# Relazione tecnica progetto MOBIDEV

# Integrazione sistema di allineamento automatico in ARL Creator

Paolo Giua

16 maggio 2025

# Indice

# 1 Introduzione

La presente relazione documenta le attività svolte durante il progetto per il corso di **Sviluppo di applicazioni per dispositivi mobili**, tenuto dal **Prof. Sergio Mascetti** presso l'Università degli Studi di Milano. Il lavoro è basato sull'applicazione iOS **ARL Creator**, sviluppata da **Danil Lugli** come parte della sua tesi di laurea magistrale.

## 1.1 Contesto del progetto

**ARL Creator** è un'applicazione per iOS progettata per fornire un'esperienza avanzata di scansione di ambienti 3D utilizzando **RoomPlan API**, **SceneKit** e **RealityKit** di Apple su dispositivi dotati di LiDAR. Questa applicazione consente agli utenti di acquisire e ricostruire un modello 3D dettagliato di uno spazio indoor, che può essere ulteriormente analizzato e personalizzato per varie applicazioni di realtà aumentata (AR).

**ARL Creator** funge da strumento di configurazione per la libreria **MARS**, dove gli utenti vengono localizzati e guidati attraverso gli spazi interni utilizzando le tecnologie AR. Il calcolo della posizione del dispositivo all'interno di ambienti 3D viene svolto utilizzando la configurazione `ARWorldTrackingConfiguration` di ARKit.

## 1.2 Obiettivi specifici dell'aggiornamento

Il progetto descritto in questa relazione ha l'obiettivo di introdurre un nuovo meccanismo di **allineamento automatico tra una stanza mappata e la planimetria di riferimento dell'intero piano**. Questo sistema è pensato per affiancare il metodo di allineamento manuale già integrato nell'applicazione, offrendo un supporto più efficiente e preciso all'utente.

L'intervento si è concentrato sulla progettazione e l'implementazione di un algoritmo capace di confrontare triplette di nodi 3D, determinando la trasformazione rigida ottimale — composta da una **rotazione** attorno all'asse verticale e una **traslazione** nel piano orizzontale — per un allineamento preciso tra due insiemi di punti nello spazio.

L'obiettivo dell'intervento si propone di **semplificare il flusso di lavoro**, automatizzando una funzionalità che altrimenti richiederebbe una configurazione manuale.

Nel seguito della relazione verranno illustrati:

- l'analisi del problema iniziale e delle sue criticità,
- la progettazione e l'implementazione dell'algoritmo di allineamento,
- le strutture dati e i metodi di supporto sviluppati,
- i risultati ottenuti e le possibili evoluzioni future del sistema.

# 2 Analisi del problema

Nel processo di mappatura di un edificio tramite l'app *ARL Creator*, l'utente ha la possibilità di associare a ogni stanza una rappresentazione tridimensionale ottenuta tramite scansione, da posizionare manualmente sopra la planimetria generale del piano. Questo passaggio è fondamentale per garantire la coerenza spaziale tra le singole stanze e l'intero edificio, e risulta cruciale per il corretto funzionamento del sistema di indoor positioning.

## 2.1 Allineamento manuale

Nella versione preesistente dell'app, l'allineamento tra una stanza e la planimetria avveniva esclusivamente in modo manuale. L'utente doveva quindi traslare e ruotare la rappresentazione 3D della stanza fino a ottenere una sovrapposizione coerente rispetto alla posizione reale della stanza sull'edificio.

## 2.2 Limiti della soluzione precedente

Sebbene il posizionamento manuale offra un alto grado di controllo, presenta alcuni limiti importanti:

- È un'operazione soggettiva, influenzata dalla percezione e precisione dell'utente;
- Può richiedere molto tempo per ottenere un risultato soddisfacente.

