Politecnico di Torino

Computer Engineering

A.a. 2025/2026

Sessione di laurea Marzo 2026

UiVLA: Vision–Language–Action Model per l'Adattamento delle Interfacce Utente in Mixed Reality

Relatori:

Luigi De Russis

Per Ola Kristensson

Candidato:

Giacomo Ponzuoli

Sommario

Le interfacce utente in *Mixed Reality* (MR) offrono nuove modalità di interazione, ma la loro integrazione nello spazio reale introduce criticità legate a visibilità, interferenza con il compito e sicurezza. Le soluzioni attuali per il posizionamento degli elementi virtuali si basano spesso su regole statiche, senza considerare in modo esplicito il contesto visivo e l'attenzione dell'utente.

Questa tesi propone *UiVLA*, un sistema *end-to-end* basato su un *Vision–Language Model* (VLM), nello specifico *Qwen2.5-VL*, esteso in una pipeline *vision–language–action* per l'adattamento contestuale di interfacce MR. Il sistema è in grado di comprendere il contesto semantico della scena osservata al fine di produrre decisioni operative sulla UI. In particolare, affronta due compiti principali: la valutazione della *visibility* (mostrare o nascondere un overlay) e la scelta del *placement* più appropriato tra posizioni candidate.

Il lavoro si conclude con l'integrazione del modello su **HoloLens 2**, dimostrando la fattibilità di un sistema *context-aware* potenzialmente *real-time* per l'adattamento dinamico dell'interfaccia in scenari reali.

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Introduzione

Le interfacce utente per la *Mixed Reality* (MR) stanno diventando sempre più diffuse, offrendo nuove possibilità per esperienze altamente immersive. Con l'aumento della complessità e dell'adozione di queste interfacce, emerge la necessità di sviluppare sistemi capaci di adattarsi in modo intelligente e sensibile al contesto d'uso. Tale adattamento è fondamentale per garantire un'esperienza d'uso fluida, accessibile e usabile in ambienti e situazioni eterogenee.

Dal punto di vista della *Human–Computer Interaction* (HCI), le interfacce MR pongono sfide particolarmente rilevanti, in quanto l'interazione avviene in ambienti fisici reali, dinamici e spesso imprevedibili. A differenza delle interfacce tradizionali, gli elementi della UI in MR condividono lo spazio percettivo con il mondo reale, competendo per l'attenzione dell'utente e influenzando direttamente fattori critici come il carico cognitivo, la sicurezza e la continuità dell'esperienza. In questo contesto, un adattamento inadeguato dell'interfaccia può compromettere non solo l'usabilità, ma anche l'efficacia e l'accettabilità del sistema da parte dell'utente.

Nonostante i progressi recenti, l'adattamento contestuale delle interfacce in MR rimane un problema aperto e particolarmente complesso. Diversi fattori possono influenzare in modo significativo l'esperienza dell'utente, tra cui:

1. la natura e l'aspetto dell'ambiente fisico, con cui l'interfaccia deve integrarsi preservando la continuità dell'esperienza

2. il compito che l'utente sta svolgendo in un dato momento

3. le capacità e le caratteristiche individuali dell'utente

Ciascuno di questi elementi introduce variabilità e incertezza contestuale, rendendo difficile affidarsi a metodi progettuali tradizionali basati su regole o euristiche

definite a mano. Tali approcci, tipicamente sviluppati nell'ambito della progettazione HCI classica e adottati in diversi sistemi di adattamento contestuale per la Mixed Reality, risultano spesso fragili in condizioni *out-of-distribution*, ovvero quando il contesto reale differisce significativamente dalle condizioni, dai dati o dalle assunzioni per cui il sistema è stato originariamente progettato o ottimizzato [1, 2].

Gli approcci convenzionali per la progettazione di interfacce MR adattive affrontano tipicamente il problema in termini di ottimizzazione: a partire dalle osservazioni sul contesto, l'obiettivo è individuare i parametri di design ottimali — ad esempio visibilità, posizionamento e aspetto degli elementi dell'interfaccia. In molti lavori di area HCI, vincoli e obiettivi vengono definiti in modo esplicito, matematico e sostanzialmente statico [1, 2]. Sebbene tali approcci siano efficaci in scenari semplici e ben definiti, mostrano limiti evidenti quando l'ambiente è complesso, dinamico o mai incontrato prima, e quando l'adattamento richiede una comprensione più profonda e sensibile della scena, come richiesto da molte applicazioni reali di Mixed Reality.

Per affrontare queste sfide, con questa tesi proponiamo *UiVLA*, un *proof-of-concept* basato su un modello *vision–language–action* (VLA) [3] per l'adattamento delle interfacce utente. UiVLA mappa le osservazioni dell'ambiente dell'utente, acquisite tramite un visore MR, in azioni sulla UI. Nel presente lavoro, l'adattamento dell'interfaccia viene formulato limitando lo spazio delle azioni a due compiti fondamentali:

1. **Visibility task**: decidere se mostrare o nascondere un elemento dell'interfaccia
2. **Placement task**: scegliere la posizione più adatta tra un insieme finito di alternative

Questa formulazione consente di interpretare l'adattamento dell'interfaccia come un processo decisionale guidato dal contesto, in cui le scelte del modello hanno un impatto diretto sull'esperienza dell'utente, in linea con i principi della HCI orientata all'utente e al contesto.

UiVLA nasce dall'ispirazione dei recenti progressi nel campo della robotica, dove i *vision–language models* (VLMs) [4, 5] hanno dimostrato una notevole capacità di generalizzazione *zero-shot* e *few-shot*, consentendo ai robot di eseguire nuove attività e adattarsi a scenari non visti. Un esempio rilevante è il modello RT-2 di Google DeepMind [6], che introduce un paradigma VLA in cui un *backbone* VLM è *fine-tuned* su dati di azioni robotiche rappresentate come token testuali. Questo approccio sfrutta la capacità dei VLM di riconoscere pattern visivi e linguistici complessi, come dettagli fini, caratteristiche degli oggetti e relazioni spaziali [5]. Nel presente lavoro, adottiamo tale paradigma al fine di sfruttare le capacità di generalizzazione e comprensione multimodale offerte dai Vision–Language Models, estendendole al dominio decisionale dell'adattamento delle interfacce MR.

I contributi principali di questa tesi sono i seguenti:

1. la realizzazione e condivisione di un dataset per l'addestramento e la valutazione di modelli per l'adattamento delle interfacce utente in MR

2. l'introduzione di UiVLA come prima pipeline end-to-end per l'adattamento di interfacce utente in MR

3. uno studio sulle principali scelte progettuali tramite *ablation study*, analizzando la risoluzione delle patch e il *fine-tuning* tramite LoRA

4. l'integrazione del modello ottenuto all'interno di applicazioni reali basate sul visore *HoloLens 2*

Questo lavoro di tesi è stato svolto nell'ambito di un progetto di ricerca condotto in collaborazione con il Dipartimento di Computer Science dell'Università di Cambridge, durante un periodo di ricerca di sei mesi (settembre 2025 – febbraio 2026), sotto la supervisione del Prof. Per Ola Kristensson e del Prof. John Dudley.

1.2 Obiettivo

L'obiettivo di questa tesi è duplice: da un lato investigare, in chiave scientifica, la fattibilità del paradigma vision–language–action per l'adattamento delle interfacce utente in Mixed Reality; dall'altro progettare e implementare, in chiave ingegneristica, una pipeline end-to-end funzionante su dispositivo reale.

A differenza degli approcci modulari tradizionali, in cui il processo di adattamento è suddiviso in componenti indipendenti (ad esempio object detection, stima della scena, decisione finale), la proposta mira a ridurre la frammentazione architetturale integrando percezione e decisione all'interno di un unico modello multimodale. Tale scelta intende limitare la propagazione di errori tra moduli distinti, fenomeno tipico delle pipeline composte da sottosistemi ciascuno con affidabilità parziale.

L'obiettivo generale si articola nei seguenti sotto-obiettivi:

1. definire una formulazione del problema di adattamento dell'interfaccia come task decisionale, modellando esplicitamente le azioni sulla UI (visibility e placement)

2. costruire un dataset dedicato che rappresenti scenari realistici di utilizzo in Mixed Reality, includendo variabilità ambientale e contestuale

3. adattare e fine-tunare un modello vision–language per produrre azioni sull'interfaccia sotto forma di token testuali, analizzando sperimentalmente l'impatto di scelte progettuali chiave, quali la risoluzione delle patch visive e l'impiego di tecniche di fine-tuning parametrico efficiente

4. integrare il modello all'interno di un'applicazione reale eseguita su visore *HoloLens 2*, verificandone il funzionamento in scenari concreti e sotto vincoli reali di latenza e risorse computazionali

La tesi non si propone di dimostrare, tramite user study comparativi, la superiorità del sistema rispetto ad approcci esistenti, bensì di esplorare e validare la possibilità di un adattamento dell'interfaccia interamente guidato da un modello multimodale end-to-end, basato sull'impiego di Vision–Language Models, riducendo la dipendenza da componenti euristici separati e proponendo una formulazione unificata del problema di adattamento.

1.3 Struttura della tesi

La tesi è organizzata in 11 capitoli, ognuno dei quali affronta un aspetto specifico del lavoro svolto, dalla definizione del problema alla validazione sperimentale del modello proposto.

- **Capitolo 1** introduce il contesto della Mixed Reality e le principali sfide legate all'adattamento contestuale delle interfacce utente. Il capitolo presenta il modello *UiVLA*, i task di *visibility* e *placement*, e riassume gli obiettivi e i contributi della tesi, illustrando inoltre l'evoluzione del lavoro a seguito dell'analisi critica della pipeline sperimentale di partenza

- **Capitolo 2** fornisce il quadro teorico di riferimento della tesi. Il capitolo analizza gli approcci esistenti all'adattamento delle interfacce utente in Mixed Reality, sia basati su ottimizzazione sia su modelli *vision–language*, introducendo successivamente i fondamenti dei modelli *vision–language* e *vision–language–action*. Infine, viene presentata in dettaglio l'architettura del backbone adottato, *Qwen2.5-VL*, che costituisce la base tecnologica di UiVLA

- **Capitolo 3** introduce *UiVLA*, una pipeline *Vision–Language–Action* end-to-end progettata per l'adattamento dinamico delle interfacce utente in Mixed Reality. Il capitolo presenta la motivazione del sistema e ne descrive l'architettura concettuale basata su un ciclo iterativo di percezione, decisione e azione. Viene quindi illustrata la pipeline decisionale che governa i task di *placement* e *visibility*, discutendo le implicazioni progettuali per il modello e il passaggio da un generico *Vision–Language Model* a un modello *Vision–Language–Action* specializzato per la generazione di decisioni operative sull'interfaccia

- **Capitolo 4** formalizza i due task centrali della tesi, *visibility* e *placement*, che modellano il problema dell'adattamento contestuale delle interfacce in Mixed Reality. Il capitolo introduce il contesto HCI di riferimento e definisce il task

di visibility come una decisione binaria sull'accettabilità di una sovrapposizione, arricchita da una spiegazione testuale interpretabile. Successivamente, il framework viene esteso al task di placement, formulato come un problema discreto ispirato al paradigma *Set-of-Mark (SoM)* per la selezione della posizione ottimale dell'overlay

- **Capitolo 5** descrive la pipeline di *pre-processing* e costruzione della rappresentazione contestuale. Il capitolo presenta l'acquisizione e la sincronizzazione dei frame con le misure di *eye-gaze*, successivamente introduce le mappe pixel-wise di *aesthetic*, *functionality* e *safety/social*. Viene inoltre integrata la stima di profondità monoculare per modulare il contributo di contenuti lontani, discutendo le scelte di soglia e l'approccio percentile-based. Infine, le mappe vengono aggregate in una *combined map*, usata come base comune per la label di *visibility* e lo scoring nel task di *placement*

- **Capitolo 6** descrive la costruzione del dataset supervisionato per i task di *visibility* e *placement*, utilizzando la *combined map* come segnale unificato per generare automaticamente le istanze di training. Per *visibility* vengono derivate label binarie e *reason* testuali deterministiche dalle mappe contestuali; per *placement* viene introdotta la formulazione discreta *Set-of-Mark (SoM)* e vengono confrontate diverse strategie per la generazione delle posizioni candidate, motivando la scelta finale basata su *Mask2Former*

- **Capitolo 7** presenta la valutazione sperimentale di Qwen2.5-VL sui task di *visibility* e *placement*. Il capitolo descrive il setup sperimentale comune, le metriche adottate e il protocollo di training, analizzando in dettaglio i risultati quantitativi e qualitativi. Vengono inoltre condotti *ablation studies* per studiare l'impatto della risoluzione visiva e del LoRA rank, e discusse le implicazioni delle diverse strategie di supervisione

- **Capitolo 8** esplora l'estensione del modello in regime di *Multi-Task Learning (MTL)*, con l'obiettivo di unificare in un unico *Vision–Language–Action model* le competenze di *visibility* e *placement*. Il capitolo motiva l'adozione dell'MTL nel contesto Mixed Reality e ne richiama i principi, descrivendo quindi il fine-tuning di **Qwen2.5-VL** su un dataset ottenuto dall'unione dei due training set. Infine, viene presentato il setup sperimentale e l'analisi dei risultati, discutendo l'impatto di *visual patches* e LoRA rank e valutando il compromesso tra accuratezza, coerenza decisionale ed efficienza rispetto ai setting single-task

- **Capitolo 9** descrive l'integrazione della pipeline *UiVLA* all'interno di un sistema reale basato su *HoloLens 2*, adottando un'architettura distribuita client–server. Il capitolo presenta l'organizzazione generale della pipeline, il meccanismo di rilevamento del cambiamento di contesto tramite sensori di

luminosità e rilocazione spaziale, e l'architettura modulare del client con i componenti *ContextBootstrap*, *ContextManager* e *ContextHttpClient*. Infine, vengono illustrati l'esecuzione remota del modello su server locale, la comunicazione HTTP/REST tra visore e nodo computazionale e la struttura del modulo server sviluppato per supportare i task di *visibility* e *placement*

- **Capitolo 10** presenta i risultati ottenuti dopo l'integrazione della pipeline *UiVLA* su *HoloLens 2*. Il capitolo analizza la latenza della pipeline client–server tramite la metrica `duration_request`, discutendo i tempi di risposta osservati nel sistema reale, e include una valutazione qualitativa del comportamento del modello nei task di *visibility* e *placement*, supportata da esempi di scene reali e da una registrazione video del funzionamento della pipeline
- **Capitolo 11** riassume i principali risultati del lavoro e discute criticamente i punti di forza e le limitazioni della pipeline *UiVLA*. In particolare, viene evidenziata la fattibilità del paradigma *vision–language–action* per l'adattamento della UI in Mixed Reality e la coerenza tra risultati sperimentali e comportamento del sistema in scenari reali. Il capitolo conclude delineando alcune possibili direzioni di ricerca future

Capitolo 2

Background

Questo capitolo inquadra il problema dell'adattamento delle interfacce utente in Mixed Reality all'interno della letteratura esistente, con l'obiettivo di chiarire i principali paradigmi metodologici adottati e i loro limiti strutturali. L'analisi si concentra su come il contesto venga rappresentato e utilizzato nei processi decisionali, e su quali assunzioni progettuali ne derivino in termini di generalizzazione, flessibilità e scalabilità.

In primo luogo vengono introdotti i *Vision–Language Models* (VLM), una classe di modelli multimodali in grado di integrare informazioni visive e linguistiche all'interno di una rappresentazione condivisa. Questi modelli costituiscono una componente tecnologica chiave per lo sviluppo di sistemi capaci di interpretare scene complesse e prendere decisioni contestuali a partire da osservazioni visive.

Successivamente, il capitolo analizza i principali approcci proposti in letteratura per l'adattamento delle interfacce utente in Mixed Reality. In particolare, vengono discussi sia i metodi basati su formulazioni di ottimizzazione esplicita, che modellano l'adattamento come un problema vincolato o probabilistico definito a priori, sia gli approcci più recenti che sfruttano modelli *vision–language* per incorporare informazioni contestuali in modo più flessibile.

2.1 Modelli Vision–Language (VLM)

I *vision–language models* (VLMs) [5, 4] sono modelli multimodali di larga scala progettati per comprendere e generare informazioni che coinvolgono simultaneamente contenuti visivi e testuali. A differenza dei modelli unimodali, che elaborano immagini o linguaggio in modo indipendente, i VLM integrano le due modalità all'interno di una rappresentazione condivisa, consentendo forme di ragionamento congiunto tra ciò che viene osservato visivamente e ciò che viene espresso linguisticamente. Grazie a questa integrazione, i VLM sono in grado di affrontare una

vasta gamma di compiti multimodali, tra cui *image captioning*, *visual question answering* (VQA), riconoscimento di oggetti e scene, classificazione visiva guidata da istruzioni testuali, generazione di descrizioni dettagliate e forme più avanzate di *multimodal reasoning*.

Dal punto di vista architetturale, i VLM seguono tipicamente una struttura composta da un encoder visivo e da un modello linguistico autoregressivo di grandi dimensioni, come illustrato in Figura 2.1. L'encoder visivo, spesso basato su architetture come il Vision Transformer (ViT) [7], ha il compito di estrarre rappresentazioni latenti ad alto livello a partire dall'input visuale. Tali rappresentazioni devono poi essere integrate con un language model, come LLaMA [8], PaLM [9], Qwen [10] o modelli di tipo GPT [11], che opera nello spazio discreto dei token linguistici e genera l'output testuale in modo autoregressivo.

Poiché le rappresentazioni visive e linguistiche risiedono in spazi di embedding profondamente diversi, i VLM impiegano specifici moduli di *bridging multimodale* per rendere compatibili le feature visive con il modello linguistico. Tali moduli svolgono il ruolo di proiettare, rielaborare o comprimere l'informazione visiva in una sequenza di *visual tokens* utilizzabili dal language model. Le soluzioni proposte in letteratura includono semplici proiezioni lineari, architetture dedicate come il Q-Former, o meccanismi di resampling e compressione delle feature visive. Attraverso questi moduli, il modello linguistico può trattare l'informazione proveniente dall'immagine in modo analogo ai token testuali, abilitando un ragionamento realmente multimodale.

Un elemento centrale dell'architettura dei VLM è il modello linguistico autoregressivo, che funge da componente decisionale e generativa del sistema. Un language model autoregressivo apprende a modellare la distribuzione di probabilità di una sequenza di token condizionata sul contesto precedente, generando l'output un token alla volta secondo una fattorizzazione causale della forma

$$p(y_1, \ldots, y_T \mid x) = \prod_{t=1}^{T} p(y_t \mid y_{<t}, x),$$

dove y_t rappresenta il t-esimo token della sequenza di output generata dal modello (ad esempio una parola, un subword token o un simbolo strutturato), $y_{<t}$ indica la sequenza dei token precedentemente generati, e x rappresenta il contesto di input, che nei VLM include sia token testuali sia rappresentazioni derivate dall'input visivo.

Questa formulazione consente al modello di integrare progressivamente l'informazione visiva e linguistica nel processo di generazione, producendo descrizioni, risposte o output strutturati coerenti con il contenuto dell'immagine. Nei VLM moderni, le feature visive vengono incorporate nel contesto autoregressivo come token multimodali, influenzando direttamente la probabilità dei token successivi.

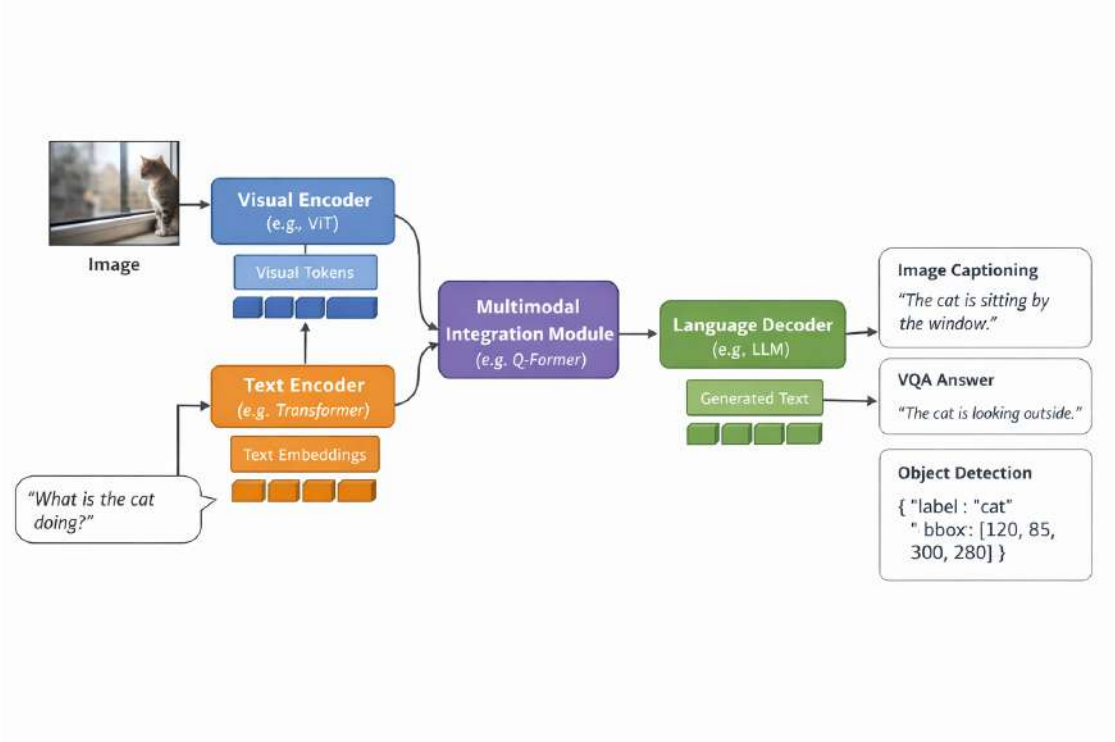


Figura 2.1: Schema architetturale generico di un *Vision–Language Model* (VLM). L'input visuale viene elaborato da un encoder visivo (ad es. ViT) e trasformato in una sequenza di *visual tokens*, che vengono integrati con le rappresentazioni linguistiche tramite un modulo di *bridging multimodale* (ad es. proiezioni lineari o Q-Former). Il language model autoregressivo genera quindi output testuali per diversi compiti multimodali, come image captioning, visual question answering e object detection, espresso sotto forma di output strutturato (ad es. coordinate di bounding box).

Dal punto di vista funzionale, l'autoregressività conferisce ai VLM una notevole flessibilità espressiva: il medesimo modello può essere impiegato per compiti diversi semplicemente variando l'istruzione testuale fornita in input, senza modificare l'architettura o introdurre teste di output specializzate. Questa proprietà risulta particolarmente rilevante nei contesti multimodali, in quanto consente di unificare compiti eterogenei all'interno di un'unica formulazione di generazione sequenziale.

Gli approcci più recenti si orientano prevalentemente verso modelli autoregressivi condizionati visivamente, in cui il language model genera l'output un token alla volta sulla base di un contesto che include sia testo sia rappresentazioni visive. Esempi rappresentativi di questo paradigma sono BLIP-2 [12], LLaVA [13], Flamingo [14],

Qwen-VL [10] e Kosmos-2 [15]. Questi modelli mostrano notevoli capacità di generalizzazione *zero-shot* e *few-shot*, grazie all'addestramento su dataset estremamente ampi e diversificati.

Un aspetto distintivo dei VLM rispetto ai modelli di visione tradizionali è la loro capacità di eseguire *ragionamento multimodale*, ossia combinare informazioni visive e linguistiche per inferire contenuti che non sono direttamente osservabili nell'immagine. Ad esempio, un VLM può dedurre relazioni spaziali implicite, interpretare azioni in corso, riconoscere oggetti in contesti non convenzionali o rispondere a domande complesse che richiedono una comprensione globale della scena. Queste capacità emergenti hanno reso i VLM componenti chiave in ambiti quali la robotica, la Human–Computer Interaction, la generazione di contenuti e i sistemi interattivi intelligenti.

