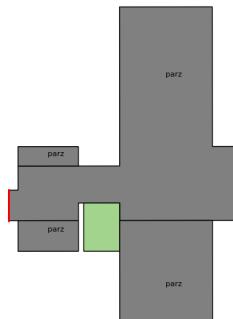
Figura A.7: Esempio 7. (a) rappresenta la mappa metrica parziale in input. (b) rappresenta la mappa completa dell'ambiente. (c) rappresenta il layout, facendo distinzione tra stanze complete e parziali. (d) rappresenta l'azione geometrica 1 applicata al layout. (e) rappresenta il risultato finale, dopo aver completato l'azione geometrica complessa. (f) rappresenta la sovrapposizione tra il layout predetto e la mappa GT.



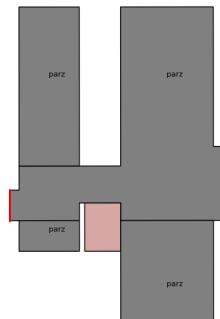
(a) mappa metrica



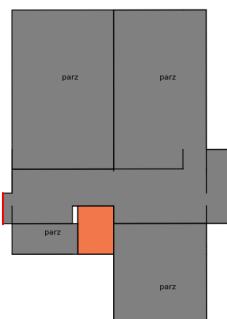
(b) mappa completa



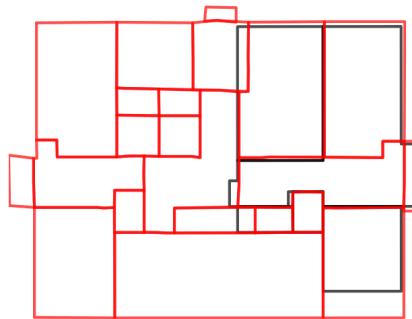
(c) stanze e stanze parziali



(d) azione geometrica 1



(e) azione geometrica complessa



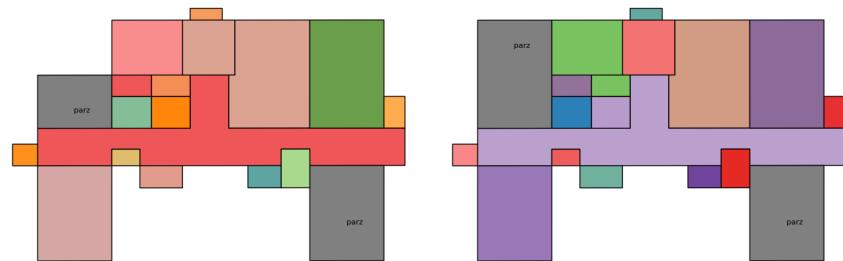
(f) sovrapposizione tra layout e GT

Figura A.8: Esempio 8. (a) rappresenta la mappa metrica parziale in input. (b) rappresenta la mappa completa dell'ambiente. (c) rappresenta il layout, facendo distinzione tra stanze complete e parziali. (d) rappresenta l'azione geometrica 1 applicata al layout. (e) rappresenta il risultato finale, dopo aver completato l'azione geometrica complessa. (f) rappresenta la sovrapposizione tra il layout predetto e la mappa GT.