## 2.3 Requisiti della nuova funzionalità

Alla luce di queste considerazioni, è emersa la necessità di introdurre un sistema automatico per supportare l'allineamento delle stanze. I requisiti principali di questa nuova funzionalità sono:

- Ridurre il tempo richiesto per l'allineamento;
- Fornire un risultato oggettivo, ripetibile e indipendente dall'utente;
- Mantenere la possibilità di intervento manuale, per eventuali correzioni e perfezionamenti da parte dell'utente.

L'obiettivo è quindi quello di rendere il processo di mappatura più efficiente, affidabile e scalabile, senza compromettere la versatilità dell'interfaccia esistente.

# 3 Progettazione della soluzione

La funzionalità di allineamento automatico implementata in *ARL Creator* si basa su una strategia geometrica che mira a determinare la trasformazione (composta da rotazione e traslazione) in grado di sovrapporre nel modo più coerente possibile una stanza rispetto alla planimetria dell'edificio.

## 3.1 Strategia di allineamento

L'idea alla base della strategia è quella di confrontare due insiemi di punti 3D: il primo insieme è costituito da nodi selezionati nella stanza, il secondo da nodi corrispondenti presenti nella planimetria. Ogni nodo rappresenta un oggetto fisico nello spazio. In particolare, sono stati selezionati i nodi di tipo "Door", "Opening" e "Window", in quanto associati a elementi architettonici stabili, la cui posizione rimane fissa nel tempo. Al contrario, oggetti come tavoli o sedie possono essere spostati, risultando meno affidabili per il processo di allineamento.

Per confrontare efficacemente questi due insiemi, è necessario trovare la trasformazione geometrica che applichi rotazione e traslazione al primo insieme in modo tale da farlo coincidere, il più possibile, con il secondo.

## 3.2 Confronto tra triplette di nodi

Per semplificare il problema e renderlo computazionalmente gestibile, la trasformazione viene calcolata a partire da sottogruppi di nodi: in particolare, vengono selezionate tutte le combinazioni possibili di tre nodi per ciascuna configurazione (stanza e planimetria). Ogni tripletta rappresenta un sistema minimo di riferimento, sufficiente per determinare una trasformazione rigida nello spazio 2D.

Tutte le possibili combinazioni di triplette vengono confrontate tra loro: per ciascuna coppia di triplette (una della stanza e una della planimetria), viene inizialmente calcolato un errore di compatibilità. Quanto più questo valore si avvicina allo zero, tanto maggiore è la compatibilità tra le due triplette. L'errore di compatibilità tra due triplette viene calcolato confrontando varie proprietà dei nodi, come la disposizione spaziale, l'altezza, il volume e il tipo di nodo. Ogni confronto produce un errore (in genere un MSE – errore quadratico medio), che viene poi combinato in un unico valore finale tramite pesi configurabili.

In particolare, l'errore di compatibilità somma cinque componenti:

1. **Distanza tra i nodi**: confronta la distanza tra tutte le coppie di nodi nei due cluster.
2. **Altezze relative**: confronta le altezze dei nodi rispetto agli altri all'interno del cluster; utile per capire se la disposizione verticale è simile.
3. **Volumi**: confronta le dimensioni dei nodi; due triplette compatibili dovrebbero avere oggetti di dimensioni simili.
4. **Corrispondenza di tipo**: penalizza i nodi che hanno tipo diverso nella stessa posizione.
5. **Diversità di tipo**: valuta la varietà di tipi delle due triplette, penalizzando triplette troppo eterogenee.

Al termine di questa fase, al fine di ridurre il carico computazionale, vengono selezionate le 1000 coppie di triplette più compatibili. Per ciascuna di esse, viene poi calcolata la trasformazione che allinea i primi tre punti con i corrispondenti secondi tre e viene calcolata la bontà della trasformazione.

La trasformazione cercata è composta da:

- **Rotazione**: un angolo attorno all'asse $y$, che orienta correttamente la stanza rispetto alla planimetria.
- **Traslazione**: uno spostamento lungo gli assi $x$ e $z$, che posiziona la stanza nel punto corretto della planimetria.