Nonostante i loro punti di forza, i VLM presentano alcune limitazioni strutturali. In primo luogo, la loro percezione visiva non sempre raggiunge il livello di accuratezza dei modelli di visione specializzati addestrati per compiti specifici. Inoltre, possono soffrire di fenomeni di *hallucination* [16], generando output sintatticamente plausibili ma non coerenti con il contenuto visivo. Un'ulteriore criticità riguarda la calibrazione delle risposte: i VLM tendono spesso a produrre predizioni eccessivamente confidenziali anche in presenza di ambiguità percettive o di informazione visiva incompleta [17].

Nel contesto di questa tesi, i VLM rivestono un ruolo centrale in quanto costituiscono il backbone dell'architettura UiVLA. La loro capacità di comprendere simultaneamente immagini e istruzioni linguistiche consente al modello di analizzare il contesto visivo dell'utente e di generare decisioni di adattamento dell'interfaccia in modo più flessibile, generalizzabile e sensibile al contesto rispetto agli approcci tradizionali basati su formulazioni di ottimizzazione esplicita. Questa proprietà rende i VLM particolarmente adatti ad affrontare scenari complessi, dinamici e non visti, tipici delle applicazioni di Mixed Reality.

2.2 Approcci all'adattamento delle interfacce utente in Mixed Reality

In questa sezione vengono analizzati i principali approcci utilizzati per l'adattamento delle interfacce utente in Mixed Reality, evidenziandone il funzionamento, le differenze e le limitazioni. In particolare, verranno discussi sia i metodi basati sull'ottimizzazione, sia gli approcci più recenti basati su modelli *vision–language*.

2.2.1 Approcci basati su optimization

Gli *optimization approaches* affrontano l'adattamento delle interfacce utente in Mixed Reality formulando il problema come un processo di ottimizzazione. In questi metodi, l'obiettivo è identificare automaticamente la configurazione dell'interfaccia che massimizza una determinata misura di qualità — ad esempio la leggibilità, l'utilità percepita o la non invasività — nel rispetto di specifici vincoli, come evitare di coprire informazioni salienti o limitare il carico cognitivo dell'utente. Gli approcci tradizionali possono essere suddivisi principalmente in due categorie: (1) ottimizzazione vincolata e (2) ottimizzazione probabilistica.

Nel primo caso, come nel lavoro di Lindlbauer et al. [1], viene definita una funzione obiettivo accompagnata da un insieme di vincoli che modellano, ad esempio, il livello di dettaglio degli elementi visualizzati e il corrispondente costo cognitivo per l'utente. Sebbene tali metodi siano in grado di effettuare adattamenti in tempo reale e di generare layout complessi in pochi millisecondi, presentano limitazioni significative: richiedono la raccolta di dati sensibili (come l'attività pupillare) e dipendono da funzioni obiettivo rigide e progettate manualmente, che non riescono a cogliere la ricchezza e la variabilità del contesto reale.

Gli approcci probabilistici, come la Bayesian optimization o la stima *maximum a posteriori* (MAP), utilizzano modelli statistici per inferire le preferenze degli utenti a partire da dati raccolti tramite osservazioni dirette o tramite feedback crowd-sourced [18, 19]. Anche questi metodi, però, soffrono di una forte dipendenza dalla raccolta di dati esterni, spesso rumorosi, costosi o difficili da reperire in modo sistematico.

Rispetto a questi approcci convenzionali, UiVLA introduce un paradigma differente: invece di basarsi su funzioni obiettivo predefinite o su dati aggiuntivi dell'utente, sfrutta un modello *vision–language* che prende decisioni adattive direttamente dalle osservazioni visive e dal tracciamento dello sguardo, ossia informazioni già disponibili nei moderni visori MR. Questo elimina la necessità di costruire modelli ad hoc per il carico cognitivo o per il contesto, e consente di esprimere vincoli e preferenze in linguaggio naturale, rendendo l'intero processo più flessibile, scalabile e sensibile al contesto reale.

2.2.2 Approcci basati su modelli vision–language

Negli ultimi anni, i modelli *vision–language* (VLMs) sono stati applicati con successo alla comprensione delle interfacce utente [20] e, più recentemente, all'adattamento delle UI in contesti di Mixed Reality [1]. Questi approcci sfruttano la capacità dei VLM di integrare informazioni visive e linguistiche per interpretare layout, componenti dell'interfaccia e relazioni spaziali in modo più flessibile rispetto ai metodi tradizionali.

Un contributo rappresentativo in questa direzione è il lavoro di Gang Li e Yang Li [20], che introduce un modello VLM progettato per estrarre e interpretare caratteristiche provenienti da interfacce web e mobile a partire da semplici screenshot. Il modello è in grado di riconoscere elementi dell'interfaccia, comprenderne la struttura e supportare compiti di analisi e reasoning a livello semantico, ponendo le basi per applicazioni più avanzate nell'ambito dell'HCI.

Un lavoro più vicino al contesto di questa tesi, invece, è *SituationAdapt* di Zhipeng Li et al. [1], un sistema che ottimizza il layout delle interfacce Mixed Reality posizionando gli elementi della UI in aree contestualmente appropriate. Il sistema utilizza un modello vision–language per valutare la qualità dei posizionamenti in base a fattori contestuali predefiniti e impiega un modulo di ottimizzazione per generare i layout finali a partire da tali valutazioni, come mostrato in Figura 2.2.

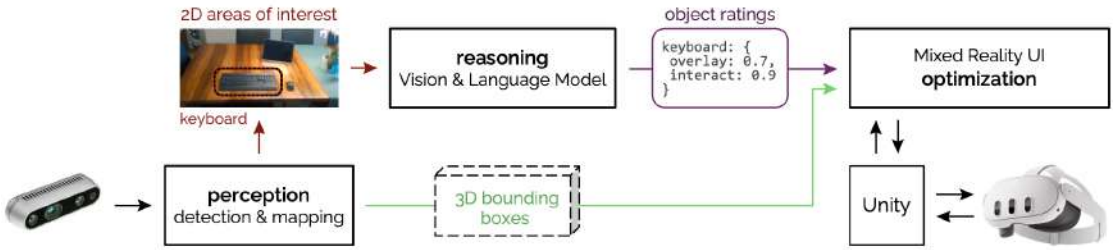


Figura 2.2: Pipeline di *SituationAdapt* per l'adattamento della UI in Mixed Reality: dalla percezione della scena e il reasoning vision–language alla valutazione degli oggetti e all'ottimizzazione dell'interfaccia in Unity.

Gli studi con utenti mostrano che *SituationAdapt* produce posizionamenti più adeguati e più adatti all'interazione rispetto ai metodi precedenti basati sull'ottimizzazione. Tuttavia, nonostante il sistema sia presentato come una soluzione end-to-end, i suoi moduli non possono essere addestrati congiuntamente. Il modulo di percezione, ad esempio, si basa su modelli di *object detection* pre-addestrati, come YOLOv3 [21], e su algoritmi di mappatura rigidi quali SLAM (*Simultaneous Localization and Mapping*) [22].

I modelli di *object detection* pre-addestrati sono reti neurali che hanno già appreso ad individuare e classificare oggetti all'interno di immagini, tipicamente a partire da enormi dataset generici come COCO [23]. YOLOv3, ad esempio, suddivide l'immagine in griglie e predice simultaneamente le bounding boxes e le categorie degli oggetti, raggiungendo buone prestazioni in tempo reale. Sebbene questi modelli siano molto efficaci per scenari generali, presentano una limitata adattabilità: non sono ottimizzati per contesti Mixed Reality specifici e spesso necessitano di ulteriori fasi di fine-tuning per riconoscere oggetti non presenti nei dataset di addestramento.

Analogamente, gli algoritmi di mappatura e localizzazione come SLAM permettono al sistema di ricostruire la struttura tridimensionale dell'ambiente e di stimare la posizione dell'utente al suo interno. SLAM è ampiamente utilizzato in robotica e realtà aumentata; tuttavia, nella sua forma tradizionale, si basa su pipeline rigide composte da moduli indipendenti (feature extraction, data association, pose graph optimization). Questo approccio modulare limita la flessibilità del sistema, in quanto ogni componente opera tramite algoritmi deterministici non apprendibili e difficilmente ottimizzabili congiuntamente al resto della pipeline.

La dipendenza da componenti predefiniti e non integrati riduce la capacità di *SituationAdapt* di adattarsi a nuovi compiti e ambienti, limitandone la generalizzazione complessiva. Inoltre, la natura fortemente modulare della pipeline può introdurre fenomeni di propagazione degli errori: eventuali imprecisioni nei moduli iniziali, come l'object detection o la ricostruzione della scena, possono influenzare negativamente le fasi successive di reasoning e ottimizzazione, compromettendo la qualità finale del posizionamento dell'interfaccia.

Al contrario, UiVLA adotta un'architettura end-to-end unificata basata su un modello *vision–language–action* (VLA), che permette l'ottimizzazione di tutte le sue componenti. Tale progettazione consente al modello di apprendere rappresentazioni più robuste, flessibili e trasferibili sia tra diversi compiti sia attraverso una varietà di contesti visivi. Inoltre, la natura multimodale dei VLM fornisce a UiVLA una maggiore sensibilità al contesto e una più ampia capacità di generalizzazione rispetto ai sistemi che dipendono da moduli predefiniti e non integrati.

2.3 Qwen2.5-VL

Qwen2.5-VL [24] è una famiglia di *Vision–Language Models* (VLM) progettata per integrare in modo nativo comprensione visiva (immagini e video) e generazione testuale autoregressiva. Il modello è stato sviluppato con un focus esplicito su capacità particolarmente rilevanti per applicazioni interattive e contestuali, tra cui: (i) riconoscimento visuale *fine-grained*, (ii) *visual grounding* tramite punti e bounding box, (iii) lettura e parsing di documenti complessi, e (iv) comprensione di video lunghi con capacità di individuare segmenti temporali rilevanti [24].

Queste caratteristiche rendono Qwen2.5-VL particolarmente adatto a scenari in cui il modello non è chiamato solo a descrivere una scena, ma a produrre output strutturati e azionabili a partire da osservazioni visive complesse, come nel caso dell'adattamento delle interfacce utente in Mixed Reality.

A livello architetturale, Qwen2.5-VL segue un paradigma *encoder–decoder* multimodale, illustrato in Figura 2.3. Un **vision encoder** basato su Vision Transformer (ViT) [7] trasforma l'input visuale in una sequenza di feature latenti; tali feature vengono successivamente compresse e proiettate nello spazio di embedding del

language model decoder, basato su Qwen2.5, che genera output testuali o testuali strutturati (ad es. JSON contenenti coordinate o attributi).

2.3.1 Vision Encoder a risoluzione nativa

Il vision encoder di Qwen2.5-VL è un Vision Transformer (ViT) riprogettato per gestire input a *risoluzione nativa* in modo efficiente. Durante training e inferenza, altezza e larghezza dell'immagine vengono ridimensionate a multipli di 28 e suddivise in patch con **stride/patch size pari a 14**. Questa scelta consente di preservare dettagli spaziali fini, cruciali per compiti di grounding e reasoning visivo.

Per ridurre il costo computazionale su sequenze lunghe, il modello introduce un meccanismo di *window-based attention*. In questo schema, l'attenzione non viene calcolata globalmente tra tutti i token della sequenza, ma è limitata a finestre locali che comprendono solo un sottoinsieme di token spazialmente o temporalmente vicini. In questo modo, ciascun token interagisce principalmente con il proprio intorno locale, riducendo significativamente la complessità computazionale dell'attenzione rispetto al caso di *full attention*. Alcuni layer mantengono comunque un'attenzione globale completa, consentendo al modello di preservare una visione d'insieme della scena e di catturare dipendenze a lungo raggio.

2.3.2 Vision–Language Merger

Un limite strutturale dei VLM è che la tokenizzazione visuale produce sequenze molto lunghe (un token per patch), difficili da gestire da un language model autoregressivo. Qwen2.5-VL affronta questo problema tramite un **merger MLP**, che comprime le feature visive prima di passarle all'LLM.

In particolare, il modello raggruppa **blocchi spazialmente adiacenti di 4 patch** (schema 2×2), concatena le feature corrispondenti e applica una **MLP a due layer** che proietta il risultato nella dimensionalità degli embedding testuali. Questo meccanismo riduce significativamente il costo computazionale e la memoria, consentendo una compressione *dinamica* che si adatta a input di dimensione variabile.

Dal punto di vista del paradigma VLA, questa fase è cruciale in quanto consente di trasformare osservazioni visive ad alta dimensionalità in una rappresentazione compatta, direttamente utilizzabile dal modello linguistico per prendere decisioni discrete.

2.3.3 LM Decoder (Qwen2.5) e integrazione multimodale

La componente linguistica è inizializzata dai pesi pre-addestrati di Qwen2.5 e funge da *decoder autoregressivo*. Le feature visuali, una volta compresse dal merger,

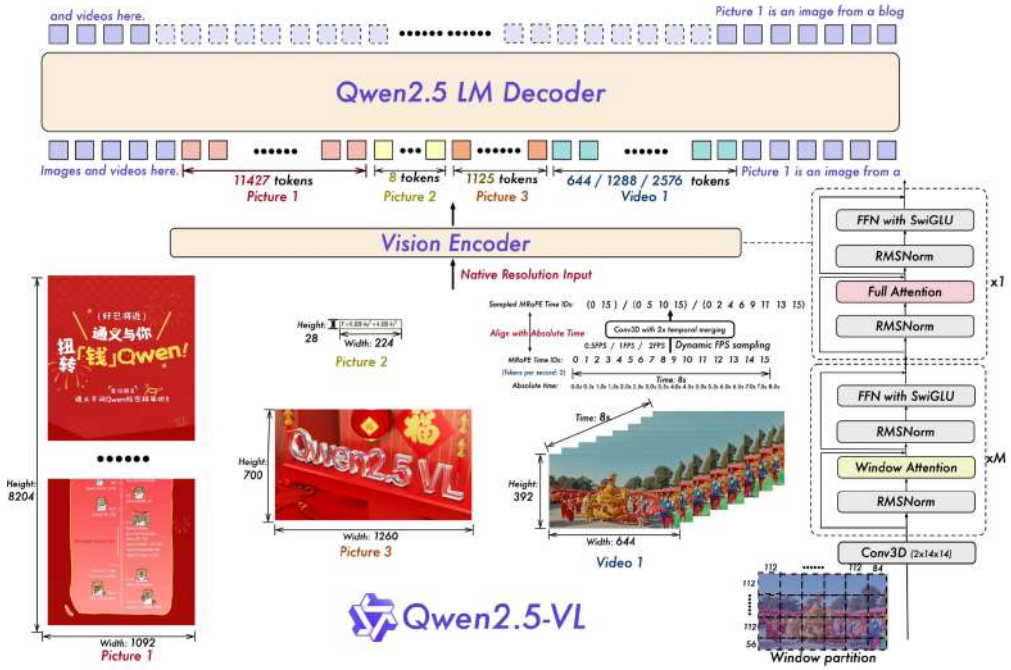


Figura 2.3: Architettura di Qwen2.5-VL. Il Vision Encoder elabora immagini e video a risoluzione nativa producendo sequenze di feature visive; tali feature vengono compresse dal modulo di *vision–language merger* e fornite al LM decoder autoregressivo (Qwen2.5). L'uso di MRoPE e di campionamento FPS dinamico consente una gestione efficace di video lunghi e di dinamiche temporali complesse.

vengono inserite nel contesto dell'LLM come token multimodali, consentendo una generazione condizionata su testo e visione.

Questo design permette al modello di produrre non solo descrizioni naturali, ma anche output strutturati interpretabili come decisioni o azioni simboliche, un aspetto fondamentale per l'integrazione del modello in pipeline di controllo o adattamento dell'interfaccia.

2.3.4 Motivazione della scelta di Qwen2.5-VL per UiVLA

La scelta di Qwen2.5-VL come backbone vision–language di UiVLA è motivata da considerazioni architetturali e funzionali legate al tipo di decisioni che il sistema è chiamato a produrre. In UiVLA, infatti, il modello non deve generare descrizioni libere o predizioni continue, ma fornire decisioni discrete e simboliche: nel *visibility task*, una scelta binaria (mostrare o nascondere un elemento) accompagnata da

una motivazione testuale; nel *placement task*, la selezione di una posizione tra un insieme finito di alternative, codificata come un identificatore discreto.

In questo contesto, Qwen2.5-VL risulta particolarmente adatto in quanto consente di formulare entrambi i task come problemi di *generazione autoregressiva controllata*. L'output del modello è espresso come una sequenza di token appartenenti a uno spazio di risposta ristretto e ben definito (ad esempio {0,1} per la visibilità o un insieme di simboli discreti per il placement), riducendo l'ambiguità dell'output e facilitandone l'interpretazione come azione sulla UI. Questa proprietà è cruciale per l'integrazione del modello in pipeline decisionali, dove le predizioni devono essere affidabili, interpretabili e direttamente azionabili.

Un ulteriore elemento distintivo è l'architettura autoregressiva unificata di Qwen2.5-VL, che permette di apprendere direttamente una mappatura da osservazioni visive e istruzioni linguistiche a decisioni discrete, senza separare esplicitamente le fasi di percezione, valutazione e decisione. Ciò rende naturale l'estensione del modello verso un paradigma *vision–language–action*, in cui l'azione non è un comando continuo ma una scelta simbolica che modifica lo stato dell'interfaccia.

Infine, la disponibilità di varianti di scala e l'attenzione all'efficienza computazionale rendono Qwen2.5-VL compatibile con i vincoli tipici degli scenari di Mixed Reality. In particolare, la variante 3B offre un compromesso efficace tra capacità di comprensione multimodale e sostenibilità computazionale, risultando adatta sia alla generazione del dataset sia alle fasi di training e inferenza in contesti real-time o near real-time. Questo equilibrio è fondamentale per un sistema come UiVLA, che mira a operare in modo reattivo su dispositivi con risorse limitate, mantenendo al contempo una buona capacità di generalizzazione a contesti non visti.

Capitolo 3

UiVLA: Vision–Language–Action Pipeline End-to-End

In questo capitolo viene introdotto **UiVLA**, un sistema Vision–Language–Action progettato per governare l'adattamento dinamico dell'interfaccia utente in scenari di Mixed Reality.

L'obiettivo non è soltanto proporre un modello multimodale capace di prendere decisioni isolate, ma definire un sistema end-to-end in cui percezione, decisione e azione siano integrate all'interno di un unico ciclo operativo.

Questo capitolo fornisce la visione sistemica che motiva l'intero lavoro: la costruzione del dataset, le scelte di preprocessing, il fine-tuning del modello e l'adozione del Multi-Task Learning sono tutti passaggi necessari per rendere possibile la realizzazione concreta di UiVLA.

3.1 Motivazione del sistema UiVLA

Nel contesto della Mixed Reality, l'interfaccia utente non può essere considerata un elemento statico. La sua efficacia dipende dalla relazione dinamica tra contenuto virtuale, ambiente reale e comportamento dell'utente.

Un sistema realmente adattivo deve essere in grado di rispondere a domande quali:

- Dove è opportuno posizionare un elemento UI?
- Il posizionamento scelto è percettivamente accettabile?
- Quando è necessario rivalutare la decisione?

Queste decisioni non sono indipendenti, ma costituiscono un processo iterativo che si sviluppa nel tempo. L'output di una decisione influenza lo stato della scena e condiziona le decisioni successive.

L'obiettivo di questo lavoro è quindi progettare un sistema che non si limiti a classificare o predire in modo isolato, ma che sia in grado di operare all'interno di un ciclo continuo di percezione, decisione e azione. UiVLA nasce con questa finalità: definire una pipeline Vision–Language–Action in grado di governare dinamicamente l'adattamento dell'interfaccia in ambienti reali.

3.2 Architettura concettuale

UiVLA è concepito come un sistema **Vision–Language–Action** end-to-end in cui un modello multimodale svolge il ruolo di motore decisionale all'interno di una pipeline iterativa.

Dal punto di vista concettuale, il sistema integra tre componenti fondamentali:

- una componente di **percezione**, responsabile dell'acquisizione del frame RGB corrispondente al campo visivo dell'utente
- una componente di **decisione**, basata su un modello Vision–Language che analizza il contesto visivo e produce una risposta strutturata
- una componente di **azione**, che traduce l'output del modello in una modifica concreta dello stato dell'interfaccia

La sfida principale consiste nel progettare un modello capace di gestire decisioni eterogenee, quali valutazione della visibilità e selezione della posizione, mantenendo coerenza semantica e percettiva nel tempo.

Per rendere possibile tale comportamento, sarà necessario costruire un dataset adeguato, definire strategie di preprocessing coerenti con il paradigma multimodale e progettare un regime di addestramento che consenta al modello di apprendere competenze multiple in modo integrato. I capitoli successivi affrontano precisamente queste problematiche.

3.3 Pipeline decisionale

La pipeline operativa di UiVLA è strutturata come un ciclo decisionale continuo, attivato in risposta agli eventi percettivi e contestuali dell'utente. Il sistema opera secondo una sequenza di fasi ben definite, illustrate schematicamente in Figura 3.1.

Il funzionamento del sistema può essere descritto come segue:

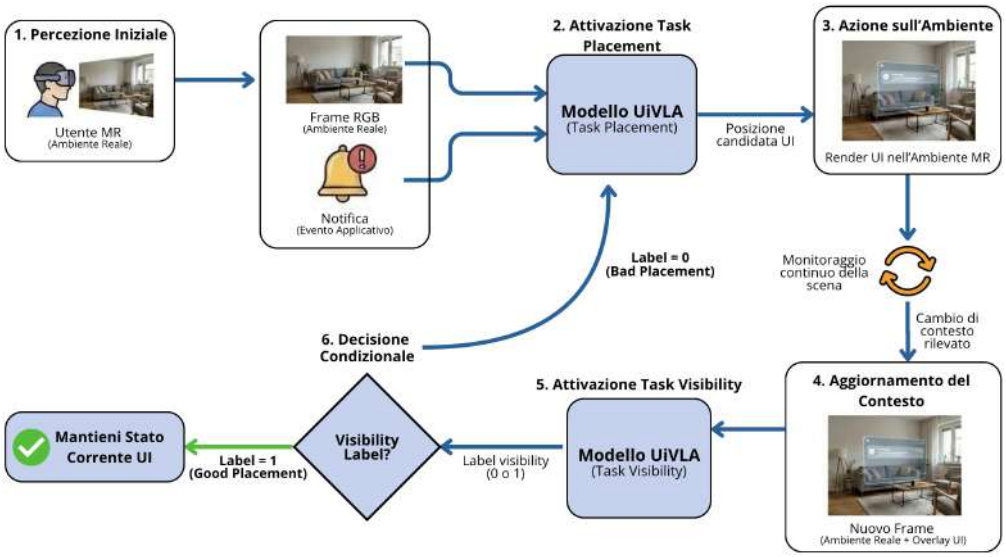


Figura 3.1: Pipeline end-to-end del sistema UiVLA. Il modello Vision–Language–Action governa in modo iterativo i task di placement e visibility all'interno di un ciclo di percezione, decisione e azione.

1. **Percezione iniziale.** L'utente interagisce con l'ambiente attraverso un visore di Mixed Reality.

2. **Attivazione del task di placement.** In seguito a un evento applicativo, come la ricezione di una notifica, il frame RGB corrente corrispondente al campo visivo dell'utente viene acquisito e fornito in input al modello UiVLA nel contesto del task di *placement*. Il modello restituisce la posizione candidata ritenuta più adeguata per il posizionamento dell'elemento di interfaccia.