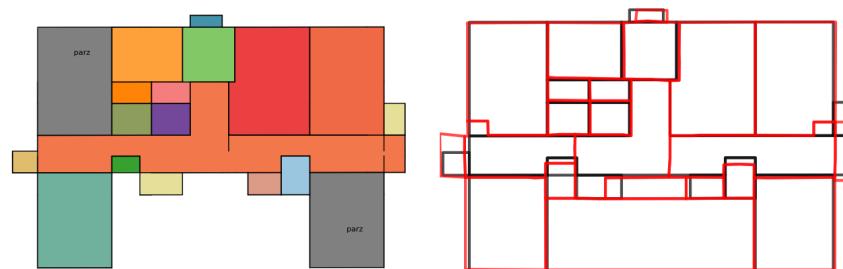
(a) *mappa metrica*

(b) *mappa completa*



(c) *stanze e stanze parziali*

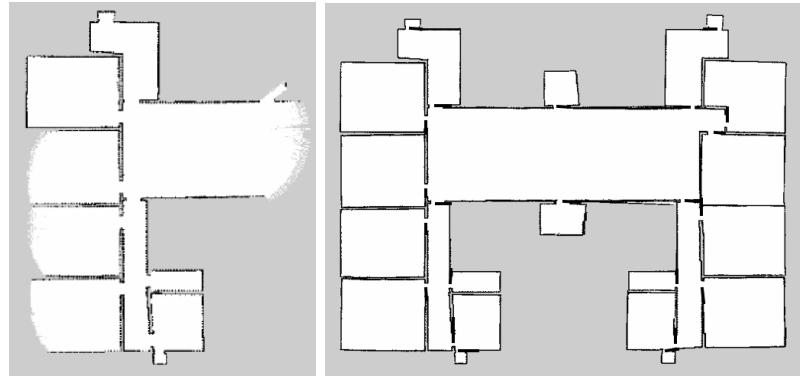
(d) *azione geometrica 1*



(e) *azione geometrica complessa*

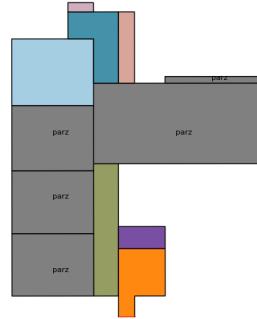
(f) *sovraposizione tra layout e GT*

Figura A.9: Esempio 9. (a) rappresenta la mappa metrica parziale in input. (b) rappresenta la mappa completa dell'ambiente. (c) rappresenta il layout, facendo distinzione tra stanze complete e parziali. (d) rappresenta l'azione geometrica 1 applicata al layout. (e) rappresenta il risultato finale, dopo aver completato l'azione geometrica complessa. (f) rappresenta la sovrapposizione tra il layout predetto e la mappa GT.

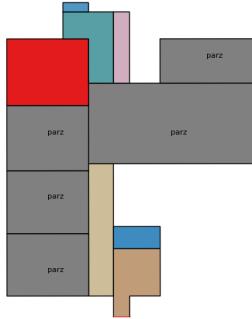


(a) *mappa metrica*

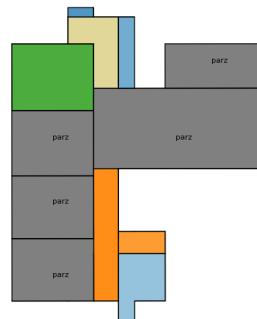
(b) *mappa completa*



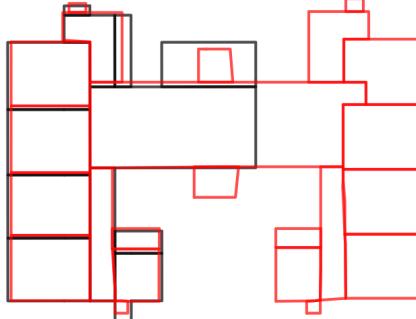
(c) *stanze e stanze parziali*



(d) *azione geometrica 1*



(e) *azione geometrica complessa*



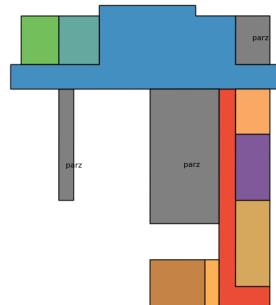
(f) *sovraposizione tra layout e GT*

Figura A.10: Esempio 10. (a) rappresenta la mappa metrica parziale in input. (b) rappresenta la mappa completa dell'ambiente. (c) rappresenta il layout, facendo distinzione tra stanze complete e parziali. (d) rappresenta l'azione geometrica 1 applicata al layout. (e) rappresenta il risultato finale, dopo aver completato l'azione geometrica complessa. (f) rappresenta la sovrapposizione tra il layout predetto e la mappa GT.