Dopo aver calcolato l'errore di compatibilità e l'errore di trasformazione, i due valori vengono sommati per ottenere un errore totale, utilizzato come criterio per selezionare la trasformazione migliore.

## 3.3 Trasformazione Affine 2D Basata sui Centroidi

Per allineare due insiemi di punti tridimensionali, è stato adottato un approccio basato su una trasformazione rigida, che include una rotazione attorno all'asse verticale (asse y) e una successiva traslazione. L'obiettivo è trasformare il *cluster sorgente* in modo che si sovrapponga il più possibile al *cluster target*, mantenendo inalterate le distanze relative tra i punti. Tale metodo, denominato **Trasformazione Affine 2D basata sui centroidi**, è suddiviso in sei passaggi:

1. **Verifica preliminare**: la trasformazione richiede che i due cluster abbiano lo stesso numero di punti e che questo numero sia almeno pari a tre, condizione necessaria per rendere significativa la stima della rotazione nello spazio tridimensionale.
2. **Calcolo dei centroidi**: si calcolano i centroidi geometrici dei due cluster, ovvero i punti medi dei rispettivi gruppi. I centroidi vengono utilizzati per eliminare l'effetto della traslazione iniziale e concentrare l'attenzione sulla componente rotazionale.
3. **Centratura dei punti**: ogni punto di ciascun cluster viene traslato rispetto al proprio centroide. Questo passaggio, noto come *centratura*, consente di riportare entrambi gli insiemi di punti in un sistema di riferimento locale con origine nel centroide, facilitando il confronto diretto tra le due configurazioni.
4. **Stima dell'angolo di rotazione**: per determinare l'**angolo di rotazione** $\theta$ nel piano xz, che meglio allinea i punti centrati del cluster sorgente con quelli del cluster target, si utilizza la funzione `atan2`, applicata a due sommatorie costruite sulle coordinate dei punti centrati:

$$\theta = \text{atan2}\left(\sum(z_s \cdot x_t - x_s \cdot z_t), \sum(x_s \cdot x_t + z_s \cdot z_t)\right)$$

   Questa formula consente di stimare in modo diretto l'angolo che minimizza l'errore quadratico tra i due insiemi di punti, in un contesto simile a quello dell'analisi di Procrustes. L'uso di `atan2` garantisce una stima stabile dell'angolo anche nei casi in cui le coordinate abbiano segni contrastanti o si trovino in quadranti diversi.

5. **Costruzione della matrice di rotazione**: utilizzando l'angolo ottenuto, si costruisce la relativa matrice di rotazione attorno all'asse y, che viene poi applicata a tutti i punti del cluster sorgente.

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

6. **Calcolo della traslazione**: infine, si calcola la traslazione residua necessaria per allineare il centroide del cluster ruotato con il centroide del cluster target. Questo vettore di traslazione, sommato alla rotazione precedentemente calcolata, completa la trasformazione rigida.

### 3.4 Assunzioni semplificative

Per ridurre la complessità del problema, sono state introdotte alcune assunzioni:

- Si considera una **rotazione limitata all'asse y**, coerente con l'ipotesi che le stanze e la planimetria giacciano approssimativamente sullo stesso piano orizzontale (il piano $xz$).
- Non viene considerata alcuna variazione di scala: si assume che le stanze e la planimetria siano rappresentate in una stessa unità di misura.
- La traslazione non viene calcolata per tutte le possibili coppie di triple: si assume che le prime mille coppie più compatibili siano sufficienti per individuare la trasformazione ottimale.

Questa progettazione consente di ottenere un buon compromesso tra efficienza e accuratezza, mantenendo al tempo stesso la flessibilità necessaria per operare in casi realistici.

## 4 Implementazione

La funzionalità di allineamento automatico in *ARL Creator* è stata sviluppata attorno a due strutture fondamentali: `NodeCluster` e `AutoPositionUtility`.