3. **Azione sull'ambiente.** L'elemento UI viene renderizzato e posizionato nella scena secondo la posizione suggerita dal modello, modificando lo stato visivo dell'ambiente.

4. **Aggiornamento del contesto.** Dopo il posizionamento dell'UI, il sistema entra in una fase di monitoraggio della scena e attende il rilevamento di un cambio effettivo di contesto, inteso come una variazione significativa dell'ambiente o della prospettiva dell'utente. Solo al verificarsi di tale condizione viene acquisito un nuovo frame rappresentativo dello stato aggiornato della scena. A differenza del frame utilizzato nella fase *Attivazione del task di placement*, che contiene esclusivamente l'ambiente reale, questo frame include anche l'ologramma dell'elemento UI precedentemente renderizzato, consentendo al modello di valutare la visibilità dell'interfaccia nel contesto visivo corrente.

5. **Attivazione del task di visibility.** Il nuovo frame viene fornito in input al modello, interrogato nel contesto del task di *visibility*, con l'obiettivo di valutare se la posizione corrente dell'UI risulti percettivamente accettabile.

6. **Decisione condizionale.** Se il modello restituisce una label pari a 1 (*good placement*), il sistema mantiene lo stato corrente dell'interfaccia e rientra nella fase di *Aggiornamento del contesto*, continuando a monitorare la scena in attesa di eventuali variazioni significative. Se invece la label è pari a 0 (*bad placement*), l'elemento UI precedentemente inserito viene prima disattivato e rimosso dalla scena. Successivamente il sistema ritorna alla fase di *Attivazione del task di placement*, avviando una nuova iterazione del ciclo decisionale al fine di individuare una posizione alternativa più adeguata.

Questo meccanismo definisce un ciclo in cui lo stesso modello Vision–Language–Action viene utilizzato per governare dinamicamente il comportamento dell'interfaccia nel tempo, adattandosi ai cambiamenti del contesto visivo e alle conseguenze delle proprie azioni.

3.4 Implicazioni per la progettazione del modello

La definizione di UiVLA come sistema end-to-end impone vincoli specifici alla progettazione del modello sottostante.

In primo luogo, il modello deve essere in grado di affrontare task eterogenei all'interno dello stesso ciclo decisionale. Non è sufficiente ottenere buone prestazioni su un singolo compito isolato: le decisioni devono essere coerenti tra loro e stabili nel tempo.

In secondo luogo, il sistema deve operare in regime iterativo. L'output del modello modifica lo stato della scena e diventa parte dell'input per le decisioni successive. Ciò richiede una rappresentazione interna capace di mantenere consistenza semantica e percettiva.

Infine, l'intero processo deve essere compatibile con vincoli computazionali realistici, sia in termini di latenza che di throughput.

Queste considerazioni motivano le scelte metodologiche adottate nei capitoli successivi: la costruzione di dataset specifici per *visibility* e *placement*, la definizione di una strategia di preprocessing multimodale e l'adozione di un regime di addestramento multi-task volto a unificare competenze differenti all'interno di un unico backbone Vision–Language.

3.5 Modelli Vision–Language–Action (VLA)

Negli ultimi anni è emerso un nuovo paradigma nell'ambito dei modelli multimodali, noto come *Vision–Language–Action* (VLA) [3]. In questo paradigma, i modelli non sono utilizzati esclusivamente per comprendere o descrivere contenuti visivi, ma per produrre decisioni operative che influenzano direttamente il comportamento di un sistema.

Mentre i tradizionali *Vision–Language Models* (VLM) sono tipicamente impiegati per compiti di analisi o generazione linguistica — come *image captioning*, *visual question answering* o ragionamento multimodale — i modelli VLA estendono questa capacità introducendo una terza dimensione: l'**azione**. In questo contesto, l'output del modello non rappresenta soltanto una descrizione o una risposta testuale, ma costituisce un comando o una decisione interpretabile dal sistema che lo utilizza.

Il paradigma VLA è particolarmente rilevante nei sistemi interattivi e nei contesti *embodied*, come robotica, agenti autonomi o applicazioni di Mixed Reality, in cui il modello deve operare all'interno di un ciclo continuo di percezione, decisione e azione. In tali scenari, l'input multimodale rappresenta lo stato corrente dell'ambiente, mentre l'output del modello determina come il sistema debba intervenire per modificarlo.

Nel contesto di questa tesi, l'azione non corrisponde a un controllo motorio continuo, come avviene nella robotica, ma a una decisione simbolica che modifica lo stato dell'interfaccia utente. In particolare, il modello è chiamato a produrre decisioni discrete relative al posizionamento degli elementi dell'interfaccia e alla valutazione della loro visibilità all'interno della scena.

Questa interpretazione dell'output del modello come azione operativa rappresenta il passaggio concettuale fondamentale che consente di trasformare un Vision–Language Model generico in un componente decisionale all'interno della pipeline UiVLA. La sezione successiva descrive quindi come tale trasformazione venga realizzata concretamente nel sistema proposto.

3.6 Dal Vision–Language Model al Vision–Language–Action

Alla luce del paradigma Vision–Language–Action introdotto nella sezione precedente, è possibile interpretare UiVLA come un caso specifico di applicazione di tale modello concettuale al problema dell'adattamento delle interfacce utente in Mixed Reality.

La definizione di UiVLA come sistema Vision–Language–Action end-to-end rende evidente una necessità fondamentale: un modello Vision–Language generico,

pre-addestrato su dati web-scale, non è intrinsecamente specializzato per governare l'adattamento dinamico dell'interfaccia in scenari di Mixed Reality.

I modelli multimodali di larga scala sono tipicamente addestrati per compiti di descrizione, ragionamento visivo o risposta a domande su immagini. Nel contesto di UiVLA, tuttavia, il modello è chiamato a svolgere una funzione diversa: produrre decisioni operative che modificano concretamente lo stato del sistema.

Nel task di *placement*, l'output del modello non consiste in una descrizione testuale, ma nell'identificazione di una regione o di una posizione candidata nella scena. Nel task di *visibility*, il modello restituisce una valutazione binaria che determina se mantenere o modificare lo stato corrente dell'interfaccia. In entrambi i casi, l'output del modello viene interpretato come un'istruzione operativa che altera lo stato dell'ambiente virtuale.

È proprio questa caratteristica a trasformare il modello da Vision–Language Model (VLM) a Vision–Language–Action (VLA): il linguaggio generato non rappresenta il fine dell'inferenza, ma costituisce un'interfaccia simbolica attraverso cui vengono governate azioni concrete del sistema.

Affinché un VLM possa assumere tale ruolo, è necessario specializzarlo rispetto ai task specifici di interesse. Questa specializzazione non può essere ottenuta unicamente tramite prompt engineering, ma richiede la costruzione di un dataset supervisionato che formalizzi in modo esplicito la corrispondenza tra osservazioni visive e decisioni desiderate.

La costruzione del dataset rappresenta quindi un passaggio strutturale del lavoro: essa consente di definire in modo controllato quali competenze il modello debba apprendere, quali vincoli percettivi debbano essere rispettati e quale formato di output debba essere prodotto affinché possa essere interpretato correttamente all'interno della pipeline.

Alla luce di queste considerazioni, risulta necessario definire una strategia di addestramento che consenta di adattare il backbone Vision–Language ai task specifici della pipeline UiVLA. In particolare, è necessario stabilire come strutturare il processo di fine-tuning e in quale ordine affrontare i diversi compiti che il modello dovrà apprendere.

La sezione successiva descrive quindi la strategia di addestramento adottata, illustrando il percorso metodologico seguito per specializzare progressivamente il modello sui task di *visibility* e *placement*.

3.7 Strategia di addestramento del modello

Per adattare il backbone Vision–Language ai requisiti della pipeline UiVLA è stato necessario definire una strategia di addestramento progressiva, in grado di specializzare il modello rispetto ai compiti decisionali richiesti dal sistema.

Prima di affrontare l'integrazione congiunta dei task, è stato infatti opportuno verificare preliminarmente se il modello fosse effettivamente in grado di specializzarsi sui singoli compiti. Per questo motivo, la prima fase sperimentale è stata condotta in regime di *Single-Task Learning*, addestrando separatamente il modello sui task di *visibility* e *placement*.

Questa scelta metodologica risponde a due esigenze principali. In primo luogo, consente di valutare la capacità rappresentazionale del backbone multimodale rispetto a ciascun task in modo isolato, verificando che l'adattamento sia effettivamente possibile. In secondo luogo, permette di evitare un impiego inefficiente delle risorse computazionali: procedere direttamente con un addestramento multi-task avrebbe comportato un costo maggiore in termini di tempo e memoria GPU, senza la garanzia preliminare della fattibilità dell'adattamento.

Una volta verificata la possibilità di specializzare il modello sui task individuali, è stato quindi possibile esplorare un regime di *Multi-Task Learning*, con l'obiettivo di unificare le competenze all'interno di un unico backbone condiviso e rendere il sistema coerente con la natura iterativa della pipeline UiVLA.

Affinché tale processo di addestramento possa essere definito in modo rigoroso, è tuttavia necessario stabilire in modo esplicito quali decisioni il modello debba apprendere. In altre parole, è necessario formalizzare i task che costituiscono le unità fondamentali del processo decisionale all'interno della pipeline UiVLA.

Nel contesto di questo lavoro, l'adattamento dinamico dell'interfaccia viene modellato attraverso due problemi distinti ma complementari: il task di *placement*, che riguarda la selezione della posizione più appropriata in cui collocare un elemento di interfaccia nella scena, e il task di *visibility*, che consiste nella valutazione della qualità percettiva del posizionamento corrente.

La definizione formale di questi task rappresenta quindi un passaggio fondamentale del processo di progettazione, poiché determina la struttura dei dataset, il formato delle annotazioni supervisionate e il tipo di output che il modello dovrà apprendere a generare.

Per questo motivo, il capitolo successivo introduce in modo dettagliato i task di *visibility* e *placement*, descrivendone la formulazione e le implicazioni per la costruzione dei dataset utilizzati nel fine-tuning del modello.

Capitolo 4

Formulazione dei task di visibility e placement

4.1 Motivazione e contesto dei task

Nei sistemi di Mixed Reality (MR), gli elementi di interfaccia vengono sovrapposti direttamente alla scena reale osservata dall'utente. A differenza delle interfacce tradizionali, tali contenuti virtuali condividono lo stesso spazio percettivo del mondo fisico e competono con esso per l'attenzione visiva. Di conseguenza, un overlay può risultare problematico quando *occlude* oggetti rilevanti per l'attività in corso, persone presenti nell'ambiente o elementi critici per la sicurezza e l'orientamento spaziale.

Dal punto di vista della Human–Computer Interaction (HCI), questa problematica è strettamente legata ai concetti di *visual attention*, *interruption management* e *situational awareness*. La *visual attention* riguarda il modo in cui l'attenzione visiva dell'utente viene distribuita tra elementi rilevanti della scena, fisica e digitale; l'*interruption management* si riferisce alla capacità del sistema di presentare informazioni senza interrompere in modo inappropriato l'attività primaria dell'utente; infine, la *situational awareness* descrive il livello di consapevolezza dell'utente rispetto allo stato dell'ambiente e delle attività in corso.

Un'informazione virtuale presentata nel luogo o nel momento sbagliato può quindi distrarre l'utente, aumentare il carico cognitivo e compromettere il flusso dell'attività primaria. In ambienti immersivi e dinamici, tali effetti risultano amplificati, poiché l'utente non distingue più chiaramente tra contenuti fisici e digitali, che vengono percepiti come parte di un'unica scena integrata.

Per questo motivo, la progettazione di interfacce MR richiede meccanismi **context-aware** e **adaptive**, capaci di tenere conto non solo della geometria dell'ambiente, ma anche del compito dell'utente, della sua attenzione visiva e delle

priorità percettive del momento. L'obiettivo non è semplicemente massimizzare la quantità di informazione mostrata, ma bilanciare l'utilità dell'overlay con la minimizzazione dell'interferenza percettiva e cognitiva.

In questo lavoro di tesi, tali esigenze vengono formalizzate attraverso due task complementari: *visibility* e *placement*. Il *visibility task* affronta il problema a livello decisionale, valutando se una specifica sovrapposizione debba essere mostrata o temporaneamente nascosta in base al contesto osservato. Il *placement task*, invece, opera a livello spaziale e determina la posizione più appropriata per l'overlay all'interno della scena, selezionandola da un insieme finito di alternative e minimizzando l'interferenza con elementi rilevanti. Insieme, questi due task consentono di modellare l'adattamento dell'interfaccia come un processo decisionale contestuale, in cui il sistema valuta *se* e *dove* presentare l'informazione virtuale in modo coerente con l'esperienza dell'utente.

4.2 Task di visibility

Il **task di visibility** affronta il problema formulando una decisione binaria: dato un frame della scena reale con un elemento di interfaccia sovrapposto, il sistema deve stabilire se l'overlay **compromette la visibilità di contenuti rilevanti** (*bad placement*) oppure se può essere considerato accettabile (*good placement*). Questa formulazione consente di modellare il problema in modo controllato e direttamente connesso a scenari reali di adattamento dell'interfaccia in MR.

Tuttavia, nelle interfacce adattive una decisione puramente binaria (*show/hide*) risulta spesso insufficiente. Affinché il comportamento del sistema sia comprensibile, affidabile e accettabile per l'utente, è utile accompagnare la predizione con una **motivazione esplicita** che chiarisca *perché* una certa posizione dell'overlay è stata considerata valida o problematica.

Il framework **XAIUI** [25] per interfacce adattive contest-aware evidenzia che, in ambienti dinamici, le spiegazioni devono essere **brevi, mirate e legate al contesto**, evitando motivazioni generiche o ridondanti. In particolare, l'obiettivo non è "spiegare tutto", ma fornire all'utente un *razionale operativo* che supporti tre aspetti chiave:

1. **trasparenza** (capire quali segnali hanno influenzato la decisione)
2. **prevedibilità** (anticipare come il sistema si comporterà in situazioni simili)
3. **fiducia calibrata** (ridurre l'eccessiva fiducia)

Inoltre, XAIUI sottolinea che una spiegazione efficace deve privilegiare fattori **semanticamente interpretabili** (es. "hai coperto le mani" / "stai occludendo un oggetto rilevante" / "stai interferendo con una persona") rispetto a descrizioni

puramente tecniche o geometriche. In linea con tali principi, il modello è quindi progettato per produrre non solo una **label** binaria, ma anche una **reason**: una spiegazione concisa e contestualmente rilevante, ottenuta esplicitando i fattori che hanno determinato la valutazione (ad es. rischio per sicurezza, interferenza funzionale, impatto sociale o degrado estetico). Questo consente di rendere la decisione più interpretabile e di mantenere un collegamento diretto con l'esperienza utente, obiettivo centrale nell'ottica HCI.

4.2.1 Fattori contestuali

A differenza di un'analisi puramente geometrica — ad esempio basata esclusivamente su bounding box o su criteri di non sovrapposizione spaziale — la valutazione della *visibility* di un elemento di interfaccia in Mixed Reality dipende da una molteplicità di fattori eterogenei e fortemente dipendenti dal contesto. Un overlay può infatti risultare geometricamente non invasivo ma comunque problematico dal punto di vista percettivo, funzionale o sociale, a seconda della situazione in cui viene presentato.

In questo lavoro adottiamo come riferimento concettuale i fattori proposti da Li et al. [1], che forniscono una tassonomia utile per caratterizzare le diverse forme di interferenza che un'interfaccia virtuale può introdurre in ambienti MR condivisi. Tali fattori non vengono intesi come regole rigide, ma come dimensioni interpretative che guidano la valutazione dell'adeguatezza di una sovrapposizione rispetto al contesto osservato:

1. **Funzionalità**: l'overlay ostacola la capacità dell'utente di completare l'attività in corso, ad esempio coprendo oggetti rilevanti, strumenti utilizzati attivamente o informazioni necessarie al compito?

2. **Estetica**: la presenza dell'overlay degrada l'aspetto visivo complessivo dell'ambiente o ne riduce la leggibilità, introducendo clutter visivo o distraendo l'attenzione da elementi più rilevanti?

3. **Accettabilità sociale**: l'overlay interferisce con interazioni sociali, ad esempio coprendo il volto o il corpo di altre persone, o risultando inappropriato rispetto al contesto sociale in cui l'utente è immerso?

4. **Salute e sicurezza**: l'overlay introduce potenziali rischi per l'utente, ad esempio nascondendo ostacoli, segnali di pericolo o elementi critici per la navigazione e l'orientamento nello spazio?

Questi fattori riflettono diverse dimensioni dell'esperienza utente e mettono in evidenza come la *visibility* non possa essere ridotta a un singolo criterio oggettivo. Al contrario, essa emerge dall'interazione tra il contenuto dell'interfaccia, il contesto

visivo circostante e le priorità percettive e funzionali dell'utente in un dato momento. Nel contesto di UiVLA, tali dimensioni costituiscono il riferimento concettuale su cui si basa la formulazione dei task di *visibility* e *placement*, consentendo al modello di ragionare sull'interferenza dell'interfaccia in modo più vicino alle esigenze reali dell'interazione in Mixed Reality.

4.2.2 Rappresentazione densa, profondità e interpretabilità

Per modellare i fattori contestuali descritti in precedenza, il task viene affrontato utilizzando una **rappresentazione densa** della scena, definita a livello *pixel-wise*. In questa formulazione, l'ambiente osservato non viene descritto tramite singoli oggetti o regioni isolate, ma attraverso mappe che assegnano un valore a ciascun pixel dell'immagine, catturando in modo continuo e fine-grained le proprietà contestuali della scena.

A ciascuna dimensione contestuale (ad esempio funzionalità, estetica, accettabilità sociale e sicurezza) è associata una **mappa dedicata**, che codifica il grado di interferenza dell'interfaccia rispetto a quella specifica dimensione. A queste mappe si affianca una **mappa di profondità**, che fornisce un'informazione fondamentale sulla distanza relativa tra l'utente e le diverse regioni della scena. La profondità consente di distinguere tra elementi prossimi all'utente — tipicamente più salienti e potenzialmente critici — ed elementi lontani, che risultano in generale meno invasivi dal punto di vista percettivo.

L'insieme delle mappe contestuali e della mappa di profondità costituisce una rappresentazione unificata del contesto visivo e guida sia (i) la decisione finale di *visibility* (accettabile o non accettabile), sia (ii) l'estrazione di una *reason* testuale che motiva la decisione. In questo modo, il modello può tenere conto non solo della presenza di elementi rilevanti, ma anche della loro disposizione spaziale rispetto all'utente, aspetto cruciale negli scenari di Mixed Reality.

Questo approccio consente di mantenere un legame diretto tra i segnali utilizzati dal modello e l'esito della predizione: la spiegazione fornita non è generata in modo post-hoc, ma deriva dalle stesse informazioni che hanno determinato la decisione. Di conseguenza, il processo decisionale risulta più interpretabile, in quanto è possibile ricondurre la scelta del modello a specifiche regioni della scena e alle loro caratteristiche percettive e spaziali che hanno contribuito alla valutazione dell'interferenza dell'interfaccia.

4.2.3 Istanziazione dell'overlay

Nel contesto di questo lavoro, la visibility è istanziata considerando un **singolo elemento di interfaccia** sovrapposto, progettato come una **notifica email** (vedi Figura 4.1). Per ogni frame viene utilizzata una sola notifica, scelta casualmente tra

una variante grafica **chiara** e una **scura**, introducendo una moderata variabilità visiva senza alterare la semantica funzionale dell'overlay. La valutazione si concentra quindi sull'impatto della **posizione** dell'interfaccia sulla visibilità della scena.

(a) Light mode **(b)** Dark mode

Figura 4.1: Esempi di notifica email utilizzata come overlay nei due temi grafici.

4.3 Task di placement

Dopo aver definito il problema della *visibility* come una decisione binaria — ovvero stabilire se un elemento di interfaccia debba essere mostrato o nascosto — estendiamo il framework ad un task più operativo: la **selezione della posizione ottimale** di un elemento di interfaccia all'interno della scena. Questo problema, indicato come **placement task**, consiste nel determinare *dove* collocare l'overlay in modo da **minimizzare l'interferenza** con il contesto visivo, funzionale e sociale dell'utente.

A differenza della visibility, che risponde alla domanda *se* un'informazione possa essere mostrata, il placement affronta la questione complementare del *dove*, richiedendo una valutazione comparativa tra diverse posizioni candidate. In un ambiente di Mixed Reality, tale valutazione deve tenere conto non solo della geometria della scena, ma anche della disposizione degli oggetti, della profondità relativa rispetto all'utente e delle priorità percettive del momento.

4.3.1 Coerenza con visibility e riuso delle mappe

Nel framework **UiVLA**, il placement non viene trattato come un problema indipendente, ma come una naturale estensione del task di visibility. In particolare, entrambi i task si basano sulla stessa rappresentazione contestuale della scena. Le **mappe contestuali** — che codificano aspetti quali funzionalità, estetica, salute e sicurezza, e accettabilità sociale — vengono riutilizzate per valutare il grado di interferenza associato a ciascuna posizione candidata. A queste si affianca un segnale di **profondità**, che consente di distinguere regioni prossime all'utente, generalmente più salienti e potenzialmente più interferenti, da regioni più lontane, che risultano spesso più adatte al posizionamento di elementi di interfaccia.

Questa scelta progettuale garantisce una forte **coerenza semantica** tra i due task: le regioni giudicate critiche nel contesto della visibility sono le stesse che guidano la valutazione della qualità di un posizionamento. In questo modo, il sistema adotta un criterio uniforme di interferenza percettiva, evitando incoerenze tra la decisione di mostrare un overlay e la scelta della sua collocazione spaziale all'interno della scena.

4.3.2 Formulazione discreta tramite Set-of-Mark

Il task di placement viene formulato come un problema **discreto e strutturato** attraverso il paradigma *Set-of-Mark (SoM)* [26], recentemente introdotto come tecnica di *visual prompting* per modelli multimodali vision–language. L'idea chiave alla base del SoM consiste nel trasformare un problema di decisione spaziale intrinsecamente continuo — ad esempio *dove posizionare un elemento di interfaccia* — in una **selezione tra un insieme finito di alternative esplicite**, denominate *mark*.

Nel paradigma SoM, l'immagine originale viene preliminarmente suddivisa in un insieme di **regioni significative** tramite una procedura di *image partition*. Ciascuna regione viene quindi resa *selezionabile* dal modello mediante la sovrapposizione di un **identificatore visivo discreto**, tipicamente alfanumerico, che consente al modello di fare riferimento esplicito a porzioni specifiche della scena durante il ragionamento e la generazione della risposta. In questo modo, una scelta geometrica continua viene ricondotta a una decisione simbolica e verificabile.

Questa formulazione è progettata per soddisfare due requisiti fondamentali. In primo luogo, le regioni candidate devono essere **semanticamente coerenti** e sufficientemente estese, in modo da rappresentare unità visive significative per il task. In secondo luogo, i mark associati a tali regioni devono essere **"speakable"**, ovvero facilmente riconoscibili e referenziabili dal modello attraverso il linguaggio naturale. La combinazione di questi due aspetti consente di stabilire un **grounding esplicito** tra predizione testuale e contenuto visivo.

Nel contesto del placement, l'adozione del paradigma SoM permette di riformulare il problema non come una regressione di coordinate spaziali continue, ma come una **scelta discreta** tra posizioni candidate esplicitamente identificate. Il modello non è quindi chiamato a produrre direttamente una posizione geometrica, bensì a *selezionare* una regione tra quelle disponibili, motivando implicitamente la decisione sulla base del contenuto visivo e del contesto.