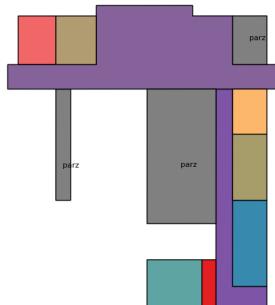


(a) *mappa metrica*

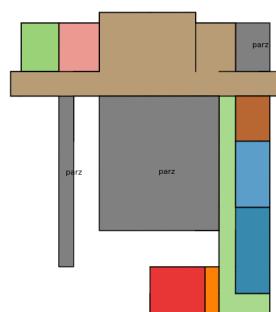
(b) *mappa completa*



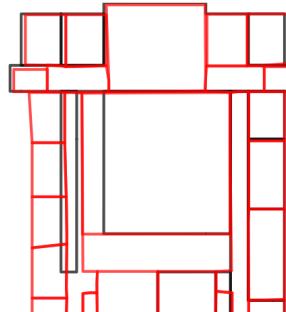
(c) *stanze e stanze parziali*



(d) *azione geometrica 1*



(e) *azione geometrica complessa*



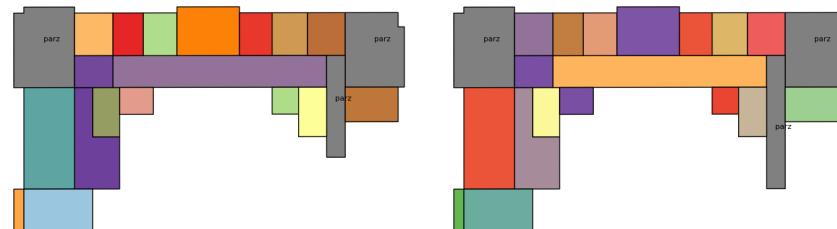
(f) *sovraposizione tra layout e GT*

Figura A.11: Esempio 11. (a) rappresenta la mappa metrica parziale in input. (b) rappresenta la mappa completa dell'ambiente. (c) rappresenta il layout, facendo distinzione tra stanze complete e parziali. (d) rappresenta l'azione geometrica 1 applicata al layout. (e) rappresenta il risultato finale, dopo aver completato l'azione geometrica complessa. (f) rappresenta la sovrapposizione tra il layout predetto e la mappa GT.



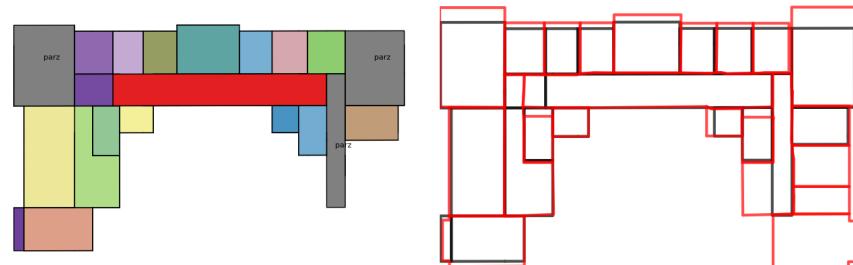
(a) *mappa metrica*

(b) *mappa completa*



(c) *stanze e stanze parziali*

(d) *azione geometrica 1*



(e) *azione geometrica complessa*

(f) *sovraposizione tra layout e GT*

Figura A.12: Esempio 12. (a) rappresenta la mappa metrica parziale in input. (b) rappresenta la mappa completa dell'ambiente. (c) rappresenta il layout, facendo distinzione tra stanze complete e parziali. (d) rappresenta l'azione geometrica 1 applicata al layout. (e) rappresenta il risultato finale, dopo aver completato l'azione geometrica complessa. (f) rappresenta la sovrapposizione tra il layout predetto e la mappa GT.