### 4.1 Rappresentazione dei gruppi di nodi: `NodeCluster`

Per rappresentare in modo strutturato i gruppi (o cluster) di nodi da confrontare, è stata introdotta la struttura dati `NodeCluster`. Questa consente di organizzare e gestire i nodi in maniera strutturata ed efficiente, facilitando il calcolo della compatibilità tra insiemi differenti.

A tal fine, ogni oggetto `NodeCluster` include le seguenti proprietà:

- `centroid`: il punto medio dei nodi nello spazio tridimensionale (di tipo `simd_float3`).
- `nodes`: la lista dei nodi ordinata in modo crescente in base alla distanza dal centroide (di tipo `[SCNNode]`).
- `points`: le coordinate spaziali dei singoli nodi (di tipo `[simd_float3]`).
- `distanceMatrix`: la matrice delle distanze tra ciascun nodo e il centroide (di tipo `[[Float]]`).
  - ogni cella $(i, j)$ della matrice rappresenta la distanza tra il nodo $i$-esimo e il nodo $j$-esimo. L'ultima riga e l'ultima colonna contengono le distanze tra ciascun nodo e il centroide. Di conseguenza, la diagonale principale è composta da soli 0, poiché la distanza di un nodo con sé stesso è nulla.
- `relativeHeights`: l'altezza relativa di ogni punto rispetto agli altri del cluster (di tipo `[Float]`).
  - calcolata come la differenza tra la coordinata y del punto considerato e quella del punto più basso.
- `volumes`: il volume del bounding box di ciascun nodo (di tipo `[Float]`).

- `types`: il tipo (`"Door"`, `"Opening"` o `"Window"`) assegnato a ciascun nodo (di tipo `[String?]`).

La struct `NodeCluster` mette a disposizione un insieme di metodi che permettono di confrontare due gruppi di nodi tra loro, restituendo un errore numerico che ne quantifica la somiglianza geometrica e semantica. Il metodo principale è `computeCompatibilityError`, che aggrega cinque metriche parziali, ciascuna con un peso configurabile:

- **Errore sulle distanze tra nodi (`computeDistanceMSE`)**: confronta la matrice delle distanze interne ai cluster, ignorando differenze inferiori a una soglia. La metrica considera solo le distanze tra coppie diverse di nodi.
- **Errore sulle altezze relative (`computeRelativeHeightMSE`)**: valuta lo scostamento tra le posizioni verticali dei nodi nei due cluster, in maniera relativa rispetto al proprio insieme.
- **Errore sui volumi (`computeVolumeMSE`)**: confronta i volumi geometrici dei nodi corrispondenti. Le differenze minime sono trascurate.
- **Errore di corrispondenza di tipo (`computeTypeMismatchMSE`)**: misura il disallineamento tra i tipi associati ai nodi. Per ogni coppia di nodi (uno per cluster), se i tipi non coincidono, viene aggiunto `1.0` all'errore, altrimenti `0.0`. L'MSE finale è la media di questi valori.
- **Errore di diversità di tipo (`computeTypeDiversityMSE`)**: considera l'insieme dei tipi presenti in ciascun cluster e ne misura la diversità complessiva, penalizzando i cluster con caratteristiche troppo eterogenee tra loro.

Il valore finale restituito da `computeCompatibilityError` è una somma pesata delle cinque componenti sopra elencate (il valore predefinito dei pesi è 1.0), e rappresenta una stima complessiva del grado di compatibilità tra i due cluster. Tale valore è fondamentale per selezionare, tra le varie combinazioni, quella che meglio approssima l'allineamento ottimale tra nodi mappati e nodi del piano.