Questa impostazione risulta particolarmente adatta ai modelli vision–language autoregressivi, poiché:

1. evita la regressione diretta di coordinate continue, spesso instabile e sensibile alla risoluzione dell'immagine

2. consente di formulare il placement come un problema di **classificazione** o **ranking** tra un insieme finito di alternative

3. abilita un **grounding esplicito e interpretabile**, in cui ogni decisione è riferita a una regione visiva marcata e verificabile

In questo lavoro di tesi, il paradigma SoM viene integrato in modo naturale con la rappresentazione contestuale già introdotta per il task di visibility. A partire dalla **combined map**, che sintetizza i diversi fattori di interferenza percettiva (funzionalità, sicurezza, accettabilità sociale, estetica e profondità), a ciascuna posizione candidata viene associato un **costo di interferenza**. Il task di placement consiste quindi nell'individuare la regione che minimizza tale costo complessivo, selezionando il mark corrispondente.

All'interno del framework proposto sono state esplorate diverse strategie di *image partition* per la generazione delle posizioni candidate, al fine di analizzare l'impatto di differenti modalità di discretizzazione dello spazio visivo. In particolare, sono stati considerati:

1. una **griglia fissa** di celle equidistanti

2. una segmentazione basata su **superpixel SLIC**

3. una segmentazione **prompt-based** mediante Segment Anything

4. una segmentazione semantica per-pixel tramite **Mask2Former**

In tutti i casi, ciascuna regione ottenuta viene interpretata come un *mark* nel senso del paradigma SoM, ovvero come un'alternativa discreta, semanticamente coerente e direttamente selezionabile dal modello. Questa formulazione consente di mantenere un **criterio decisionale uniforme** tra visibility e placement, estendendo il framework da una decisione binaria (*se* mostrare un overlay) a una decisione spaziale strutturata (*dove* posizionarlo), senza introdurre incoerenze concettuali o architetturali.

Questa formulazione consente di mantenere un **criterio decisionale uniforme** tra visibility e placement, estendendo il framework da una decisione binaria (*se* mostrare un overlay) a una decisione spaziale strutturata (*dove* posizionarlo), senza introdurre incoerenze concettuali o architetturali. Un esempio illustrativo del paradigma SoM, tratto direttamente dal lavoro originale, è riportato in Figura 4.2.

4.3.3 Vantaggi e limiti

L'introduzione dell'approccio *Set-of-Mark (SoM)* presenta diversi vantaggi rilevanti. In primo luogo, esso consente di **vincolare e controllare lo spazio delle soluzioni**, evitando le ambiguità tipiche dei metodi continui e garantendo che ogni

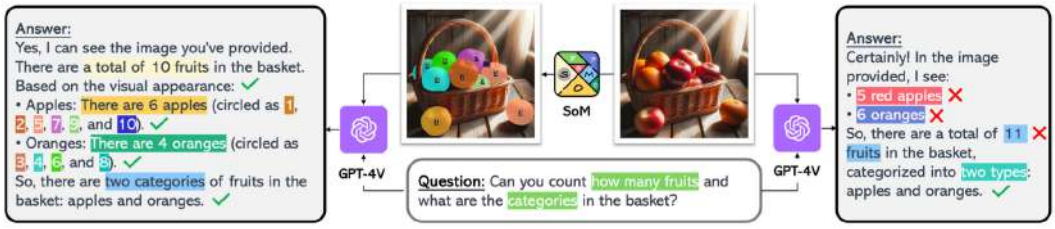


Figura 4.2: Esempio del paradigma *Set-of-Mark (SoM)* tratto da [26]. Nel lavoro originale, il paradigma viene utilizzato in combinazione con GPT-4V come modello *vision–language*. La scena viene arricchita con un insieme finito di mark, ciascuno identificato da un indice discreto, consentendo al modello di riferirsi esplicitamente a elementi o regioni dell'immagine durante il ragionamento e la generazione della risposta. In questo modo, una decisione spaziale continua viene trasformata in una selezione discreta e verificabile.

decisione del modello sia ancorata ad una posizione chiaramente identificabile nella scena. Questo aspetto è particolarmente importante in contesti di Mixed Reality, dove le decisioni di posizionamento devono essere verificabili e interpretabili.

In secondo luogo, la formulazione discreta rende il task di placement **naturalmente compatibile con modelli vision–language autoregressivi**, che possono esprimere la decisione come la selezione di un indice o di un token, senza richiedere architetture dedicate alla regressione di coordinate spaziali. Ciò semplifica l'integrazione del placement all'interno di pipeline multimodali basate su generazione testuale.

Un ulteriore vantaggio riguarda l'**interpretabilità del processo decisionale**. Ciascuna posizione candidata può essere analizzata indipendentemente attraverso il relativo **costo di interferenza**, calcolato a partire dalla *combined map*. Questo consente di confrontare esplicitamente configurazioni alternative e di motivare la scelta finale in termini di impatto sui diversi fattori contestuali (funzionalità, sicurezza, accettabilità sociale, estetica e profondità). Il placement si riduce così a un confronto strutturato tra alternative, selezionando quella che presenta il **minor impatto complessivo sulla scena**.

L'approccio SoM mantiene inoltre una forte **coerenza concettuale con il task di visibility**, riutilizzando la stessa rappresentazione contestuale e abilitando un framework decisionale unificato, in cui decisioni di complessità crescente (mostrare/nascondere l'overlay, quindi determinarne la posizione) vengono risolte in modo consistente.

Tra i limiti, va evidenziato che la discretizzazione introduce inevitabilmente un **errore di quantizzazione**: la posizione ottimale continua potrebbe non coincidere

esattamente con una delle posizioni candidate. Inoltre, la qualità della soluzione dipende dalla strategia adottata per la generazione dei *mark*, rendendo necessario un compromesso tra granularità spaziale, costo computazionale e robustezza del dataset.

Nel complesso, l'approccio Set-of-Mark rappresenta un compromesso efficace tra **semplicità, controllabilità, interpretabilità e compatibilità con modelli vision–language**, risultando particolarmente adatto a scenari di Mixed Reality con vincoli di real-time e risorse computazionali limitate.

4.3.4 Assunzioni sperimentali

Analogamente al task di visibility, in questo lavoro consideriamo il caso di un **singolo elemento di interfaccia** sovrapposto al frame originale. Tale scelta riflette scenari realistici di MR, nei quali le informazioni contestuali vengono mostrate in modo mirato e non intrusivo, e consente di isolare l'effetto della sola variabile *posizione*.

Capitolo 5

Pre-processing e rappresentazione contestuale

Questo capitolo descrive la pipeline di **pre-processing** e di **costruzione della rappresentazione contestuale** utilizzata in questo lavoro. L'obiettivo di tale pipeline è trasformare i dati grezzi acquisiti dal dispositivo in una forma strutturata e interpretabile, adatta a supportare i task di *visibility* e *placement* descritti nel Capitolo 4.

In particolare, il pre-processing include le fasi di acquisizione ed estrazione dei frame, la sincronizzazione con le misure di eye-gaze e la normalizzazione geometrica delle immagini. A partire da tali dati, vengono poi costruite diverse **mappe contestuali pixel-wise** che modellano fattori eterogenei rilevanti per il posizionamento di elementi di interfaccia in scenari di Mixed Reality, quali funzionalità, estetica, safety e accettabilità sociale.

Il capitolo introduce inoltre un'integrazione della **stima di profondità monoculare**, utilizzata per arricchire l'analisi bidimensionale con informazioni strutturali coerenti con la percezione tridimensionale dell'utente. Le diverse mappe vengono infine aggregate in una **combined map**, che rappresenta una sintesi unificata del grado di interferenza associato all'occlusione di ciascuna regione della scena.

5.1 Acquisizione, estrazione dei frame e pre-processing

Il dataset è costruito a partire da registrazioni acquisite con le **Aria glasses**, un dispositivo di ricerca sviluppato nell'ambito del progetto **Project Aria**, una piattaforma progettata per accelerare lo sviluppo di tecnologie di *Augmented Reality* e *Contextual AI*. Le Aria glasses consentono la raccolta di dati multimodali

di alta qualità in scenari reali, fornendo una vista *first-person* dell'ambiente e misure sincronizzate utili allo studio della percezione, dell'attenzione visiva e dell'interazione uomo–ambiente. In questo lavoro, le registrazioni sono memorizzate in formato `.vrs` e accompagnate da file `.csv` contenenti le misure di *eye-gaze*.

Le sequenze utilizzate sono state selezionate e scaricate tramite l'**Aria Dataset Explorer**, che fornisce accesso strutturato alle registrazioni e ai relativi metadati. I video considerati coprono una varietà di **contesti sia indoor che outdoor**, includendo ambienti domestici, aree pubbliche e scene all'aperto. Questa eterogeneità consente di catturare condizioni visive e contestuali molto diverse tra loro, come variazioni di illuminazione, varietà di oggetti, profondità della scena e dinamiche di movimento, risultando particolarmente rilevante per lo studio dell'adattamento delle interfacce in Mixed Reality. Lo strumento permette inoltre di filtrare e ispezionare le sequenze in modo sistematico, garantendo la tracciabilità delle condizioni di acquisizione e delle informazioni sensoriali associate a ciascun frame.

Per ridurre la ridondanza temporale e contenere il costo computazionale delle successive fasi di elaborazione, i frame RGB vengono campionati **ogni 10 frame** dal flusso video originale, acquisito a 20 fps. Ne risulta un rate effettivo di **2 fps**, tale per cui ciascuna immagine rappresenta uno snapshot della scena ogni **0.5 secondi**. Questo compromesso consente di preservare una buona copertura temporale del contesto visivo, evitando al contempo un'eccessiva correlazione tra frame consecutivi.

Le misure di *eye-gaze*, acquisite ad una frequenza superiore rispetto al flusso video, vengono sincronizzate temporalmente con ciascun frame selezionando il campione di gaze più vicino in termini di timestamp. La direzione dello sguardo, espressa in termini di angoli di yaw e pitch, viene quindi proiettata sul piano immagine mediante un **modello pinhole**, cioè modello di proiezione geometrica che approssima la fotocamera come un punto ideale, mappando una direzione nello spazio tridimensionale su una coordinata bidimensionale nel piano immagine tramite una trasformazione prospettica, ottenendo coordinate pixel (x_g, y_g) che rappresentano il punto di fissazione dell'utente all'interno dell'immagine.

Poiché le immagini acquisite dal dispositivo risultano ruotate rispetto all'orientamento naturale di visualizzazione, ogni frame viene ruotato di **90° in senso orario** e successivamente ridimensionato a una risoluzione uniforme di **1408×1408** pixel. Le coordinate di eye-gaze vengono trasformate in modo coerente applicando le stesse operazioni geometriche, garantendo la consistenza spaziale tra contenuto visivo e punto di fissazione.

Il risultato di questo processo è un insieme di immagini PNG normalizzate, ciascuna associata alle corrispondenti coordinate di eye-gaze. Questo insieme costituisce la base visiva del dataset e viene utilizzato come input per la generazione delle mappe contestuali e per l'annotazione delle label relative ai task di visibility e placement.

5.2 Mappe contestuali

Come discusso nella sezione sui fattori contestuali (vedi Sezione 4.2.1), la valutazione dell'interferenza di un overlay in Mixed Reality non può essere ridotta a un singolo indicatore visivo o a un criterio puramente geometrico. Al contrario, essa richiede di considerare simultaneamente diverse dimensioni dell'esperienza dell'utente, che riflettono aspetti percettivi, funzionali e sociali della scena osservata.

Per rendere tali fattori operativi all'interno del framework proposto, ogni frame viene arricchito con un insieme di **mappe contestuali**, definite a livello *pixel-wise*. Ciascuna mappa fornisce una rappresentazione spaziale continua di una specifica dimensione contestuale, consentendo di stimare in modo fine-grained l'impatto potenziale di un overlay sulle diverse regioni dell'immagine.

In particolare, utilizziamo tre mappe principali, che costituiscono una traduzione diretta dei fattori concettuali introdotti in precedenza:

- **Aesthetic map** M_{aes}: rappresenta la salienza percettiva della scena, evidenziando le regioni visivamente più rilevanti o informative. Overlay collocati in tali aree tendono a introdurre clutter visivo o a distrarre l'attenzione dell'utente.
- **Functionality map** M_{fun}: identifica le regioni funzionalmente rilevanti rispetto al compito dell'utente, come oggetti manipolati attivamente, strumenti in uso o elementi necessari al completamento dell'attività corrente. L'occlusione di tali regioni compromette direttamente la funzionalità dell'interfaccia.
- **Safety & social acceptability map** M_{safe}: evidenzia le regioni che non dovrebbero essere occluse per motivi di sicurezza o di accettabilità sociale, ad esempio persone presenti nella scena, volti, parti del corpo, ostacoli fisici o segnali critici per l'orientamento e la navigazione.

Queste mappe forniscono una rappresentazione spaziale esplicita dei principali fattori di interferenza e costituiscono la base per le decisioni di *visibility* e *placement*. Integrandole con l'informazione di profondità, descritta nelle sezioni successive, il sistema può distinguere non solo *quali* regioni siano rilevanti, ma anche *quanto* esse siano prossime all'utente, permettendo una valutazione più coerente con la percezione umana in ambienti di Mixed Reality.

5.2.1 Aesthetic map

La **aesthetic map** ha l'obiettivo di modellare la *salienza percettiva* della scena, individuando le regioni che risultano visivamente rilevanti per l'utente e che, di conseguenza, sono meno adatte a essere occluse da un elemento di interfaccia. In

generale, aree caratterizzate da colori saturi o da strutture ad alto contrasto tendono ad attirare maggiormente l'attenzione visiva e a contribuire in modo significativo alla leggibilità e all'estetica complessiva dell'ambiente.

Seguendo l'intuizione proposta da Dudley et al. [19], la mappa viene costruita combinando due segnali visivi complementari:

1. la **saturazione cromatica**, misurata tramite il canale S nello spazio colore HSV, che indica quanto un colore sia intenso o "vivo" rispetto al grigio. Valori elevati di saturazione corrispondono a regioni ricche di colore, che tendono ad attirare maggiormente l'attenzione visiva dell'utente;

2. una mappa di **contrasto dei bordi**, ottenuta applicando l'operatore di Scharr [27], che evidenzia le variazioni brusche di intensità luminosa nell'immagine. Tali variazioni corrispondono tipicamente a contorni, spigoli e strutture geometriche, ovvero elementi visivamente salienti che contribuiscono in modo significativo alla percezione della scena.

Le due mappe vengono quindi aggregate mediante un'operazione di massimo a livello pixel-wise:

$$M_{\text{aes}}(p) = \max\Big(M_{\text{sat}}(p),\, M_{\text{edge}}(p)\Big), \tag{5.1}$$

dove p indica un generico pixel dell'immagine.

Questa formulazione consente di catturare in modo compatto sia regioni visivamente salienti dal punto di vista cromatico sia strutture geometriche ad alto contrasto, fornendo una stima interpretabile e direttamente collegata a caratteristiche percettive della scena. La *aesthetic map* risultante penalizza quindi il posizionamento dell'interfaccia in aree che contribuiscono in modo significativo all'aspetto visivo e alla leggibilità dell'ambiente.

5.2.2 Functionality map

La **functionality map** ha l'obiettivo di modellare la regione della scena che è direttamente correlata al **focus funzionale** dell'utente, ovvero l'area che risulta più rilevante per il compito che l'utente sta svolgendo in un dato momento. In questo lavoro utilizziamo il punto di fissazione dello sguardo come **segnale funzionale**, assumendo che esso rappresenti l'area della scena che l'utente sta attualmente osservando e con cui sta interagendo o che sta monitorando attivamente nell'ambito del compito in corso.

Dato un frame, viene innanzitutto eseguita una fase di *object detection* tramite **YOLO-World** [28], ottenendo un insieme di bounding box $\{b_i\}$ associate agli oggetti presenti nella scena. YOLO-World è un detector *open-vocabulary*, ovvero un modello in grado di riconoscere un ampio insieme di categorie senza richiedere un addestramento specifico per ciascuna classe. Questa caratteristica è particolarmente

rilevante nel nostro scenario *first-person*, in cui gli oggetti osservati dall'utente possono variare significativamente tra scene e contesti, rendendo preferibile un approccio più generalizzabile rispetto a detector chiusi su un insieme limitato di classi.

Per individuare l'oggetto funzionalmente rilevante, si seleziona il bounding box il cui centro $(c_{x,i}, c_{y,i})$ risulta più vicino al punto di fissazione (x_g, y_g):

$$i^* = \arg\min_i \|(x_g, y_g) - (c_{x,i}, c_{y,i})\|_2. \tag{5.2}$$

La selezione viene accettata solo se la distanza tra il centro dell'oggetto e il punto di fissazione è inferiore a una soglia T. In questo lavoro, la soglia è fissata a $T = W/4$, dove W indica la larghezza dell'immagine. Tale valore è stato scelto empiricamente come compromesso tra la tolleranza agli errori di stima dell'eye-gaze e dell'object detection e la necessità di associare l'oggetto selezionato al reale focus attentivo dell'utente. In pratica, questa scelta consente di individuare correttamente l'oggetto osservato anche in presenza di lievi imprecisioni, evitando al contempo associazioni spurie con oggetti significativamente distanti dal punto di fissazione. Qualora nessun oggetto soddisfi questo vincolo, si assume che non vi sia un focus funzionale chiaramente identificabile e la functionality map viene lasciata nulla.

La mappa finale è definita in forma binaria:

$$M_{\text{fun}}(p) = \begin{cases} 255 & \text{se } p \in b_{i^*}, \\ 0 & \text{altrimenti}, \end{cases} \tag{5.3}$$

dove p indica un pixel dell'immagine. In questo modo, la functionality map evidenzia esplicitamente la regione della scena che non dovrebbe essere occlusa dall'interfaccia in quanto direttamente rilevante per l'attività dell'utente.

In aggiunta, viene registrata la classe predetta da YOLO-World per l'oggetto selezionato. Questa informazione non influisce sulla costruzione della mappa, ma risulta utile per analisi qualitative e per la generazione della *reason* nel task di visibility, consentendo di motivare la decisione del modello facendo riferimento all'oggetto effettivamente osservato e/o utilizzato dall'utente.

5.2.3 Safety and social acceptability map

La **safety & social acceptability map** ha l'obiettivo di evidenziare le regioni della scena che non dovrebbero essere occluse da un overlay per motivi di **sicurezza fisica** o di **accettabilità sociale**. In contesti di Mixed Reality, infatti, la sovrapposizione di contenuti virtuali su elementi critici dell'ambiente reale può introdurre rischi concreti per l'utente (ad esempio ostacolando la percezione del percorso o di segnali di pericolo) oppure risultare socialmente inappropriata, come nel caso dell'occlusione di persone presenti nella scena.

Per costruire questa mappa utilizziamo una segmentazione semantica basata su **Mask2Former** [29], un modello *transformer-based* per la segmentazione universale che unifica in un'unica architettura i compiti di segmentazione semantica, di istanza e panottica. Mask2Former adotta una formulazione di *mask classification*: a partire da feature multi-scala estratte dall'immagine, il modello predice un insieme di maschere associate a classi semantiche, evitando pipeline complesse e risultando robusto su scene reali caratterizzate da forte variabilità e da molteplici oggetti.

L'impiego di Mask2Former è particolarmente vantaggioso nel nostro scenario per due motivi principali. In primo luogo, grazie all'uso di feature multi-scala e al ragionamento globale tipico dei transformer, il modello mantiene una buona accuratezza anche su strutture estese o sottili — come pavimenti, scale o bordi di porte — che sono particolarmente rilevanti dal punto di vista della sicurezza. In secondo luogo, la segmentazione pixel-wise consente di ottenere una rappresentazione **interpretabile** delle regioni critiche, direttamente utilizzabile per penalizzare l'occlusione di aree sensibili senza ricorrere a euristiche basate su bounding box o a detector specializzati per singole categorie.

Nel nostro setup, definiamo un insieme di classi semantiche considerate **critiche**:

$$\mathcal{C}_{\text{safe}} = \begin{Bmatrix} \texttt{floor}, \texttt{road}, \texttt{window}, \texttt{door}, \texttt{stairs}, \\ \texttt{traffic light}, \texttt{rug}, \texttt{person}, \texttt{car}, \texttt{bus}, \texttt{truck} \end{Bmatrix}$$

La selezione di queste classi è motivata da considerazioni legate alla sicurezza e all'interazione sociale in scenari reali. Superfici come *floor*, *road*, *stairs* e *rug* sono direttamente connesse alla locomozione e all'equilibrio dell'utente; la loro occlusione può compromettere la navigazione nello spazio e aumentare il rischio di incidenti. Elementi strutturali quali *door* e *window* sono importanti per l'orientamento e per la comprensione dell'ambiente. Oggetti dinamici o potenzialmente pericolosi, come veicoli (*car*, *bus*, *truck*) e segnali (*traffic light*), rivestono un ruolo critico soprattutto in contesti urbani. Infine, la classe *person* è inclusa per motivi di accettabilità sociale: l'occlusione di altre persone o dei loro corpi risulta spesso inappropriata e può interferire con le interazioni sociali.

A partire dalla label map prodotta dalla segmentazione, $S \in \{0, \ldots, K-1\}^{H \times W}$, costruiamo una maschera binaria:

$$M_{\text{safe}}(p) = \begin{cases} 255 & \text{se } S(p) \in \mathcal{C}_{\text{safe}}, \\ 0 & \text{altrimenti}, \end{cases} \tag{5.4}$$

dove p indica un pixel dell'immagine. Il risultato è una mappa binaria (0/255) che identifica in modo esplicito e interpretabile le regioni della scena da evitare durante il placement dell'overlay, contribuendo a garantire sia la sicurezza dell'utente sia il rispetto delle norme implicite di accettabilità sociale.

5.3 Integrazione della stima di profondità per scenari di Mixed Reality

Sebbene il sistema operi a partire da input visivi bidimensionali (frame 2D), negli scenari di Mixed Reality (MR) un aspetto cruciale è rappresentato dalla relazione tridimensionale tra l'utente e gli elementi presenti nella scena. In particolare, la **distanza degli oggetti dal punto di vista dell'utente** influisce in modo significativo sull'impatto percettivo e funzionale di un'eventuale occlusione introdotta da un'interfaccia sovrapposta: coprire un oggetto vicino tende a risultare molto più invasivo rispetto all'occlusione di un elemento distante o sullo sfondo.

Per incorporare questa informazione senza ricorrere a sensori 3D dedicati, introduciamo una componente di **stima della profondità monoculare**, basata su *Dense Prediction Transformer* (DPT) [30]. DPT è un modello che, a partire da una singola immagine RGB, produce una mappa di profondità che fornisce una stima *relativa* della distanza di ciascun pixel dalla camera. Pur non essendo metricamente accurata, questa stima è sufficiente a catturare la struttura spaziale della scena e a distinguere regioni prossime all'utente da regioni più lontane.

L'informazione di profondità consente in particolare di:

- distinguere in modo esplicito tra elementi **vicini** e **lontani** rispetto al punto di vista dell'utente
- attenuare penalizzazioni non necessarie quando l'overlay si sovrappone a contenuti distanti, come persone sullo sfondo o strutture architettoniche lontane
- definire criteri decisionali più aderenti agli scenari di Mixed Reality, nei quali la rilevanza di un'occlusione dipende non solo dalla posizione bidimensionale, ma anche dalla distanza dell'oggetto rispetto all'utente

Nel framework proposto, l'integrazione della profondità è realizzata modulando il contributo delle diverse mappe contestuali in funzione della profondità stimata. In questo modo, regioni caratterizzate da elevata salienza visiva ma localizzate a grande distanza dall'utente esercitano un'influenza ridotta sul processo decisionale, mentre elementi prossimi o potenzialmente interattivi mantengono un peso maggiore.

Questa scelta consente di superare un approccio puramente bidimensionale e di introdurre nel sistema segnali percettivi più coerenti con l'esperienza dell'utente in ambienti di Mixed Reality, migliorando la qualità delle decisioni di visibility e placement senza richiedere informazioni 3D esplicite o sensori aggiuntivi.

5.3.1 Implementazione della mappa di profondità

Per integrare in modo efficace l'informazione di profondità nel processo decisionale, adottiamo una strategia semplice e interpretabile che consiste nel **modulare il contributo delle mappe contestuali in funzione della distanza stimata dall'utente**. In particolare, la profondità viene utilizzata per ridimensionare l'influenza delle mappe *safety* e *aesthetic*, entrambe normalizzate nell'intervallo [0,255], in modo coerente con la percezione umana negli scenari di Mixed Reality.

L'intuizione alla base di questa scelta è che un pixel può risultare visivamente o semanticamente rilevante, ma avere un impatto limitato sull'esperienza dell'utente se localizzato a grande distanza. Di conseguenza, un pixel che presenti un valore elevato in una delle mappe contestuali ma risulti allo stesso tempo **lontano** dall'utente — ovvero associato a valori alti nella depth map — viene escluso dal processo decisionale, annullandone il contributo. Questo meccanismo evita che elementi distanti, come superfici architettoniche o persone sullo sfondo, influenzino in modo sproporzionato la scelta della posizione dell'overlay.

Un'eccezione importante riguarda la **functionality map**, sulla quale l'informazione di profondità non viene applicata. In questo caso, infatti, la mappa rappresenta il **focus funzionale corrente** dell'utente, derivato dal punto di fissazione dello sguardo. Anche se l'oggetto osservato si trova fisicamente a una certa distanza, la sua eventuale occlusione rimane critica per l'attività in corso. Ad esempio, se l'utente sta osservando uno schermo o un pannello informativo posto a distanza, l'interfaccia non deve interferire con tale contenuto indipendentemente dalla profondità stimata.

La stima della profondità è ottenuta mediante **DepthPro** [31], un modello di *monocular depth estimation* di recente introduzione nella libreria *transformers* di Hugging Face. DepthPro è in grado di produrre **depth map dense e ad alta risoluzione** a partire da una singola immagine RGB, preservando la struttura geometrica della scena anche in condizioni complesse, come ambienti interni con illuminazione variabile o la presenza di oggetti sottili e parzialmente occlusi (vedi Figura 5.1).

La mappa di profondità predetta viene ridimensionata alla risoluzione originale dell'immagine e normalizzata nell'intervallo [0,1] mediante una normalizzazione min–max effettuata separatamente per ciascun frame. In tale rappresentazione, valori prossimi a 0 indicano regioni **più vicine** all'utente, mentre valori prossimi a 1 corrispondono a regioni **più lontane**. Alcuni esempi qualitativi sono riportati in Figura 5.2, che mostra la relazione tra la struttura visiva della scena e i valori di profondità stimati.

Figura 5.1: Esempi qualitativi di stima della profondità ottenuti con **DepthPro** presi direttamente dal paper ufficiale [31]. La prima riga mostra le immagini RGB di input, mentre la seconda riga riporta le corrispondenti depth map predette. I riquadri evidenziano la capacità del modello di preservare dettagli fini e strutture sottili (ad es. peli, fili e griglie) e di mantenere una separazione coerente tra primo piano e sfondo anche in scene complesse.

Filtraggio basato su soglia

Una volta ottenuta la depth map normalizzata, applichiamo un **filtraggio basato su soglia** alle mappe *safety* e *aesthetic* prima del calcolo della *combined map*. Fissata una soglia di profondità τ, tutti i pixel che soddisfano la condizione $D(p) > \tau$ vengono considerati sufficientemente lontani e quindi esclusi, forzando il loro contributo a zero. Al contrario, i pixel per cui $D(p) \leq \tau$ vengono mantenuti invariati, in quanto ritenuti spazialmente rilevanti per l'utente.

Dal punto di vista implementativo, questo filtraggio è realizzato imponendo una semplice condizione di "vicinanza" tramite una maschera booleana, applicata alle mappe contestuali. Il dettaglio operativo è riportato nel Listing 5.1.

```
# --- Depth-filtered safety map ---
safety_mask = (safety_and_social_acceptability_map > 0)
close_enough_safety = (depth_map <= self.depth_threshold)

# Keep only pixels that are both safety-related AND not too far
keep_safety = safety_mask & close_enough_safety

# Convert to binary map (0 or 255)
safety_and_social_acceptability_map = keep_safety.astype(np.uint8) *
    255

# --- Depth-filtered aesthetics map ---
close_enough_aesthetic = (depth_map <= self.depth_threshold)
aesthetics_map[~close_enough_aesthetic] = 0
```

(a) Frame originale.

(b) Depth map normalizzata.

(c) Frame originale.

(d) Depth map normalizzata.

Figura 5.2: Esempi qualitativi di stima della profondità: per ciascuna coppia, a sinistra il frame RGB originale e a destra la corrispondente depth map normalizzata.

Listing 5.1: Filtraggio depth-based delle mappe *safety* e *aesthetic*.

Questa strategia garantisce che **solo i contenuti spazialmente rilevanti influenzino le decisioni di visibility e placement dell'overlay**, rendendo il

comportamento del sistema più coerente con i requisiti percettivi e funzionali tipici degli scenari di Mixed Reality, pur mantenendo un'implementazione semplice e facilmente interpretabile.

5.4 Combined map

Le mappe contestuali descritte nelle sezioni precedenti vengono aggregate in un'unica rappresentazione, denominata **combined map** S. Questa mappa sintetizza, a livello *pixel-wise*, il grado di interferenza atteso qualora una specifica regione della scena venisse occlusa da un elemento di interfaccia. L'obiettivo della combined map non è stimare una salienza visiva generica, bensì costruire una **mappa di regioni da evitare**, in cui valori elevati indicano pixel la cui occlusione risulterebbe problematica per almeno una delle dimensioni considerate: funzionalità, sicurezza/accettabilità sociale o estetica.

5.4.1 Normalizzazione e semantica dei valori

Le tre mappe utilizzate nella costruzione di S — **aesthetic map** M_{aes}, **safety & social acceptability map** M_{safe} e **functionality map** M_{fun} — sono rappresentate sulla stessa scala di intensità, con valori normalizzati nell'intervallo [0,255]. In questa codifica, il valore 0 indica l'assenza di interferenza, mentre il valore 255 corrisponde a una regione altamente critica che non dovrebbe essere occlusa.

In particolare, le mappe M_{safe} e M_{fun} sono binarie (0/255), riflettendo la presenza o assenza di vincoli forti di sicurezza o funzionalità. La mappa M_{aes}, invece, può assumere valori continui in funzione della saturazione cromatica e del contrasto dei bordi, ma viene comunque ricondotta alla stessa scala per garantire confrontabilità tra le diverse dimensioni contestuali.

5.4.2 Aggregazione tramite massimo pixel-wise

La combined map viene calcolata applicando un'operazione di massimo a livello di pixel:

$$S(p) = \max\Big(M_{\text{aes}}(p),\, M_{\text{safe}}(p),\, M_{\text{fun}}(p)\Big), \tag{5.5}$$

dove p denota un generico pixel dell'immagine.

L'uso del massimo è una scelta deliberata e coerente con l'obiettivo di minimizzare l'interferenza dell'interfaccia. In particolare, se una regione è critica per almeno una delle dimensioni considerate — ad esempio perché funzionalmente rilevante, safety-critical o percettivamente saliente — la sua occlusione dovrebbe essere penalizzata indipendentemente dagli altri fattori. Rispetto a combinazioni additive o somme pesate, l'operatore di massimo presenta diversi vantaggi:

1. mantiene la combined map facilmente interpretabile, poiché il valore finale riflette la dimensione contestuale dominante

2. riduce la necessità di introdurre pesi e iperparametri per bilanciare mappe eterogenee

3. evita che segnali forti e localizzati vengano attenuati o "diluiti" da operazioni di media

5.4.3 Implementazione

La costruzione della combined map è implementata direttamente a livello pixel-wise, in modo aderente alla formulazione teorica. Come discusso nella sezione precedente, l'informazione di profondità viene utilizzata esclusivamente come filtro per le componenti *aesthetic* e *safety/social*, mentre la *functionality map* viene sempre preservata, in quanto rappresenta il focus funzionale corrente dell'utente.

Il Listing 5.2 riflette direttamente la definizione dell'Eq. 5.5, mostrando come la combined map venga ottenuta applicando una regola di massimo pixel-wise dopo un filtraggio adattivo basato sulla profondità. Questa implementazione garantisce una stretta coerenza tra la definizione concettuale della combined map e il comportamento operativo del sistema, rendendo il processo decisionale trasparente e interpretabile.

```
# Filter safety map and aesthetic map based on depth
[...]

# Combine maps using pixel-wise maximum
combined_map = np.max(
    np.stack([
        aesthetic_map,
        safety_map,
        functionality_map
    ]),
    axis=0
)
```

Listing 5.2: Costruzione della combined map con filtraggio basato sulla profondità.

5.4.4 Integrazione della profondità come filtro

L'informazione di profondità viene integrata nel framework **non come una quarta mappa da aggregare**, ma come un **meccanismo di filtraggio (gating)** che regola l'influenza delle componenti *aesthetic* e *safety/social* in funzione della distanza stimata degli elementi dalla posizione dell'utente.

Dato una depth map normalizzata $D \in [0,1]^{H \times W}$, viene definita una soglia adattiva:

$$\tau = \text{percentile}(D, 60), \tag{5.6}$$

che separa, in modo *relativo e frame-specific*, le regioni considerate *più vicine* all'utente ($D(p) \leq \tau$) da quelle *più lontane* ($D(p) > \tau$). I pixel appartenenti a regioni lontane vengono esclusi dal contributo delle mappe *aesthetic* e *safety/social*, annullandone l'influenza nella fase di aggregazione.

L'uso di una soglia basata su percentili, anziché su un valore assoluto, consente di adattare dinamicamente il filtraggio a scene con profondità e geometrie differenti, evitando assunzioni rigide sulla scala metrica della profondità stimata.

Dal punto di vista percettivo, questo meccanismo riflette il fatto che, negli scenari di Mixed Reality, l'impatto di un'occlusione dipende non solo dal *tipo* di contenuto occluso, ma anche dalla sua *distanza* dall'utente. In pratica, elementi come pavimenti, finestre o persone sullo sfondo — pur potenzialmente salienti dal punto di vista estetico o semantico — esercitano un'influenza ridotta quando sono molto distanti, rendendo la decisione di placement più coerente con l'esperienza percettiva reale.

È importante notare che la **functionality map** non viene filtrata in base alla profondità: anche se l'oggetto osservato è fisicamente distante, esso rappresenta comunque il *focus funzionale corrente* dell'utente e la sua occlusione rimane critica indipendentemente dalla distanza.

5.4.5 Interpretazione operativa della combined map

La combined map S può essere interpretata come una **mappa densa di evitabilità** (*avoidance map*), che assegna a ciascun pixel un valore proporzionale al grado con cui quella regione dovrebbe essere evitata per il posizionamento di un overlay. Valori elevati indicano aree la cui occlusione risulterebbe altamente problematica per almeno una dimensione contestuale, mentre valori bassi corrispondono a regioni più idonee al posizionamento dell'interfaccia.

La Figura 5.3 illustra esempi completi della pipeline di costruzione della combined map su diversi frame. Per ciascun caso sono mostrati, da sinistra a destra, il frame RGB originale (*camera view*), le mappe contestuali individuali (functionality, aesthetics, social/safety e depth) e la combined map finale. È possibile osservare come regioni evidenziate in una qualunque delle mappe di partenza vengano propagate nella combined map, in accordo con il principio di aggregazione conservativa adottato: la presenza di un singolo fattore critico è sufficiente a rendere una regione non idonea all'occlusione.

Questa rappresentazione costituisce la base comune per entrambi i task considerati nel framework UiVLA. Nel task di *visibility*, la combined map viene integrata sull'area occupata dall'overlay per derivare una decisione binaria di accettabilità.

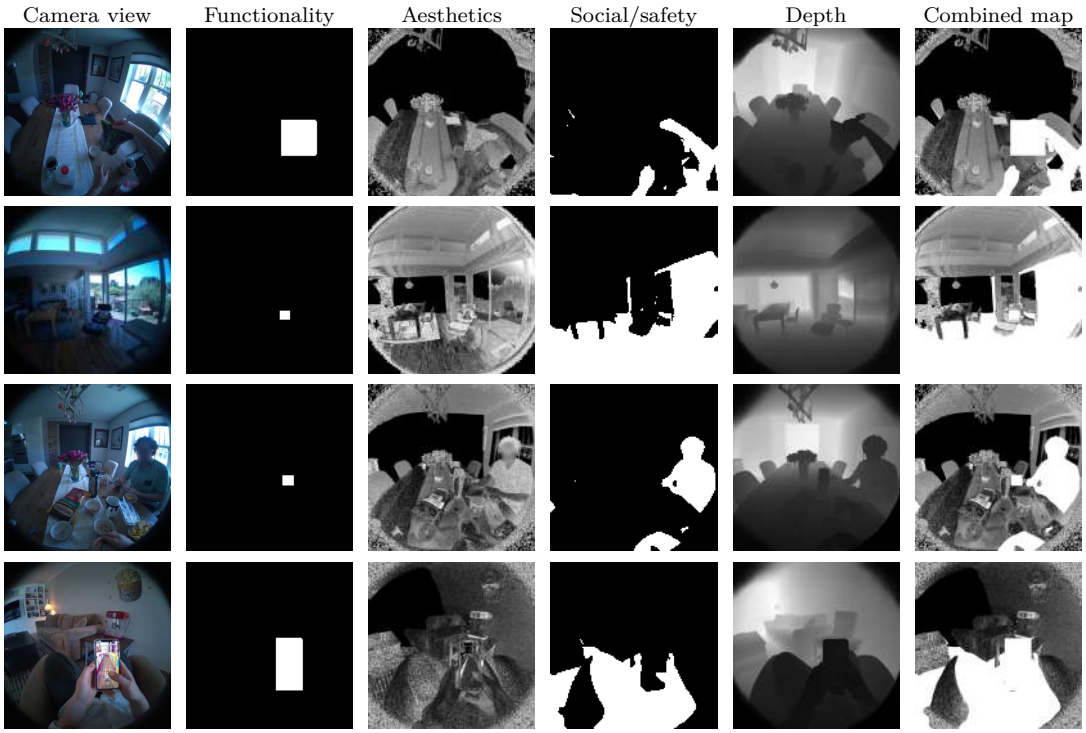


Figura 5.3: Esempi della pipeline di costruzione della combined map: dalla camera view originale alle mappe contestuali (functionality, aesthetics, social/safety e depth), fino alla combined map finale.

Nel task di *placement*, invece, la combined map viene utilizzata per assegnare uno score di interferenza alle celle della griglia *Set-of-Mark (SoM)*, consentendo di selezionare la posizione che minimizza l'impatto complessivo sull'esperienza dell'utente.

5.5 Analisi qualitativa dell'impatto della profondità e delle scelte di soglia

Questa sezione presenta un'analisi qualitativa del ruolo della profondità e delle scelte di soglia nella costruzione della *combined map*. L'obiettivo è valutare come tali componenti influenzino la selettività della mappa e la coerenza del processo decisionale con i principi percettivi e funzionali della Mixed Reality.

5.5.1 Impatto dell'integrazione della profondità nel task di visibility

L'introduzione della **mappa di profondità** consente di arricchire la rappresentazione della scena con una nozione di **struttura tridimensionale**, rendendo il processo decisionale più coerente con i requisiti della Mixed Reality, in cui la rilevanza percettiva di un'area dipende non solo dalla sua posizione nell'immagine, ma anche dalla **distanza dall'utente**. Un'analisi puramente bidimensionale tende infatti a penalizzare indiscriminatamente elementi che, pur presenti nel campo visivo, risultano percettivamente lontani e quindi poco rilevanti in termini di safety o interferenza funzionale.

L'effetto principale dell'integrazione della profondità è che alcune istanze che, in assenza della depth map, verrebbero etichettate come *bad placement*, vengono invece correttamente classificate come *good placement*. Ciò avviene perché la profondità consente di **attenuare il contributo** di regioni lontane, riducendo il carattere eccessivamente conservativo della valutazione.

Le figure riportate in Figura 5.4 mostrano esempi qualitativi di *combined map* ottenute senza e con integrazione della profondità. In assenza della depth map, la salienza tende a diffondersi su regioni estese, portando ad una penalizzazione globale della scena. Con l'integrazione della profondità, la mappa risulta invece più **localizzata e selettiva**, evidenziando principalmente le regioni effettivamente rilevanti per il placement dell'overlay.

In sintesi, l'integrazione della profondità consente di superare i limiti di una valutazione puramente 2D, producendo *combined map* meno conservative e riducendo i falsi *bad placement*, in modo coerente con i requisiti percettivi e funzionali della Mixed Reality.

5.5.2 Analisi dell'impatto della soglia sulla selezione delle regioni ammissibili

Un aspetto cruciale dell'integrazione della profondità riguarda la scelta della soglia utilizzata per discriminare le regioni considerate sufficientemente *vicine* all'utente e, di conseguenza, ammissibili per il placement dell'overlay. In questa fase sperimentale, i valori della depth map vengono **normalizzati nell'intervallo** [0,1] su base frame-wise, in modo da rendere la soglia indipendente dalla scala assoluta della stima di profondità e comparabile tra scene differenti. La soglia controlla quindi direttamente quanto il sistema sia *conservativo*: valori elevati tendono a escludere un numero maggiore di regioni, mentre valori più bassi rendono la selezione più permissiva.

In una prima fase sperimentale, è stata adottata una soglia pari a 0.7. Sebbene questa scelta garantisca un comportamento particolarmente prudente, l'analisi

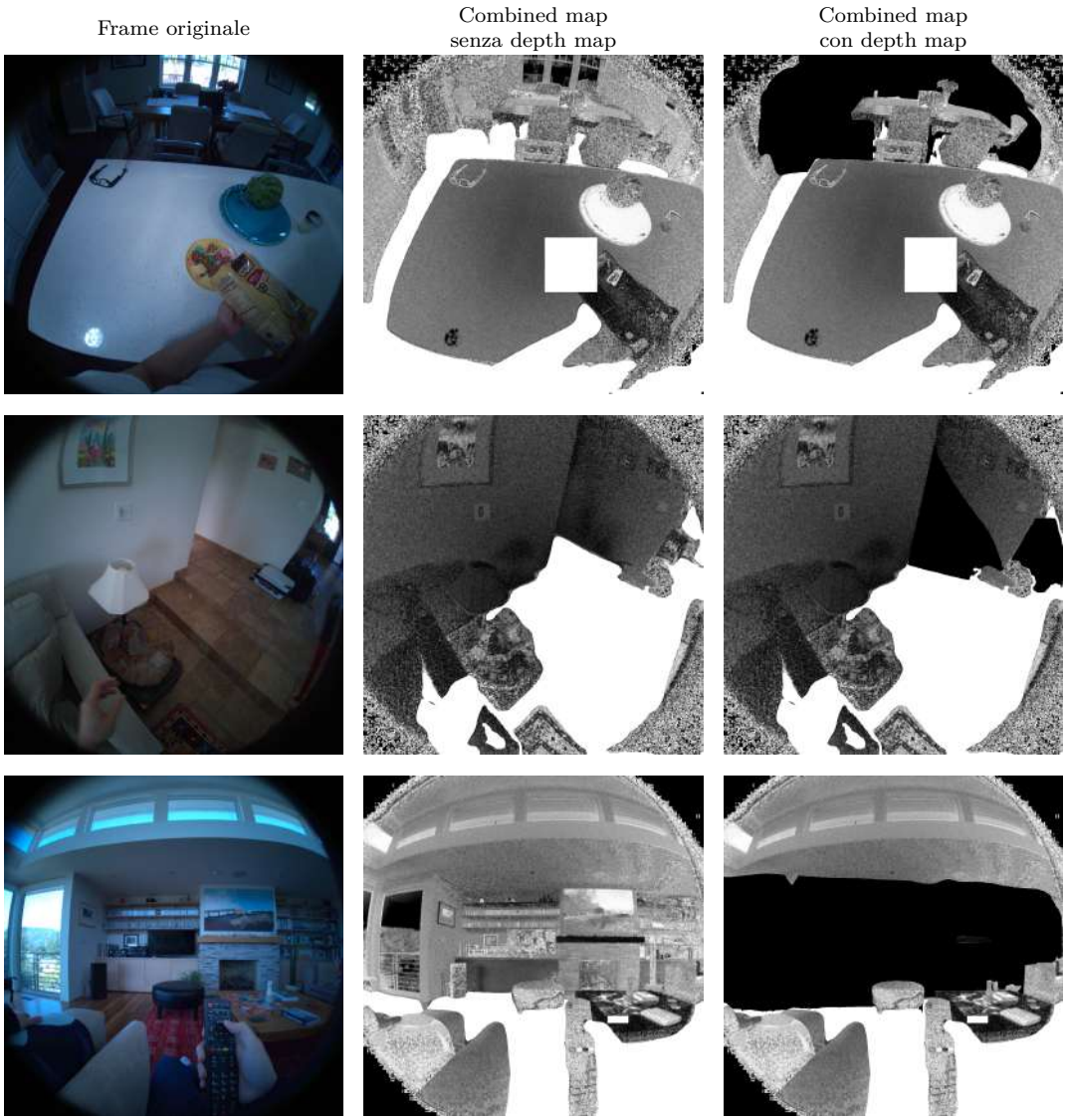


Figura 5.4: Esempi qualitativi di *combined map* ottenute in assenza e con integrazione della mappa di profondità, che evidenziano l'effetto della profondità sulla selettività delle regioni penalizzate.

qualitativa ha mostrato che essa risulta spesso **eccessivamente conservativa**, escludendo anche regioni che, dal punto di vista percettivo e funzionale, potrebbero essere considerate accettabili. In pratica, molte aree dello sfondo vengono penalizzate nonostante la loro distanza non comprometta realmente l'esperienza dell'utente.

Riducendo la soglia a 0.6, si osserva un miglioramento qualitativo significativo.

Con questa impostazione, il sistema mantiene invariati i vincoli legati alla safety e alla visibility, ma amplia lo spazio delle posizioni candidate, rendendo il placement più flessibile e utilizzabile. In altri termini, l'overlay dispone di un maggior numero di regioni valide senza introdurre interferenze rilevanti.

La Figura 5.5 riporta esempi comparativi di combined map ottenute con soglie pari a 0.7 e 0.6. È possibile osservare come una soglia meno restrittiva consenta di preservare aree visivamente lontane ma percettivamente innocue, ottenendo un compromesso più efficace tra robustezza del criterio di selezione e praticità del placement.

Nel complesso, questa analisi evidenzia come la scelta della soglia rappresenti un parametro chiave per bilanciare sicurezza e flessibilità. Un valore troppo elevato rischia di limitare inutilmente le opzioni di posizionamento dell'interfaccia, mentre una soglia moderatamente più permissiva consente di sfruttare meglio lo spazio visivo disponibile senza compromettere i requisiti di interferenza minima richiesti negli scenari di Mixed Reality.

5.5.3 Limiti della normalizzazione frame-wise della depth map

Un limite significativo dell'approccio basato sulla normalizzazione frame-wise della depth map riguarda la natura **puramente relativa** della distanza stimata. In particolare, negli scenari outdoor, la presenza di outlier estremi (ad esempio il cielo) amplia il range dei valori di profondità, comprimendo le distanze intermedie durante la normalizzazione min–max.

Di conseguenza, elementi che sono in realtà percettivamente lontani (come persone sullo sfondo, strade o facciate di edifici) possono essere erroneamente interpretati come vicini all'utente, influenzando negativamente la *combined map* e producendo penalizzazioni non desiderate del placement dell'overlay.