Ogni nodo viene confrontato con la propria controparte corrispondente, grazie all'ordinamento preliminare effettuato in fase di inizializzazione. Tale ordinamento si basa sulla distanza di ciascun nodo dal centroide del cluster, rendendo possibile un confronto coerente tra i nodi. Questo non è un metodo infallibile, poiché ad esempio due nodi potrebbero avere la stessa distanza dal centroide, ma rappresenta una buona approssimazione e permette di evitare il confronto con tutte le permutazioni possibili.

## 4.2 Ricerca della trasformazione ottimale: `AutoPositionUtility`

La struct `AutoPositionUtility` fornisce un insieme di funzioni per calcolare automaticamente l'allineamento tra due gruppi di nodi 3D (`SCNNode`) al fine di stimare la trasformazione (rotazione e traslazione) necessaria per sovrapporre al meglio due insiemi di punti differenti.

La struct è interamente composta da proprietà e metodi statici. L'unica proprietà, `targetTypes`, è un array di stringhe (`[String]`) che definisce i tipi di nodi da includere nel processo di allineamento. I metodi disponibili sono i seguenti:

- `findBestAlignment(from:to:clusterSize:maxPairs:)` – questo metodo rappresenta il punto di ingresso principale per il processo di allineamento automatico. A partire da due liste di nodi, seleziona solo quelli appartenenti ai tipi desiderati (`Door`, `Window`, `Opening`), genera cluster compatibili e calcola, per ciascuna coppia, la trasformazione ottimale (rotazione attorno all'asse Y e traslazione). La coppia con l'errore totale minimo viene scelta come miglior allineamento. Restituisce una tupla contenente:
  - L'angolo di rotazione (in radianti),
  - Il vettore di traslazione (`simd_float3`),
  - L'errore totale associato alla trasformazione selezionata.
- `filterNodesByType(nodes:)` – metodo di utilità interna che filtra un array di nodi mantenendo solo quelli il cui tipo è incluso nella lista `targetTypes`. Viene utilizzato per limitare l'allineamento ai soli nodi significativi dal punto di vista architettonico.

- `findCompatibleClusters(from:and:clusterSize:maxPairs:...)` – data la combinazione di due insiemi di nodi, questo metodo genera cluster di dimensione fissata e li confronta in coppie per calcolare un errore di compatibilità geometrica (tramite il metodo `computeCompatibilityError`). Le migliori `maxPairs` coppie (cioè quelle con errore minore) vengono restituite per essere successivamente valutate in termini di trasformazione. Sono previsti vari limiti (es. `maxNodes`, `maxClusters`) per evitare una crescita combinatoria incontrollata.
- `computeTransformation(from:to:)` – data una coppia di cluster compatibili, calcola la trasformazione rigida (rotazione sull'asse y e traslazione) che allinea il primo al secondo, utilizzando il metodo della **Trasformazione Affine 2D Basata sui Centroidi**. L'angolo di rotazione è stimato in modo ottimale minimizzando la deviazione tra punti ruotati e corrispondenti. Questa operazione presuppone che i due cluster abbiano lo stesso numero di nodi (almeno 3).
- `computeTransformationError(for:rotationAngle:translation:comparedTo:)` – valuta la bontà di una trasformazione precedentemente calcolata applicandola al cluster sorgente e confrontando i punti trasformati con quelli del cluster di destinazione. L'errore è espresso come somma degli errori quadratici tra punti corrispondenti. Questa funzione contribuisce al calcolo dell'errore totale utilizzato per selezionare la trasformazione ottimale.

## 4.3 Estensioni

Per ridurre la ripetizione del codice e rispettare il principio di responsabilità unica, si è scelto di utilizzare le estensioni di iOS al fine di migliorare la chiarezza e la manutenibilità del progetto. In particolare, sono state implementate estensioni per i tipi `SCNNode`, `simd_float3` e `Array`, con l'obiettivo di centralizzare funzionalità ricorrenti e semplificare la logica nei punti di utilizzo.