Esempio qualitativo

Un esempio concreto di questa problematica è illustrato nelle Figure 5.6. Nell'immagine di sinistra si osserva una persona posizionata sul lato sinistro dell'inquadratura (evidenziata dal cerchio arancione), che si trova ad una distanza considerevole dall'utente e che, dal punto di vista percettivo, **non dovrebbe penalizzare** il posizionamento dell'overlay in quella regione dell'immagine. Tuttavia, la presenza di elementi estremamente lontani, come il cielo, espande il range dei valori di profondità stimati dal modello.

A causa della normalizzazione min–max applicata per frame, le distanze intermedie vengono compresse verso valori bassi: di conseguenza, la persona sullo

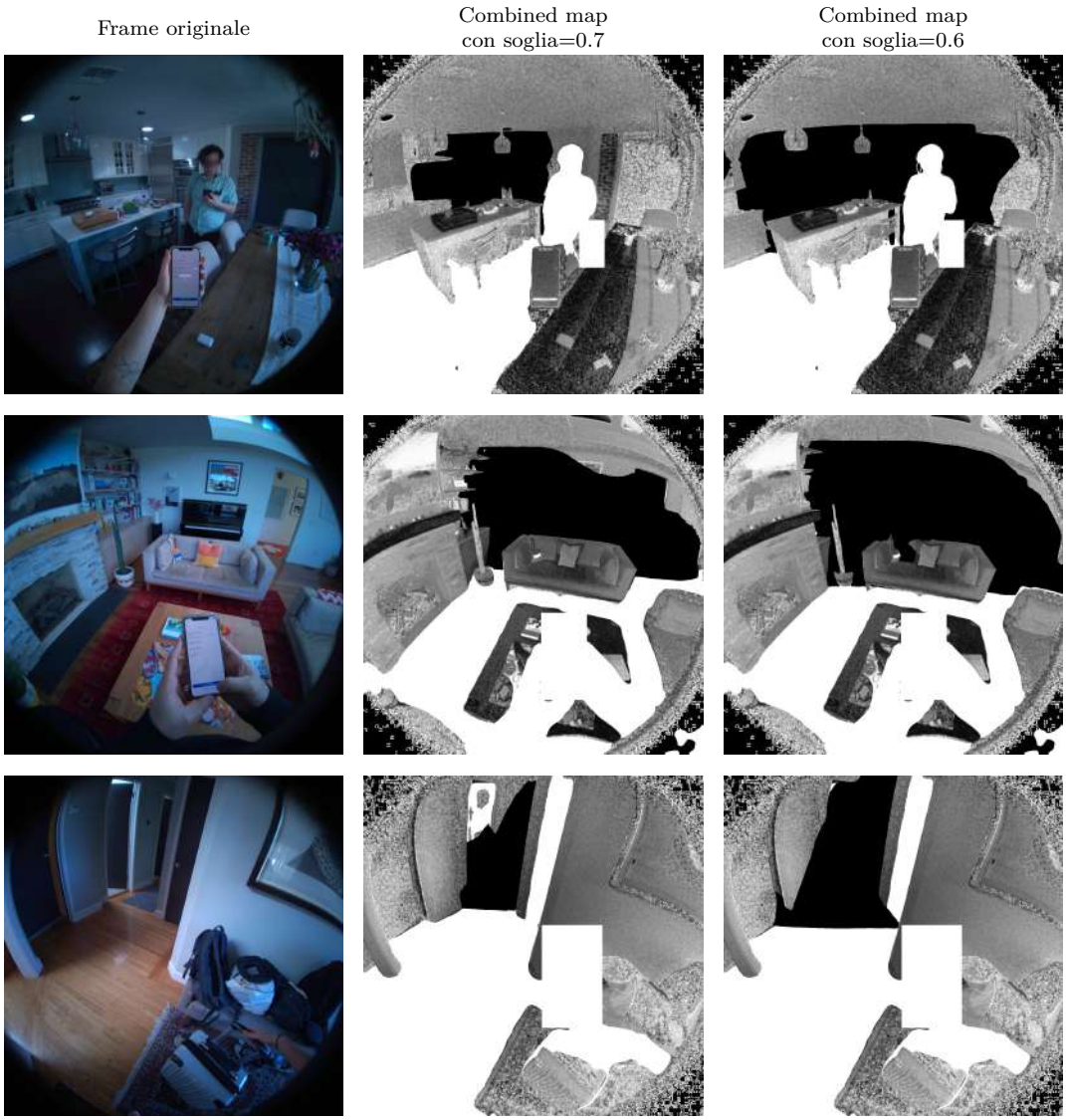


Figura 5.5: Esempi qualitativi di *combined map* ottenute con soglie pari a 0.7 e 0.6. Le figure evidenziano come l'adozione di una soglia meno conservativa (0.6) permetta un compromesso più efficace tra robustezza del criterio di selezione e utilizzabilità del placement, rispetto alla soglia più restrittiva (0.7).

sfondo viene mappata a valori di profondità relativamente ridotti e quindi **erroneamente interpretata come vicina all'utente**. Questo porta la regione corrispondente a essere classificata come *safety-critical* nella *combined map*, influenzando negativamente e in modo non desiderato la selezione delle posizioni valide per l'overlay.

(a) Frame originale con persona lontana evidenziata.

(b) Combined map ottenuta con normalizzazione frame-wise della depth map.

Figura 5.6: Esempio di fallimento della normalizzazione frame-wise della depth map in uno scenario outdoor: una persona percettivamente lontana viene erroneamente considerata vicina all'utente, influenzando la *combined map* e la selezione delle posizioni ammissibili per l'overlay.

5.5.4 Approccio basato sui percentili della depth map

Per rendere più robusta l'integrazione della profondità ed evitare le distorsioni introdotte dalla normalizzazione frame-wise, è stato adottato un approccio basato sui **percentili della depth map**. In particolare, anziché utilizzare una soglia assoluta sui valori normalizzati, i pixel appartenenti al **60% con i valori di profondità più bassi** vengono considerati come *vicini* all'utente, mentre i restanti pixel vengono trattati come *lontani*.

In termini intuitivi, questo significa che, per ogni frame, il sistema seleziona la porzione della scena che risulta relativamente più prossima al punto di vista dell'utente, indipendentemente dalla scala assoluta della profondità stimata. Tale strategia preserva l'ordinamento relativo dei valori di profondità prodotti dal modello e riduce l'influenza di outlier estremi (ad esempio superfici molto lontane), che possono comprimere la distribuzione dei valori nella normalizzazione min–max e alterare il comportamento della soglia.

Le figure comparative riportate in Figura 5.7 mostrano come l'approccio percentile-based produca *combined map* più **bilanciate e stabili**, sia in ambienti indoor sia outdoor. In particolare, regioni percettivamente lontane risultano correttamente

attenuate, evitando penalizzazioni indebite che limiterebbero inutilmente lo spazio di placement dell'overlay.

Nel complesso, l'adozione di una strategia basata sui percentili consente di definire la nozione di vicinanza in modo più robusto e coerente con la percezione umana, migliorando l'affidabilità e l'usabilità del processo di selezione delle posizioni dell'interfaccia.

Analisi qualitativa comparativa

Per valutare l'effetto dell'approccio basato sui percentili rispetto alla normalizzazione frame-wise della depth map, la Figura 5.7 riporta diversi esempi qualitativi. In ciascun caso, vengono confrontate: (i) la *combined map* ottenuta applicando una soglia pari a 0.6 sulla depth map normalizzata in [0,1] e (ii) la *combined map* ottenuta utilizzando il **percentile al 60%**, che considera come vicini all'utente i pixel appartenenti alla porzione più prossima della scena.

Negli scenari outdoor, l'approccio percentile-based evita che superfici strutturalmente lontane, come strade o facciate di edifici, vengano erroneamente interpretate come percettivamente vicine a causa della compressione dei valori introdotta dalla normalizzazione min–max. Di conseguenza, tali regioni non contribuiscono in modo significativo alla *combined map*, riducendo penalizzazioni non necessarie nel placement dell'overlay.

In modo analogo, negli scenari indoor, l'utilizzo dei percentili produce una separazione più stabile tra primo piano e sfondo. Elementi lontani, come finestre o superfici di fondo, vengono correttamente attenuati, mentre la presenza di persone o oggetti sullo sfondo non induce penalizzazioni spurie. Questo consente di mantenere il focus sulle regioni realmente rilevanti per l'interazione dell'utente.

Nel complesso, questi esempi mostrano come l'approccio basato sui percentili conduca a *combined map* più selettive e consistenti, migliorando la stabilità del processo decisionale e riducendo le distorsioni introdotte dalla normalizzazione frame-wise della profondità.

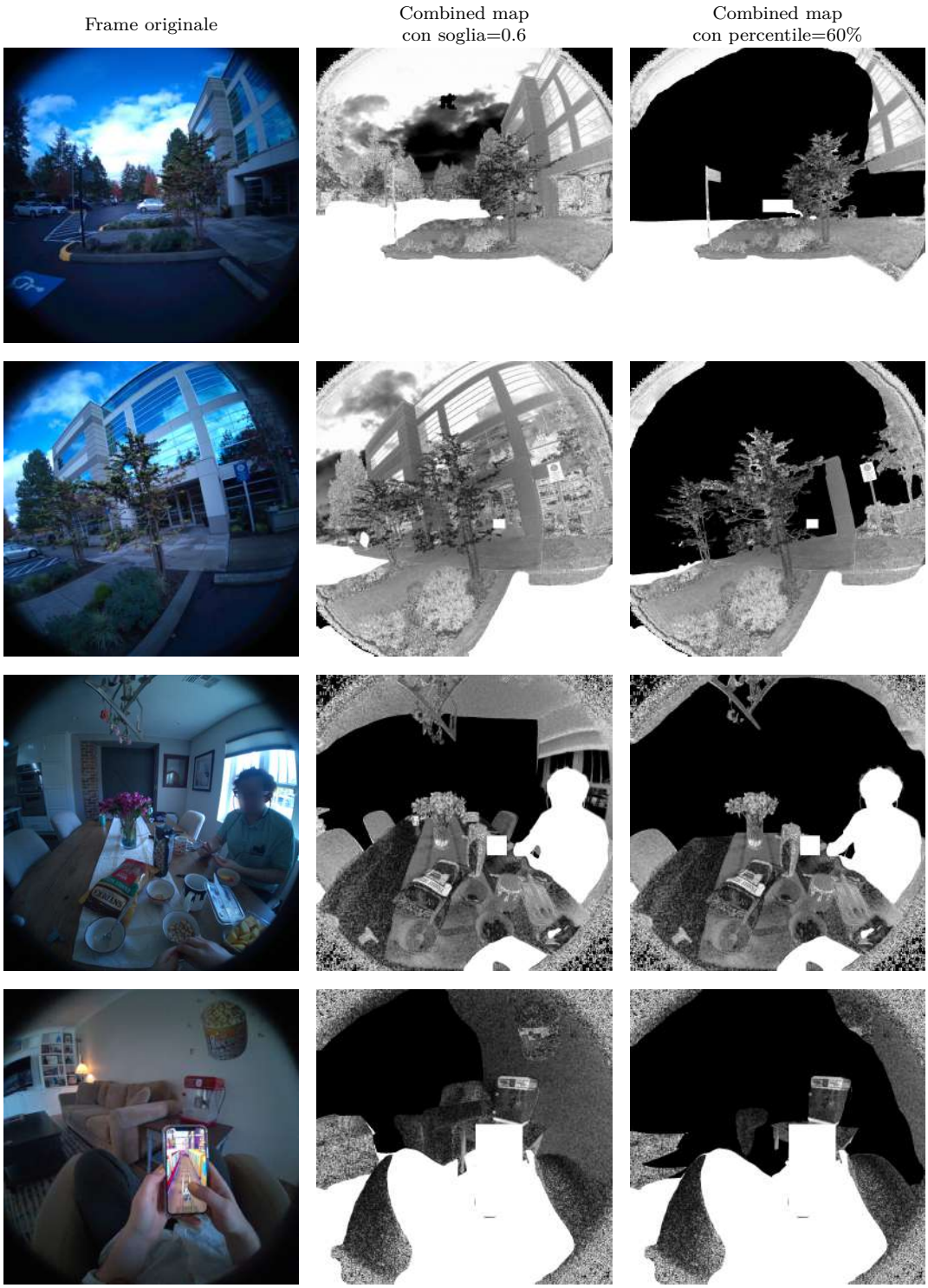


Figura 5.7: Confronto tra *combined map* ottenute con soglia fissa (0.6) e approccio percentile-based (60%) su uno scenario outdoor.

Capitolo 6

Costruzione del dataset supervisionato

6.1 Ruolo della combined map nella costruzione del dataset

Sulla base delle considerazioni discusse nel Capitolo precedente, la *combined map* rappresenta una sintesi compatta, robusta e interpretabile dei diversi fattori contestuali rilevanti per l'adattamento delle interfacce in scenari di Mixed Reality. Essa integra informazioni di funzionalità, sicurezza/accettabilità sociale, estetica e profondità in una singola rappresentazione densa, che esprime a livello pixel-wise quanto una regione della scena sia inadatta all'occlusione da parte di un overlay.

L'analisi qualitativa ha mostrato come l'integrazione esplicita della profondità, unitamente all'utilizzo di soglie basate sui percentili, consenta di ottenere mappe più selettive e coerenti con la percezione dell'utente. In particolare, tale formulazione permette di attenuare l'influenza di elementi distanti o di sfondo e di concentrarsi sulle regioni realmente rilevanti dal punto di vista percettivo e funzionale.

Nel seguito, la *combined map* viene utilizzata come **segnale unificato** per la generazione delle istanze di training dei due task considerati nel framework UiVLA. Sebbene i task di *visibility* e *placement* condividano la stessa pipeline di pre-processing e la medesima rappresentazione contestuale, essi differiscono nel modo in cui tale informazione viene sfruttata. Il task di *visibility* è formulato come un problema di classificazione binaria, in cui la combined map viene utilizzata per stimare l'accettabilità di una specifica sovrapposizione, accompagnata da una motivazione testuale esplicativa. Il task di *placement*, invece, richiede di individuare *dove* posizionare l'overlay, formulando la decisione come una selezione discreta della posizione ottimale all'interno di un insieme finito di candidati, definiti secondo il

paradigma *Set-of-Mark (SoM)*.

Le sezioni seguenti descrivono in dettaglio le procedure adottate per la costruzione dei due dataset, mettendo in evidenza le differenze metodologiche e le specificità di ciascun task, pur partendo da una base rappresentativa comune fornita dalla combined map.

6.2 Costruzione del dataset per il task di visibility

In questa sezione descriviamo il processo di costruzione del dataset utilizzato per il *task di visibility*, illustrando come le istanze vengano generate a partire da registrazioni video *first-person* e come le diverse **mappe contestuali** introdotte nelle sezioni precedenti siano impiegate per derivare sia la **label binaria** di visibility sia la corrispondente **motivazione testuale** (*reason*).

Ogni istanza del dataset è composta dai seguenti elementi:

1. **Due frame RGB**:
   - il frame originale della scena osservata dall'utente (*camera view*)
   - lo stesso frame con un **elemento di interfaccia sovrapposto** (*overlay view*)
2. **Una label binaria di visibility** $y \in \{0,1\}$:
   - $y = 1$ (*good placement*), se l'overlay non occlude contenuti funzionali, di sicurezza o visivamente rilevanti
   - $y = 0$ (*bad placement*), se l'overlay interferisce con contenuti funzionali, safety-critical o percettivamente salienti
3. **Una reason testuale**, ovvero una spiegazione in linguaggio naturale che giustifica la label assegnata. La motivazione è derivata in modo deterministico a partire dalle mappe contestuali e dai segnali computazionali del sistema, mantenendo un collegamento diretto tra la decisione di visibility e i fattori che l'hanno determinata

6.2.1 Derivazione delle istanze di visibility dalla combined map

A partire dalla *combined map*, il dataset per il task di *visibility* viene generato tramite una procedura automatica che consente di derivare in modo sistematico **istanze etichettate binariamente** (*good / bad placement*). L'approccio è ispirato a quello proposto da Dudley et al. [19] e sfrutta direttamente la semantica della combined map come misura di interferenza.

In primo luogo, la combined map viene suddivisa in una griglia regolare di **celle** aventi dimensioni pari a quelle dell'elemento di interfaccia da posizionare. L'overlay viene quindi fatto scorrere sull'immagine con uno **stride costante**, generando un insieme discreto di posizioni candidate. A ciascuna posizione (x, y) è associato un **costo di occlusione** $C(x, y)$, definito come il valore medio della salienza all'interno della regione occupata dall'overlay:

$$C(x, y) = \frac{1}{W \cdot H} \sum_{i=x}^{x+W-1} \sum_{j=y}^{y+H-1} S(i, j), \tag{6.1}$$

dove W e H indicano rispettivamente larghezza e altezza dell'elemento di interfaccia, e S è la combined map. Questo costo fornisce una stima continua dell'interferenza che l'overlay introdurrebbe se posizionato in quella regione.

Per ciascun frame, la generazione delle istanze avviene mediante una procedura di **campionamento controllato**, che determina congiuntamente la **label binaria** e la **posizione dell'overlay**. La label viene inizialmente selezionata in modo equiprobabile tra *good* (1) e *bad* (0), al fine di evitare sbilanciamenti nel dataset.

Nel caso di istanze negative (*bad placement*), la posizione dell'overlay viene selezionata tra le regioni caratterizzate da **elevato costo di occlusione**. In particolare, vengono considerate le posizioni appartenenti alla **coda superiore** della distribuzione dei costi (top 5%), imponendo al contempo una **soglia assoluta minima** sul valore del costo. Questo vincolo evita che, in scene complessivamente poco salienti, vengano etichettate come negative posizioni che in realtà non introducono interferenze significative.

In modo simmetrico, per le istanze positive (*good placement*), l'overlay viene posizionato in regioni a **bassa salienza**, campionando dalla **coda inferiore** della distribuzione dei costi (bottom 20%) e applicando una soglia assoluta massima. Tale scelta riduce il rischio di generare esempi ambigui, in cui la distinzione tra posizionamenti buoni e cattivi risulterebbe poco chiara.

Una volta individuato l'insieme delle posizioni candidate coerenti con la label selezionata, la posizione finale dell'overlay viene scelta **uniformemente a caso** tra di esse. Questo meccanismo introduce variabilità spaziale nelle istanze generate, evitando che il modello impari scorciatoie legate a posizioni preferenziali fisse.

Nel caso in cui nessuna posizione soddisfi i criteri di selezione (ad esempio a causa di una distribuzione dei costi particolarmente uniforme), viene adottata una strategia di *fallback* deterministica: si seleziona la posizione con costo massimo per le istanze negative o con costo minimo per le istanze positive. Questo garantisce che per ogni frame sia sempre possibile generare un'istanza valida.

Nel complesso, questa procedura consente di derivare automaticamente **label binarie di visibility** a partire dalla combined map, mantenendo il processo interpretabile e direttamente collegato ai fattori contestuali modellati nelle mappe

di salienza. La generazione delle motivazioni testuali (*reason*) associate alle istanze viene trattata separatamente ed è discussa nelle sezioni successive.

In Figura 6.1 sono mostrati alcuni esempi qualitativi del task di *visibility*, che illustrano sia casi di *bad placement*, in cui l'overlay occlude contenuti funzionali o safety-critical, sia casi di *good placement*, in cui l'elemento di interfaccia non interferisce con regioni rilevanti della scena.

(a) Istanza '**bad**': l'elemento UI copre l'oggetto funzionale (telefono cellulare) e copre le mani dell'utente, che sono rilevanti per la sicurezza e l'accettabilità sociale.

(b) Istanza '**good**': l'elemento dell'interfaccia utente non nasconde alcuna importante area funzionale, di sicurezza o estetica.

Figura 6.1: Esempi qualitativi del task di *visibility*. Ogni istanza è ottenuta valutando la *combined map* sulla regione occupata dall'overlay. Le istanze negative (*bad placement*) corrispondono a posizioni caratterizzate da elevato costo di occlusione, mentre le istanze positive (*good placement*) sono associate a regioni a bassa salienza media.

6.2.2 Motivazione dell'introduzione delle *reason* nel task di visibility

Nel contesto delle interfacce adattive per la Mixed Reality, una decisione automatica che stabilisce se una posizione dell'overlay sia accettabile o meno (*good / bad placement*) rappresenta solo una parte del problema. In particolare, l'approccio **XAIUI** [25] evidenzia come, nelle interfacce *context-aware*, le spiegazioni debbano

chiarire **quali fattori contestuali** hanno influenzato il comportamento del sistema e **in che modo** essi abbiano contribuito alla decisione finale.

Spiegazioni efficaci non devono essere generiche o ridondanti, ma **mirate, concise e semanticamente allineate** al contesto percettivo e funzionale dell'utente. In assenza di tali spiegazioni, anche decisioni corrette dal punto di vista tecnico possono risultare opache o essere percepite come errate, riducendo la fiducia dell'utente e l'utilizzabilità del sistema.

Alla luce di queste considerazioni, in questo lavoro il task di visibility viene esteso oltre la semplice classificazione binaria, introducendo per ciascuna istanza una ***reason***, ovvero una spiegazione testuale strutturata che esplicita *perché* una determinata posizione dell'overlay è stata giudicata accettabile o problematica. La *reason* non rappresenta una razionalizzazione post-hoc generica, ma una descrizione **direttamente derivata dai fattori contestuali** che hanno guidato la decisione.

In particolare, le *reason* sono costruite a partire dalle stesse **mappe computazionali** utilizzate per determinare la label di visibility (functionality, safety/social acceptability e aesthetics), garantendo una **coerenza semantica e causale** tra decisione e spiegazione. Questo approccio rispecchia uno dei principi chiave di XAIUI, secondo cui le spiegazioni devono essere *faithful*, ossia fedeli al processo decisionale interno del sistema.

Un ulteriore vantaggio di questa impostazione è la possibilità di rendere esplicita la **natura multifattoriale** della visibility. In molti casi, infatti, una posizione dell'overlay non risulta problematica per un singolo motivo dominante, ma per una combinazione di fattori funzionali, di sicurezza o estetici. La *reason* consente di comunicare tale complessità in forma compatta e interpretabile.

Infine, l'introduzione delle *reason* ha un impatto diretto anche sul processo di apprendimento del modello. Ogni istanza del dataset include esplicitamente sia la **label** sia la **spiegazione associata**, consentendo di addestrare il modello a produrre *decisione e motivazione in un unico passaggio autoregressivo*. Questo rende il sistema più stabile, interpretabile e allineato ai principi delle interfacce adattive spiegabili promossi dal framework XAIUI.

6.2.3 Generazione delle *reason* a partire dalle mappe contestuali

Le *reason* associate alle istanze del task di *visibility* vengono generate tramite una procedura **deterministica e rule-based**, progettata per tradurre la logica numerica utilizzata nella costruzione delle label in una spiegazione testuale **fedele, compatta e interpretabile**. In linea con i principi di *faithfulness* nell'ambito della Explainable AI for User Interfaces (XAIUI), la spiegazione non è prodotta da un modello linguistico separato, ma è **direttamente derivata dagli stessi**

segnali computazionali che hanno determinato la decisione binaria di *good* o *bad placement*.

Una volta selezionata la posizione dell'overlay (i, j) e assegnata la label di visibility, vengono analizzati localmente i segnali contestuali all'interno della regione occupata dall'elemento di interfaccia. In particolare, vengono calcolati i seguenti valori:

- il **functionality score** e il relativo **oggetto funzionale**, ottenuti combinando le informazioni di eye-gaze, object detection e la *Functionality Map*, che indicano se l'overlay interferisce con l'oggetto su cui è focalizzata l'attenzione dell'utente
- il **safety and social acceptability score** e l'elenco degli **oggetti safety/social** effettivamente coperti dall'overlay, derivati dalla *Safety and Social Acceptability Map*
- l'**aesthetics score**, che misura il grado di salienza percettiva della regione occlusa sulla base della *Aesthetic Map*
- una **soglia decisionale** coerente con quella utilizzata nella generazione della label di visibility, che consente di confrontare in modo consistente i punteggi calcolati

Questi valori costituiscono l'input di una funzione deterministica di generazione della *reason*, che produce una breve descrizione testuale della decisione presa:

```
reason = _compute_reason(
    label,
    threshold,
    functionality_score,
    safety_and_social_score,
    aesthetics_score,
    functionality_object,
    covered_safety_objects
)
```

Questo schema evidenzia come la *reason* sia una **funzione pura dei punteggi contestuali** e delle informazioni semantiche derivate dalle mappe, senza accesso diretto all'immagine o al contenuto visivo dell'overlay. Di conseguenza, la spiegazione è allineata alla decisione del sistema: ogni affermazione contenuta nella *reason* può essere ricondotta a uno o più segnali esplicitamente presenti nella rappresentazione contestuale. Tale proprietà rende le spiegazioni non solo interpretabili per l'utente, ma anche verificabili e coerenti con il comportamento del modello.

6.2.4 Identificazione degli oggetti rilevanti per safety e accettabilità sociale

Nel caso in cui la decisione di *visibility* sia influenzata da considerazioni di sicurezza o accettabilità sociale, la *reason* può fare riferimento a **oggetti semantici specifici** presenti nella scena (ad esempio *person*, *bicycle*, *car*). L'identificazione di tali oggetti è fondamentale per produrre spiegazioni interpretabili e concrete, evitando motivazioni astratte o generiche.

A tal fine, per ciascuna istanza viene verificato quali classi semantiche risultino **effettivamente intersecate dalla regione occupata dall'overlay** all'interno della mappa di segmentazione. In pratica, l'area dell'immagine coperta dall'overlay viene confrontata con la segmentazione semantica della scena, consentendo di individuare quali oggetti rilevanti per la sicurezza o per l'interazione sociale siano parzialmente o totalmente occlusi.

Dal punto di vista operativo, questa procedura può essere descritta come segue: a partire dalla segmentazione semantica dell'intero frame, si estrae la porzione corrispondente alla bounding box dell'overlay e si raccolgono le classi presenti in tale regione. L'elenco finale viene quindi filtrato mantenendo esclusivamente le classi appartenenti all'insieme di oggetti considerati critici per la sicurezza e l'accettabilità sociale definite nella Sezione 5.2.3.

Il risultato è una lista di etichette semantiche direttamente interpretabili, che può essere utilizzata per ancorare la spiegazione a entità concrete della scena. Questo consente di generare *reason* del tipo:

> *"The overlay covers **person** and **bicycle**, which are relevant for user safety and social acceptability."*

In questo modo, la motivazione testuale risulta strettamente legata al contenuto visivo effettivamente occluso dall'overlay, rafforzando la trasparenza e la verificabilità del processo decisionale. Ogni oggetto menzionato nella *reason* può infatti essere ricondotto a una specifica regione della segmentazione semantica, mantenendo un chiaro legame causale tra percezione, decisione e spiegazione.

6.2.5 Reason per istanze accettabili (*label = 1*)

Quando un'istanza viene etichettata come accettabile (*label = 1*), la generazione della *reason* segue una logica semplice e completamente deterministica. In particolare, la procedura verifica che **tutti i punteggi contestuali** associati alla regione occupata dall'overlay — funzionalità, sicurezza/accettabilità sociale ed estetica — risultino inferiori alla soglia utilizzata per la decisione di visibility.

Se questa condizione è soddisfatta, significa che l'overlay non interferisce in modo significativo con alcuna dimensione contestuale rilevante. In tal caso, viene

restituita una *reason* standardizzata che esplicita chiaramente l'assenza di occlusioni problematiche:

> *"The overlay does not occlude any important functional, safety, or aesthetic region."*

Questa formulazione è volutamente concisa e generica, in quanto riflette una situazione in cui nessun fattore contestuale domina la decisione e l'overlay risulta globalmente compatibile con la scena osservata.

Nel raro caso in cui l'istanza sia etichettata come accettabile ma uno o più punteggi risultino prossimi alla soglia (ad esempio a causa di condizioni borderline o di politiche conservative nella selezione delle label), viene restituita una formulazione alternativa più cauta, che segnala l'accettabilità dell'overlay nel contesto della politica decisionale adottata:

> *"The overlay is acceptable given the current policy and scores."*

In Figura 6.2 sono riportati esempi qualitativi di istanze con label positiva. In entrambi i casi mostrati, l'overlay non occlude regioni funzionali, di sicurezza/accettabilità sociale o esteticamente salienti secondo le mappe contestuali, e la *reason* generata risulta coerente con il contenuto visivo e con i punteggi calcolati.

6.2.6 Reason per istanze non accettabili (*label = 0*)

Nel caso di istanze non accettabili (*label = 0*), la *reason* viene generata individuando il **fattore (o i fattori) contestuali responsabili della violazione**. La procedura segue una logica deterministica basata su una **gerarchia di priorità**, progettata per riflettere l'impatto reale dell'occlusione sull'esperienza dell'utente:

1. **Funzionalità**
2. **Sicurezza e accettabilità sociale**
3. **Estetica**

Questa gerarchia riflette il fatto che interferire con l'attività dell'utente o con elementi critici per la sicurezza è generalmente più grave rispetto a una degradazione puramente estetica.

Caso 1: fattore dominante. Se uno dei punteggi contestuali supera una soglia alta di interferenza, esso viene considerato il **fattore dominante** e determina direttamente la spiegazione testuale.

Quando il fattore dominante è la **funzionalità**, la *reason* esplicita che l'overlay occlude l'oggetto con cui l'utente sta interagendo o che sta osservando attivamente:

(a) Reason: The overlay does not occlude any important functional, safety, or aesthetic region.

(b) Reason: The overlay does not occlude any important functional, safety, or aesthetic region.

Figura 6.2: Esempi qualitativi di *reason* deterministica per istanze accettabili (label 1). In entrambi i casi, l'overlay non occlude regioni funzionali, di sicurezza/accettabilità sociale o esteticamente salienti secondo le mappe contestuali.

> *"The overlay covers a functional element (cell phone) that the user is interacting with."*

Se invece il fattore dominante riguarda la **sicurezza o l'accettabilità sociale**, la spiegazione menziona esplicitamente gli oggetti semantici coinvolti (ad es. persone o veicoli), ancorando la motivazione a entità chiaramente interpretabili:

> *"The overlay covers person, which is relevant for user safety and social acceptability."*

Quando più fattori risultano contemporaneamente rilevanti (ad esempio funzionalità e sicurezza), la procedura produce una *reason* combinata che riflette entrambe le cause, mantenendo comunque la priorità definita.

Caso 2: contributi multipli senza fattore dominante. Nel caso in cui nessun singolo fattore superi la soglia alta, ma siano presenti più contributi moderati, la procedura costruisce una spiegazione **composita**, selezionando al massimo due fattori principali. Anche in questo caso, l'ordine segue la gerarchia di priorità definita.

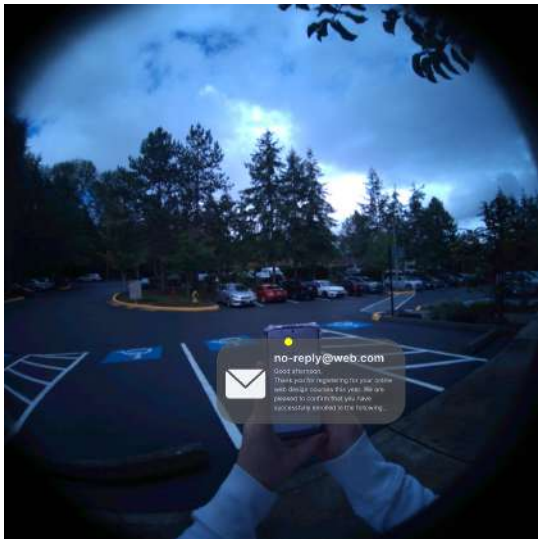
Un esempio tipico di *reason* composita è il seguente:

"The overlay covers a part of functional element (cell phone) and covers a high aesthetic region."

Questa formulazione indica che l'overlay non è accettabile non per un'unica violazione grave, ma per la combinazione di più interferenze di entità moderata.

Fallback esplicativo. Infine, nel raro caso in cui l'overlay risulti non accettabile ma nessun fattore possa essere identificato in modo chiaro come dominante o contributivo, viene restituita una spiegazione più generica, che segnala un impatto complessivo sull'usabilità senza attribuire la decisione a una singola causa specifica.

In Figura 6.3 sono riportati esempi qualitativi di *reason* associate a istanze non accettabili. In tutti i casi, la spiegazione testuale riflette in modo diretto e verificabile i fattori contestuali attivi nella regione occupata dall'overlay, garantendo coerenza tra segnali percettivi, decisione binaria e motivazione fornita all'utente.



(a) Reason: *The overlay covers a part of functional element (cell phone) and covers person's hands and road, which are relevant for safety and social acceptability.*

(b) Reason: *The overlay covers a part of functional element (cup) and covers floor and person, which are relevant for safety and social acceptability.*

Figura 6.3: Esempi qualitativi di *reason* deterministica per istanze non accettabili (label 0). Le motivazioni riflettono i fattori dominanti che determinano la violazione (funzionalità e/o sicurezza/accettabilità sociale e/o estetica), derivati dalle mappe contestuali nella regione occupata dall'overlay.

6.3 Costruzione del dataset per il task di placement

In questa sezione descriviamo il processo di costruzione del dataset impiegato per il **task di placement**, illustrando come le istanze siano generate a partire da registrazioni video *first-person* e come le **mappe contestuali**, già introdotte per il task di visibility, vengano riutilizzate per supportare la selezione della **posizione ottimale dell'overlay** all'interno della scena.

Il task di placement è formulato come un problema di **selezione discreta**: dato un frame della scena reale, il modello è chiamato ad individuare **una cella** che rappresenta la posizione più appropriata per collocare l'elemento di interfaccia, minimizzando l'interferenza con contenuti funzionali, di sicurezza o percettivamente rilevanti.

Ogni istanza del dataset è composta dai seguenti elementi:

1. **Due frame RGB**:
   - il frame originale della scena osservata dall'utente (*camera view*)
   - il frame della scena osservata dall'utente, arricchito con un insieme discreto di **posizioni candidate per il placement**, ciascuna contrassegnata da un **identificatore alfabetico** visualizzato direttamente nell'immagine. Ogni identificatore rappresenta il **centro di posizionamento** di un elemento di interfaccia di dimensione fissa, secondo il paradigma *Set-of-Mark (SoM)* [26], e costituisce una possibile alternativa per il posizionamento dell'overlay all'interno della scena
2. **Label primaria di placement**: un indice discreto che identifica la **posizione candidata selezionata come ottimale** per l'overlay. Tale posizione è determinata in modo deterministico come quella che minimizza il **costo di interferenza** stimato a partire dalla *combined map*, e rappresenta la soluzione di riferimento (*ground truth*) per il task di placement
3. **Label secondarie di placement**: un insieme di indici associati a posizioni candidate che presentano un **costo di interferenza maggiore rispetto alla soluzione ottimale**, ma che risultano comunque **semanticamente e percettivamente plausibili** come alternative per il posizionamento dell'overlay all'interno della stessa scena. L'inclusione di queste etichette secondarie consente di modellare la natura intrinsecamente **ambigua e multi-soluzione** del problema di placement in scenari di Mixed Reality, rendendo il dataset più robusto e tollerante a predizioni vicine all'ottimo
4. **Insieme completo delle posizioni candidate**: per ciascun frame è fornito l'insieme di tutte le posizioni candidate considerate per il task di placement,

ordinate secondo il **costo di interferenza decrescente**. La prima posizione dell'insieme corrisponde alla cella con il massimo livello di interferenza, mentre l'ultima rappresenta la **soluzione ottimale** (*ground truth*). Ogni posizione candidata è descritta mediante un **identificatore simbolico** (lettera) e il relativo **score medio**, calcolato sulla patch di dimensione pari all'elemento di interfaccia del nostro sistema e centrata sulla posizione candidata stessa

Questa rappresentazione consente di trasformare il problema di placement da una formulazione spaziale continua a uno **spazio discreto e finito di alternative**, rendendo il task più strutturato e direttamente compatibile con modelli vision–language autoregressivi. Al contempo, l'utilizzo della griglia SoM mantiene un forte legame tra la decisione del modello e la struttura spaziale della scena, facilitando l'**interpretabilità** del comportamento del sistema e il collegamento tra predizione e contesto visivo.

6.3.1 Generazione delle posizioni candidate per il placement

A partire dalla **combined map**, il dataset per il **task di placement** viene generato trasformando una valutazione continua della salienza in una **scelta discreta di posizione candidata**, formulata come un problema di selezione tra un insieme finito di alternative. A differenza del task di visibility, che valuta l'accettabilità di una singola posizione dell'overlay, il placement richiede di individuare **la posizione ottimale** per il posizionamento dell'elemento di interfaccia all'interno dell'intero frame, tenendo conto dell'interazione globale con la scena.

Nel framework proposto, la generazione delle posizioni candidate rappresenta un passaggio cruciale, poiché definisce lo *spazio delle alternative* su cui il modello è chiamato ad operare.

Nel corso di questo lavoro sono stati esplorati **quattro diversi approcci** per la generazione delle posizioni candidate, con l'obiettivo di analizzare trade-off differenti in termini di copertura della scena, coerenza semantica, costo computazionale e compatibilità con modelli vision–language:

1. **Griglia fissa** (*fixed grid*)
2. **Superpixel SLIC**
3. **Segmentazione con Segment Anything (SAM)**
4. **Segmentazione semantica con Mask2Former**

Nelle Sezioni successive ciascun approccio viene descritto separatamente, evidenziandone le caratteristiche principali, le modalità di generazione delle posizioni candidate e le relative implicazioni per il task di placement.

6.3.2 Griglia fissa di celle

Il primo metodo adottato per la generazione delle posizioni candidate è basato su una **griglia regolare di celle non sovrapposte**, ciascuna avente dimensioni identiche a quelle dell'elemento di interfaccia da posizionare. La griglia viene costruita in modo da **coprire uniformemente la maggiore porzione possibile dell'immagine**, evitando sovrapposizioni tra celle e introducendo **gap uniformi** sia tra celle adiacenti sia rispetto ai bordi dell'immagine. In questo modo, le posizioni candidate risultano **equidistanti** e distribuite regolarmente sull'intera area visiva.

Questo approccio implementa in modo diretto il paradigma *Set-of-Mark (SoM)*: ogni cella rappresenta una possibile posizione candidata per il placement ed è identificata da un **indice univoco**, visualizzato al centro della cella nell'immagine fornita in input al modello. La decisione di placement può quindi essere formulata come la selezione esplicita di uno di questi identificatori.

Questo processo è illustrato nella Figura 6.4, che mostra esempi qualitativi di frame suddivisi mediante una griglia di celle non sovrapposte ed equidistanti. In ciascun esempio viene riportato, a sinistra, il frame originale e, a destra, la corrispondente rappresentazione con la griglia SoM sovrapposta, in cui ogni cella costituisce una possibile posizione candidata per il posizionamento dell'overlay.

Dal punto di vista computazionale, il metodo a griglia fissa risulta **estremamente efficiente**: la generazione delle posizioni candidate richiede esclusivamente **operazioni geometriche elementari** e semplici **aggregazioni locali** sulla *combined map*. Tale semplicità lo rende pienamente compatibile con **scenari real-time** e con dispositivi caratterizzati da risorse computazionali limitate. Inoltre, la struttura regolare della griglia garantisce **facilità di interpretazione** e una gestione sperimentale particolarmente semplice e controllabile.

Tuttavia, questa efficienza è ottenuta a fronte di una **formulazione fortemente semplificata** del problema di placement, che risulta **poco adeguata per scenari di Mixed Reality complessi**. L'adozione di una griglia regolare introduce infatti una **discretizzazione rigida dello spazio**, completamente indipendente dalla struttura semantica della scena e dalla geometria degli oggetti presenti. Di conseguenza, le posizioni candidate non sono informate dal contesto visivo e possono risultare **disallineate rispetto a superfici funzionali, oggetti rilevanti o regioni di interazione dell'utente**.

In contesti di Mixed Reality, in cui il posizionamento dell'interfaccia deve rispettare vincoli spaziali, funzionali e percettivi fortemente dipendenti dalla scena osservata, tale assenza di consapevolezza semantica rappresenta una limitazione significativa. Le celle della griglia possono infatti includere regioni **semanticamente eterogenee**, attraversare i confini di oggetti o superfici, o proporre posizioni che, pur minimizzando un costo medio di salienza, risultano **innaturali o poco**

plausibili dal punto di vista dell'esperienza utente.

Di conseguenza, sebbene l'approccio a griglia fissa costituisca un utile *baseline* e un riferimento computazionalmente efficiente, esso non è sufficiente a catturare la complessità e la variabilità dei contesti reali tipici della Mixed Reality, motivando l'esplorazione di strategie alternative per la generazione di posizioni candidate maggiormente **adattive e sensibili al contenuto della scena**.

6.3.3 Superpixel SLIC

Come alternativa alla discretizzazione geometrica tramite griglia fissa, è stato esplorato l'utilizzo di **superpixel generati mediante l'algoritmo SLIC (Simple Linear Iterative Clustering)** [32], implementato tramite la funzione `slic` della libreria `skimage.segmentation`. SLIC è un algoritmo di segmentazione che suddivide l'immagine in un insieme di **regioni compatte (superpixel)** raggruppando pixel simili in termini di **colore** e **vicinanza spaziale**. L'obiettivo è ottenere regioni visivamente omogenee e localmente coerenti, che tendono a seguire i contorni di oggetti e superfici presenti nella scena.

In questo approccio, ciascun superpixel può essere interpretato come una **regione candidata adattiva**, potenzialmente più allineata alla struttura visiva della scena rispetto a una discretizzazione puramente geometrica basata su celle regolari.

In linea teorica, l'approccio SLIC consente di ottenere regioni che seguono più fedelmente i contorni di oggetti e superfici, riducendo la probabilità che una singola regione includa aree semanticamente eterogenee. Tuttavia, nei primi esperimenti condotti imponendo un numero molto ridotto di regioni candidate (**12 superpixel**), i risultati ottenuti si sono rivelati **fortemente simili a quelli di una griglia fissa**. In particolare, come mostrato nella seconda colonna della Figura 6.5, il metodo tende a convergere sistematicamente a **circa 9 regioni effettive**, disposte in maniera regolare e approssimativamente rettangolare, producendo una suddivisione dello spazio visivo che perde gran parte dell'adattività semantica attesa.

Questo comportamento non è riconducibile a errori di implementazione, ma rappresenta un **effetto intrinseco dell'algoritmo SLIC**. Il parametro che controlla il numero di superpixel definisce infatti un *obiettivo desiderato* e non un vincolo rigido: durante il processo iterativo di clustering, alcuni centroidi inizializzati possono non acquisire pixel o essere assorbiti da cluster più stabili, riducendo il numero di regioni finali. Tale fenomeno risulta particolarmente evidente quando il numero di superpixel richiesto è **molto basso rispetto alla risoluzione dell'immagine** (nel nostro caso 1408×1408), poiché il termine di regolarizzazione spaziale tende a dominare, favorendo una suddivisione uniforme dello spazio e producendo regioni compatte e regolari.

Per valutare se tale effetto fosse imputabile alla ridotta granularità imposta, il numero di superpixel è stato aumentato a **30 regioni**. In questo setting, SLIC

inizia effettivamente a produrre regioni **più aderenti ai contorni di oggetti e superfici**, riducendo l'effetto "a griglia" e introducendo una maggiore adattività alla struttura visiva della scena. Tuttavia, anche in questo caso, come illustrato nella terza colonna della Figura 6.5, il numero di regioni effettivamente ottenute rimane **inferiore al valore target**, attestandosi nei nostri esperimenti ad un massimo di circa **24 superpixel**. Questo risultato conferma la difficoltà di controllare in modo preciso la cardinalità dell'insieme di posizioni candidate quando si utilizzano approcci di clustering non vincolati. Dal punto di vista computazionale, SLIC rappresenta una soluzione **relativamente efficiente** rispetto a metodi di segmentazione semantica più complessi. Nei nostri esperimenti, considerando il tempo complessivo di segmentazione e visualizzazione delle regioni (*compute + draw*) eseguito su CPU, il tempo medio per frame passa da circa **592 ms** nel caso di 12 regioni a **797.47 ms** nel caso di 30 regioni. Nonostante ciò, il metodo non è stato adottato nel framework finale: la **limitata controllabilità del numero di regioni**, la loro **forte dipendenza dalla struttura visiva locale** e la difficoltà nel garantire una **coerenza strutturale tra istanze diverse** rendono SLIC poco adatto al task di placement, che richiede un insieme di posizioni candidate **stabile, riproducibile e direttamente compatibile** con l'approccio Set-of-Mark e con modelli vision–language autoregressivi.

6.3.4 Segmentazione con Segment Anything (SAM)

È stato inoltre esplorato l'utilizzo di **Segment Anything (SAM)** [33], un modello di segmentazione *foundation* che genera maschere a partire da **prompt spaziali**. Nel contesto di SAM, il termine *prompt-based* indica che la segmentazione non è eseguita in modo completamente libero sull'intera immagine, ma è **condizionata da un input di guida** (ad esempio punti, bounding box o maschere iniziali) che specifica *dove* il modello deve concentrarsi.

Nel nostro setup, SAM è stato impiegato con **point prompt** generati automaticamente: sull'immagine viene campionata una **griglia regolare di punti equidistanti** e ciascun punto viene fornito al modello come prompt *foreground*. Per ogni punto (x, y), SAM produce una o più maschere candidate che segmentano la regione più coerente con il vincolo spaziale imposto dal prompt, restituendo quindi un insieme di regioni potenzialmente rilevanti distribuite sull'intera scena.

Le maschere generate vengono quindi sottoposte a una fase di **post-processing** che include: (i) un **filtro sull'area minima** per rimuovere regioni troppo piccole o spurie e (ii) una procedura di **deduplicazione basata sull'overlap** per eliminare maschere ridondanti. Al fine di mantenere coerenza con il paradigma *Set-of-Mark (SoM)* adottato nel task di placement, il numero finale di regioni candidate viene limitato a un **massimo di 12 maschere per frame**, evitando di introdurre

un numero eccessivo di alternative che potrebbe rendere il processo decisionale instabile per il modello vision–language.

Dal punto di vista qualitativo, questo approccio produce regioni **altamente aderenti ai contorni di oggetti e superfici**, offrendo una segmentazione di elevata qualità, indipendente da una tassonomia semantica predefinita e capace di catturare con precisione la geometria effettiva della scena. In questo senso, SAM rappresenta una soluzione estremamente flessibile e generale per la generazione di regioni candidate. Come mostrato negli esempi qualitativi in Figura 6.6, le maschere risultanti seguono in modo accurato la struttura visiva della scena e i contorni degli oggetti presenti nel frame.

Tuttavia, a fronte dell'elevata qualità delle segmentazioni prodotte, SAM risulta **di gran lunga il metodo più oneroso dal punto di vista computazionale** tra quelli considerati. Nei nostri esperimenti, considerando l'intera pipeline di **segmentazione, selezione delle regioni, visualizzazione e salvataggio** (*compute + draw*), il tempo medio di elaborazione per frame, eseguito su CPU, è pari a circa **82.3 secondi**.