#### `SCNNode`

Sono stati aggiunti due attributi dinamici ai nodi: `volume` e `type`.
L'attributo `volume` consente di ottenere rapidamente il volume del *bounding box* del nodo, mentre `type` rappresenta il tipo del nodo, determinato a partire dal nome assegnato automaticamente durante la mappatura della stanza.
Entrambi gli attributi vengono utilizzati nel processo di confronto tra nodi per valutarne la compatibilità.

#### `simd_float3`

Nel codice dedicato all'autoallineamento della stanza, il tipo `simd_float3` viene impiegato per rappresentare la posizione tridimensionale dei nodi. Per semplificare il confronto tra i punti, è stato introdotto un metodo che applica una trasformazione, composta da rotazione e traslazione, sul piano xz di un punto nello spazio tridimensionale.

#### `Array`

Sono stati aggiunti metodi agli array per calcolare combinazioni e permutazioni di un insieme di elementi. Inoltre, sono stati introdotti metodi specifici in base al tipo di contenuto dell'array.

In particolare, per le liste di `SCNNode`, si è spesso reso necessario convertire i nodi in una lista di punti. Per semplificare questa operazione, è stato introdotto l'attributo dinamico `simdWorldPositions`, che consente di ottenere direttamente le posizioni mondiali dei nodi in formato SIMD.

Per gli array di `simd_float3` sono stati introdotti metodi specifici per analizzare gruppi di punti. In particolare, sono stati implementati metodi per il calcolo del centroide, della matrice delle distanze e delle altezze relative all'interno di un gruppo. Queste elaborazioni sono fondamentali per valutare la compatibilità tra diversi gruppi di nodi, ad esempio durante il confronto o l'allineamento tra insiemi di punti nello spazio.

# 5 Risultati e validazione

Per valutare l'efficacia del sistema di allineamento automatico implementato in *ARL Creator*, sono stati condotti diversi test su stanze reali già mappate, confrontando i risultati ottenuti con il sistema manuale precedentemente utilizzato.

## 5.1 Casi di test eseguiti

Sono stati condotti test su stanze e planimetrie con caratteristiche diverse per dimensioni e geometria, al fine di valutare la flessibilità e l'affidabilità dell'algoritmo in scenari eterogenei. Per ciascun test, l'algoritmo è stato eseguito utilizzando diverse configurazioni di parametri, variando ad esempio la dimensione dei cluster, i limiti sul numero di nodi e combinazioni considerati, i valori soglia e i pesi impiegati nel calcolo della compatibilità tra configurazioni. Questa analisi ha permesso di comprendere come ciascun parametro influisca sull'accuratezza dell'allineamento finale, sulla robustezza del sistema in presenza di rumore o imprecisioni nella mappatura, e di guidare la scelta dei valori predefiniti più adatti a un utilizzo pratico.

## 5.2 Precisione e affidabilità dell'allineamento automatico

I test finali hanno mostrato una buona precisione dell'allineamento automatico nella maggior parte dei casi. In quasi tutti gli scenari esaminati, l'algoritmo è stato in grado di individuare una trasformazione coerente tra la stanza e la planimetria, restituendo un posizionamento che si avvicina molto a quello ottenuto manualmente. Questo conferma che la nuova funzionalità può ridurre significativamente il tempo richiesto per il posizionamento, lasciando comunque spazio all'intervento dell'utente per perfezionare il risultato.

Un solo caso ha evidenziato un errore significativo: la stanza in questione presentava punti interni allineati ed equidistanti, una configurazione che ha generato ambiguità nel calcolo della trasformazione ottimale, portando a una rotazione di 180° rispetto alla posizione corretta. Si è notato come questo problema sia stato risolto aumentando la dimensione dei cluster da 3 a 5 nodi. In generale, questo aumento ha permesso un miglioramento della precisione dell'algoritmo nella maggior parte dei casi. Tuttavia, ciò ha comportato anche un aumento del numero di situazioni in cui l'algoritmo non è applicabile, a cause del fatto che alcune stanze contengono meno di 5 nodi. Inoltre, la crescita della dimensione dei cluster ha causato un incremento significativo del tempo di calcolo, rendendo meno efficiente l'intero processo.

# 6 Conclusioni

Il sistema di allineamento automatico implementato ha mostrato un buon livello di precisione e affidabilità nella maggior parte dei test. La possibilità di automatizzare il processo di allineamento delle stanze rappresenta un significativo passo avanti rispetto alla versione manuale, riducendo i tempi di intervento dell'utente e migliorando l'efficienza complessiva del flusso di lavoro.

## 6.1 Eventuali limiti riscontrati

Tuttavia, il sistema non è privo di limiti. In alcuni casi, la qualità e la coerenza delle scansioni delle stanze e dei piani, potrebbe compromettere il posizionamento automatico. In altre situazioni particolari, come nel caso citato della stanza con nodi equidistanti e allineati, l'algoritmo può produrre risultati imprecisi. Sebbene questo problema sia stato mitigato aumentando la dimensione dei cluster, questo ha introdotto un aumento dei tempi di calcolo e una maggiore difficoltà nell'applicare l'algoritmo a stanze con pochi nodi. Per questo motivo, la configurazione basata su cluster di 3 nodi si è rivelata la scelta più bilanciata tra precisione e applicabilità. Pur offrendo una buona accuratezza, garantisce tempi di calcolo contenuti e una maggiore compatibilità con un ampio numero di stanze, motivo per cui è stata adottata come impostazione predefinita.

## 6.2 Possibili miglioramenti futuri

Per il futuro, ci sono diverse aree di miglioramento che potrebbero contribuire a perfezionare ulteriormente la funzionalità.

Per superare i limiti che si hanno con cluster di 5 nodi, un possibile sviluppo futuro potrebbe riguardare la progettazione di un sistema più dinamico per la selezione delle coppie dei gruppi per il calcolo della trasformazione. Invece di fissare a priori la dimensione dei cluster da confrontare, si potrebbe cercare di ottenere gruppi della massima grandezza possibile attraverso una procedura incrementale guidata dalla compatibilità. L'idea è che gruppi compatibili siano composti da sottogruppi compatibili: partendo da una coppia di gruppi di piccola dimensione che presentano una buona compatibilità, si potrebbe tentare di espandere i due gruppi aggiungendo progressivamente a ciascun cluster un nuovo nodo, selezionandolo tra quelli che non peggiorano (o addirittura migliorano) la compatibilità. Questo processo iterativo si arresterebbe nel momento in cui l'aggiunta di ulteriori nodi comporta un degrado della compatibilità.

Un approccio di questo tipo avrebbe il vantaggio di evitare il confronto esaustivo tra tutte le possibili combinazioni di nodi di una certa dimensione dei due gruppi, come avviene nella versione attuale, e permetterebbe una ricerca più efficiente e mirata della migliore coppia di cluster. Inoltre, permetterebbe di sfruttare al meglio i dati a disposizione, con un potenziale miglioramento sia in termini di precisione che di performance.

Un'altra ottimizzazione potrebbe considerare l'utilizzo degli angoli delle stanze come ulteriori riferimenti per migliorare la precisione dell'allineamento. Inoltre, sarebbe utile introdurre un filtro per escludere automaticamente le trasformazioni che collocano la stanza in posizioni non plausibili rispetto al piano, ad esempio valutando l'area di intersezione tra i due elementi.

Nonostante il matching 3D possa sembrare una naturale estensione del sistema, non si rivela necessario nel contesto attuale, in quanto il confronto tra la planimetria del piano e le stanze avviene all'interno dello stesso sistema di riferimento e con la stessa unità di misura. L'analisi può quindi essere efficacemente condotta in due dimensioni, riducendo significativamente la complessità computazionale rispetto a un approccio tridimensionale, che risulterebbe più oneroso senza offrire benefici concreti nel contesto specifico dell'applicazione.