L'utilizzo di accelerazione GPU consente una riduzione significativa del tempo di elaborazione; tuttavia, anche in questo caso, il costo computazionale rimane **incompatibile con vincoli real-time** e con i requisiti di reattività tipici degli scenari di Mixed Reality e dei dispositivi *head-mounted*, che richiedono tempi di risposta dell'ordine dei millisecondi.

Di conseguenza, nonostante l'elevata qualità e generalità delle regioni prodotte, l'approccio basato su Segment Anything non è stato adottato nel framework finale. Il suo **elevato costo computazionale** lo rende infatti poco adatto al task di placement nel contesto applicativo considerato, dove efficienza, stabilità temporale e scalabilità rappresentano requisiti fondamentali.

6.3.5 Segmentazione semantica con Mask2Former

Infine, come ulteriore alternativa per la generazione delle posizioni candidate, è stato esplorato l'utilizzo di **Mask2Former** [29] come modello di **segmentazione semantica per-pixel**, sfruttando i pesi pre-addestrati disponibili su Hugging Face (checkpoint *ADE20K semantic*, backbone basato su **Swin Transformer**). Mask2Former rappresenta uno stato dell'arte nella segmentazione semantica e consente di ottenere una suddivisione della scena guidata direttamente dalla **struttura semantica**, senza introdurre griglie regolari o vincoli geometrici espliciti.

Oltre a essere utilizzato per la generazione delle regioni candidate nel task di placement, Mask2Former è impiegato anche nella costruzione della *combined map*. In particolare, le predizioni semantiche per-pixel sono sfruttate per individuare e localizzare **elementi rilevanti dal punto di vista della sicurezza e dell'accettabilità sociale** (ad esempio persone, superfici funzionali o oggetti potenzialmente

critici), che contribuiscono alla mappa di *safety and social acceptability*, come descritto nella Sezione 5.2.3.

Dato un frame RGB, il modello produce una **mappa semantica** $H \times W$ in cui a ciascun pixel è associato un identificatore di classe. Per contenere il costo computazionale, il frame originale (1408×1408) viene inizialmente **ridimensionato** ad una risoluzione inferiore durante l'inferenza. La mappa semantica ottenuta viene quindi riportata alla risoluzione originale mediante interpolazione *nearest neighbor*, preservando l'identità discreta delle etichette semantiche ed evitando la generazione di classi spurie lungo i bordi delle regioni.

Estrazione delle regioni candidate

Poiché Mask2Former fornisce una classificazione semantica per-pixel, è necessario trasformare la mappa semantica in un insieme discreto di **regioni candidate** per il task di placement. A tal fine, per ciascuna classe semantica c viene costruita una maschera binaria

$$M_c(x) = \mathbb{I}[y(x) = c],$$

dove x indica una posizione spaziale (pixel) dell'immagine, $y(x)$ rappresenta l'etichetta semantica assegnata dal modello al pixel x, e c è una delle classi semantiche del vocabolario di Mask2Former. La funzione indicatrice $\mathbb{I}[\cdot]$ assume valore 1 quando la condizione è verificata e 0 altrimenti.

Su ciascuna maschera M_c viene quindi applicata un'analisi delle **componenti connesse**. Ogni componente connessa (*blob*) rappresenta una regione spazialmente continua e semanticamente coerente, consentendo di distinguere istanze multiple della stessa classe presenti in posizioni disgiunte dell'immagine.

I blob estratti vengono filtrati imponendo vincoli minimi sull'**area in pixel** e sull'**area della bounding box**, al fine di rimuovere regioni troppo piccole, frammentate o instabili. Inoltre, per evitare una proliferazione eccessiva di candidati, viene imposto un limite sia sul numero massimo di componenti per classe sia sul numero totale di regioni per frame.

La scelta di operare a livello di blob è motivata dalla natura del task di placement e dall'adozione del paradigma *Set-of-Mark (SoM)*: l'obiettivo non è individuare dettagli locali, ma selezionare **regioni sufficientemente estese, stabili e semanticamente coerenti** da poter ospitare un elemento di interfaccia. Regioni di grandi dimensioni, come pareti o superfici uniformi, costituiscono candidati naturali per il placement, mentre una segmentazione eccessivamente fine risulterebbe poco informativa e aumenterebbe l'ambiguità della scelta.

Integrazione con il paradigma Set-of-Mark

Come introdotto nella Sezione 4.3, il paradigma *Set-of-Mark (SoM)* fornisce una formulazione discreta del task di placement basata su un insieme finito di regioni candidate esplicitamente indicizzate. In questa sezione, tale paradigma viene istanziato operativamente nel processo di costruzione del dataset, definendo la modalità con cui le regioni semanticamente coerenti vengono trasformate in posizioni candidate selezionabili dal modello.

Per rendere la rappresentazione compatibile con l'approccio SoM, le regioni semanticamente coerenti non vengono etichettate con il nome della classe restituito dal modello, ma con un **identificatore alfabetico causale**. Questa scelta è motivata da considerazioni di **generalizzazione tra task** e dalla necessità di evitare fenomeni di **shortcut learning**. In particolare, l'utilizzo di identificatori numerici nel task di placement potrebbe indurre il modello a privilegiare risposte frequenti (ad esempio "1") senza un reale ragionamento sul contenuto visivo. Tale rischio è amplificato dal fatto che il valore "1" coincide con la label positiva del task di visibility, aumentando la possibilità di confusione tra i due compiti.

L'adozione di identificatori alfabetici consente di disaccoppiare esplicitamente lo spazio delle azioni del placement da quello delle label binarie della visibility, riducendo il rischio di scorciatoie spurie e favorendo un apprendimento più robusto e semantico.

L'identificatore viene posizionato in una posizione centrale della regione, determinata combinando il centroide geometrico della componente con un vincolo di distanza minima dai bordi, garantendo leggibilità e stabilità visiva. In questo modo, ciascuna regione diventa una **posizione candidata indicizzata**, direttamente selezionabile dal modello vision–language.

Considerazioni computazionali

Dal punto di vista computazionale, Mask2Former risulta più oneroso rispetto ai metodi puramente geometrici o basati su clustering locale, ma significativamente più efficiente rispetto a modelli di segmentazione *foundation* di carattere generale, come Segment Anything. L'utilizzo di risoluzioni di inferenza intermedie consente di ottenere un **compromesso efficace tra qualità semantica della segmentazione e costo computazionale**, rendendo Mask2Former utilizzabile anche in scenari *near real-time* in presenza di adeguate risorse hardware.

La Tabella 6.1 riporta i tempi medi di inferenza ottenuti al variare del backbone e della risoluzione di input, sia su CPU sia su GPU. I risultati mostrano come il backbone *tiny* garantisca le prestazioni migliori in termini di velocità, ma a fronte di una **minore stabilità e precisione dei contorni semantici**, soprattutto a risoluzioni ridotte. Al contrario, i backbone *base* e *large* offrono una segmentazione

più accurata, ma risultano **non compatibili con scenari real-time**, in particolare su CPU e, in parte, anche su GPU quando si utilizza la risoluzione nativa.

Nel nostro setting sperimentale, la configurazione **small** con risoluzione di inferenza pari a 768×768 emerge come la soluzione più bilanciata: essa fornisce regioni semanticamente coerenti e contorni sufficientemente stabili, mantenendo al contempo **tempi di inferenza accettabili** sia su CPU sia su GPU (in regime *real-time*). Per questi motivi, tale configurazione è stata adottata come riferimento negli esperimenti successivi sul task di placement per la generazione delle posizioni candidate nei diversi frame.

Backbone	**Resize**	**Mean ms/frame (CPU)**	**FPS (CPU)**	**Mean ms/frame (GPU)**	**FPS (GPU)**
tiny	512×512	494.00	2.02	22.00	45.45
tiny	768×768	575.97	1.74	24.32	41.11
tiny	1408×1408	926.86	1.08	34.27	29.18
small	512×512	581.46	1.72	24.48	40.85
small	**768×768**	**678.89**	**1.47**	**27.24**	**36.71**
small	1408×1408	1119.88	0.89	39.74	25.17
base	512×512	699.98	1.43	27.84	35.92
base	768×768	792.17	1.26	30.45	32.84
base	1408×1408	1474.84	0.68	49.80	20.08
large	512×512	948.35	1.05	34.88	28.67
large	768×768	1037.62	0.96	37.41	26.73
large	1408×1408	1569.21	0.64	52.47	19.06

Tabella 6.1: Tempi medi di inferenza di Mask2Former al variare del backbone e della risoluzione di input, su CPU e GPU. I valori riportati corrispondono alla media calcolata su 30 frame selezionati uniformemente da 6 video differenti. La configurazione **small** a 768×768 (evidenziata) rappresenta il miglior compromesso tra qualità semantica e costo computazionale nel nostro setting.

Esempi qualitativi

Nella Figura 6.7 sono mostrati esempi qualitativi dell'approccio basato su Mask2Former, che illustrano: (i) il frame RGB originale, (ii) le regioni semantiche colorate individuate dal modello e (iii) la rappresentazione finale in stile Set-of-Mark, in cui ciascuna regione candidata è identificata da un indice alfabetico.

6.3.6 Confronto tra approcci e scelta del metodo finale

L'analisi comparativa dei metodi esplorati per la generazione delle posizioni candidate mostra che **Mask2Former rappresenta la soluzione più equilibrata ed efficace** nel contesto del task di placement considerato in questo lavoro. Sebbene ciascun approccio presenti vantaggi specifici, emergono limitazioni strutturali che ne riducono l'idoneità quando l'obiettivo è ottenere un insieme di candidati **semanticamente significativo**, **interpretabile** e **compatibile con modelli vision–language autoregressivi**.

L'approccio basato su **griglia fissa** costituisce un riferimento semplice e computazionalmente efficiente: garantisce **stabilità temporale**, **controllo completo** sulla cardinalità dei candidati e costi trascurabili di generazione. Tuttavia, esso introduce una discretizzazione **puramente geometrica** della scena, indipendente dalla struttura semantica e dalla geometria degli oggetti presenti. Di conseguenza, le celle candidate possono attraversare confini tra superfici diverse e includere regioni **semanticamente eterogenee**, riducendo la plausibilità delle posizioni proposte e rendendo la decisione del modello fortemente dipendente dalla risoluzione e dall'allineamento della griglia.

L'utilizzo di **superpixel SLIC** introduce una prima forma di adattività alla struttura visiva locale, producendo regioni compatte guidate da similarità cromatica e prossimità spaziale. Tuttavia, nei setting coerenti con il paradigma SoM (ovvero con un numero ridotto di regioni candidate), SLIC tende a convergere verso una partizione ancora **simile a una griglia regolare**, limitando i benefici attesi in termini di allineamento ai contorni degli oggetti. Incrementare il numero di superpixel migliora la fedeltà delle regioni alla struttura dell'immagine, ma comporta una **maggiore frammentazione**, una **controllabilità limitata** della cardinalità effettiva dei candidati e un incremento del costo computazionale, rendendo l'approccio meno stabile e meno adatto a una formulazione discreta strutturata.

L'approccio basato su **Segment Anything (SAM)** si distingue invece per l'elevata qualità delle maschere prodotte: le regioni risultano spesso **altamente aderenti ai contorni** e indipendenti da una tassonomia semantica predefinita, offrendo un meccanismo estremamente flessibile per la generazione di candidati. Ciononostante, i tempi di elaborazione osservati rendono SAM **impraticabile** sia per la costruzione del dataset su larga scala, sia per qualsiasi prospettiva di utilizzo *near real-time* nel contesto di dispositivi di Mixed Reality.

Alla luce di tali considerazioni, **Mask2Former** si colloca come il miglior compromesso tra qualità delle regioni e sostenibilità computazionale. Rispetto a griglia fissa e SLIC, Mask2Former consente di derivare posizioni candidate **guidate dalla semantica della scena**, corrispondenti a superfici e aree visivamente coerenti (ad esempio pareti, pavimenti o ampie regioni di sfondo), riducendo la probabilità

di includere porzioni semanticamente eterogenee all'interno dello stesso candidato. Inoltre, l'integrazione con SoM risulta naturale: a ciascuna regione può essere associato un **identificatore discreto** direttamente selezionabile dal modello, mantenendo un chiaro legame tra output testuale e contenuto visivo.

Un ulteriore vantaggio rilevante riguarda la **controllabilità** dell'insieme di candidati: tramite filtri geometrici (ad esempio vincoli su area minima e bounding box), è possibile regolare la qualità e la dimensione delle regioni, evitando frammentazioni eccessive e mantenendo un insieme di alternative compatibile con la natura discreta del task. Infine, come discusso nelle sezioni precedenti e sintetizzato in Tabella 6.1, l'uso di backbone leggeri e risoluzioni intermedie rende Mask2Former **computazionalmente sostenibile**, risultando nettamente più efficiente rispetto a metodi *foundation* generalisti come SAM, pur mantenendo una segmentazione semanticamente informata.

Per questi motivi, nel prosieguo del lavoro **Mask2Former** viene adottato come **metodo di riferimento** per la generazione delle posizioni candidate nel task di placement e per la costruzione del dataset associato.

6.3.7 Derivazione delle istanze di placement dalla combined map

Una volta ottenuta la rappresentazione finale in stile *Set-of-Mark (SoM)* basata sulle regioni semantiche individuate da Mask2Former, ciascuna regione candidata viene trasformata in una **patch di placement** utilizzata per la costruzione del dataset. In particolare, per ogni regione indicizzata viene definita una **patch rettangolare** avente dimensioni pari a quelle dell'elemento di interfaccia da posizionare, con **centro coincidente con l'identificatore alfabetico** associato alla regione.

Qualora la patch così definita ecceda i **bordi dell'immagine**, la sua posizione viene **traslata rigidamente** fino a rientrare completamente nei limiti del frame. Questa operazione garantisce che l'area utilizzata per il calcolo dello score di interferenza sia sempre valida e interamente contenuta nell'immagine.

Questa scelta mantiene una formulazione coerente con il task di placement: il modello non è chiamato a selezionare una regione in senso astratto, bensì a valutare l'effetto di un overlay **effettivamente posizionato** in una porzione specifica della scena. Tale porzione è associata a un oggetto o a una superficie riconosciuta dal modello, soddisfacendo i requisiti richiesti da un contesto di Mixed Reality. In questo modo, anche regioni semanticamente ampie o geometricamente irregolari vengono ricondotte a una rappresentazione **uniforme e confrontabile** tra le diverse istanze del dataset, risultando direttamente compatibili con il meccanismo di scoring basato sulla *combined map*.

Per ciascuna patch candidata viene quindi calcolato uno **score di interferenza**, definito come il **valore medio della salienza** all'interno della regione coperta

dalla patch sulla combined map S. Formalmente, dato un elemento UI di larghezza W e altezza H, lo score associato alla patch centrata in (i, j) è definito come:

$$C(i,j) = \frac{1}{W \cdot H} \sum_{x=i}^{i+W-1} \sum_{y=j}^{j+H-1} S(x,y). \tag{6.2}$$

Questo score fornisce una stima continua del grado di interferenza che l'overlay introdurrebbe se posizionato in quella specifica area dell'immagine: **valori più bassi indicano posizioni meno intrusive e più sicure**, mentre valori elevati corrispondono a patch che coprono contenuti funzionali, di sicurezza o visivamente salienti.

Nel contesto del task di placement, la **label primaria** dell'istanza viene determinata in modo deterministico selezionando la patch con **score minimo** tra tutte le posizioni candidate associate alle regioni semantiche:

$$(i^*, j^*) = \arg\min_{(i,j)} C(i,j). \tag{6.3}$$

L'**identificatore alfabetico** corrispondente alla patch selezionata viene quindi utilizzato come **label discreta di placement**. Poiché l'immagine fornita in input al modello include la rappresentazione SoM con identificatori alfabetici sovrapposti alle regioni, la predizione viene formulata come una **scelta esplicita tra posizioni candidate**.

Al fine di rendere il dataset più robusto e tollerante a predizioni prossime all'ottimo, oltre alla posizione con score minimo vengono inoltre salvate anche le **due posizioni successive migliori** (*top-3*), ordinate in base allo score di interferenza. In questo modo, una predizione che non coincida esattamente con la *ground truth* primaria, ma selezioni una delle alternative a basso costo, può essere considerata comunque **accettabile dal punto di vista semantico e funzionale**. Tale scelta riflette la natura intrinsecamente **ambigua e multi-soluzione** del problema di placement in scenari reali di Mixed Reality, in cui spesso esistono più posizioni equivalenti o quasi-equivalenti.

Nel complesso, questa procedura consente di derivare automaticamente **istanze di placement interpretabili, riproducibili e semanticamente fondate**, direttamente ancorate alla stessa *combined map* impiegata per il task di visibility. In tal modo, il placement eredita gli stessi vincoli contestuali (funzionalità, sicurezza, accettabilità sociale, estetica e profondità), ma li utilizza per risolvere un problema più complesso di **selezione spaziale discreta**, anziché di classificazione binaria.

Per rendere più chiaro e interpretabile il processo di derivazione delle patch di placement a partire dalle regioni semantiche, vengono inoltre salvati frame di supporto che visualizzano esplicitamente il risultato delle trasformazioni descritte. In particolare, a partire dalla rappresentazione SoM finale, per ciascuna regione candidata viene sovrapposta la **patch rettangolare effettivamente utilizzata**

per il calcolo dello score, avente dimensioni pari a quelle dell'elemento di interfaccia e **centrata sull'identificatore alfabetico della regione**. Le patch vengono evidenziate tramite un **contorno rosso**, consentendo di visualizzare in modo diretto quale porzione dell'immagine contribuisca al calcolo del costo di interferenza per ciascuna regione.

Il risultato di questo processo è mostrato in Figura 6.8, che affianca per ciascun esempio il frame originale, la rappresentazione SoM finale e la corrispondente visualizzazione con le patch di placement sovrapposte.

Grazie all'utilizzo di un identificatore alfabetico assegnato in maniera casuale, con un meccanismo di fallback per evitare la presenza di lettere duplicate all'interno dello stesso frame, e non di un identificatore alfabetico incrementale, è stato possibile ottenere una distribuzione fortemente bilanciata delle celle su tutti i dataset. Questa scelta progettuale ha permesso di evitare fenomeni di *overfitting* e *shortcut learning*, in cui il modello avrebbe potuto imparare a predire sistematicamente la lettera più frequente indipendentemente dal contenuto visivo della scena.

Nel nostro contesto sperimentale, le distribuzioni percentuali delle lettere risultano infatti quasi uniformi su tutti i dataset:

- **Training dataset**: le dieci lettere sono distribuite in modo omogeneo, con valori compresi approssimativamente tra l'8.8% e l'11.0% per ciascuna cella
- **Validation dataset**: la distribuzione rimane bilanciata, con percentuali comprese tra circa il 7.4% e l'11.9%
- **Test dataset**: anche in questo caso non emergono sbilanciamenti significativi, con percentuali che variano indicativamente tra il 6.9% e il 13.3%

Questa uniformità conferma che nessuna cella risulta statisticamente dominante e che il modello è costretto a basare le proprie decisioni su informazioni visive e contestuali piuttosto che su bias statistici del dataset.

Per quanto riguarda invece lo **score medio associato alla ground truth** (ossia alla cella corretta) sui diversi dataset, si osservano i seguenti valori medi:

- **Training dataset**: score medio pari a circa 18.1
- **Validation dataset**: score medio pari a circa 18.5
- **Test dataset**: score medio pari a circa 14.3

Considerando che lo score appartiene all'intervallo $[0, 255]$, questi valori relativamente bassi indicano che il modello viene addestrato e valutato su un dataset stabile, efficiente e robusto, caratterizzato dalla presenza di posizioni a bassa interferenza visiva per l'utente.

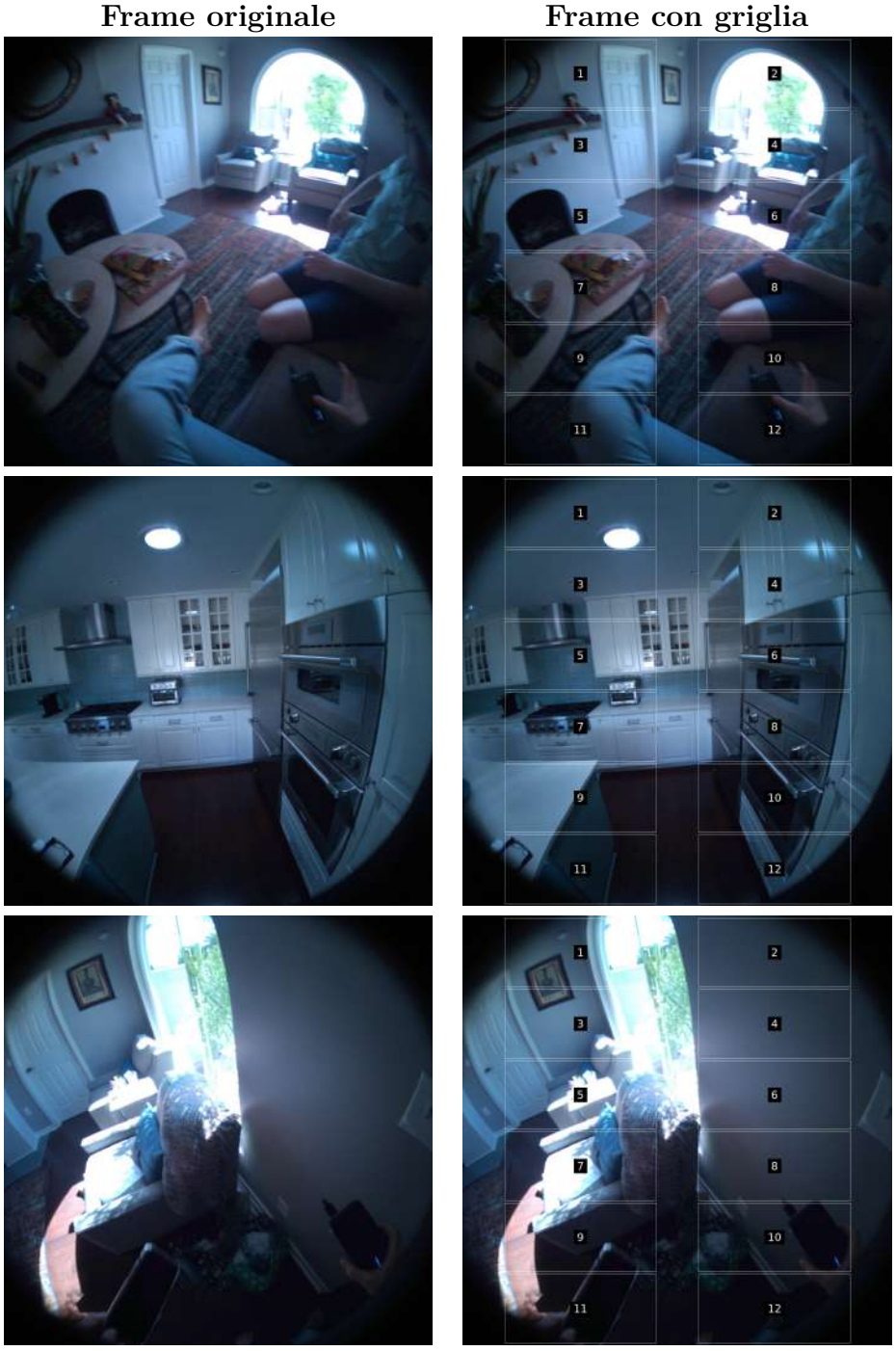


Figura 6.4: Esempi di generazione delle posizioni candidate tramite griglia fissa. A sinistra sono mostrati i frame originali, mentre a destra le corrispondenti rappresentazioni con griglia Set-of-Mark (SoM), in cui ciascuna cella identifica una possibile posizione candidata per il placement dell'overlay.

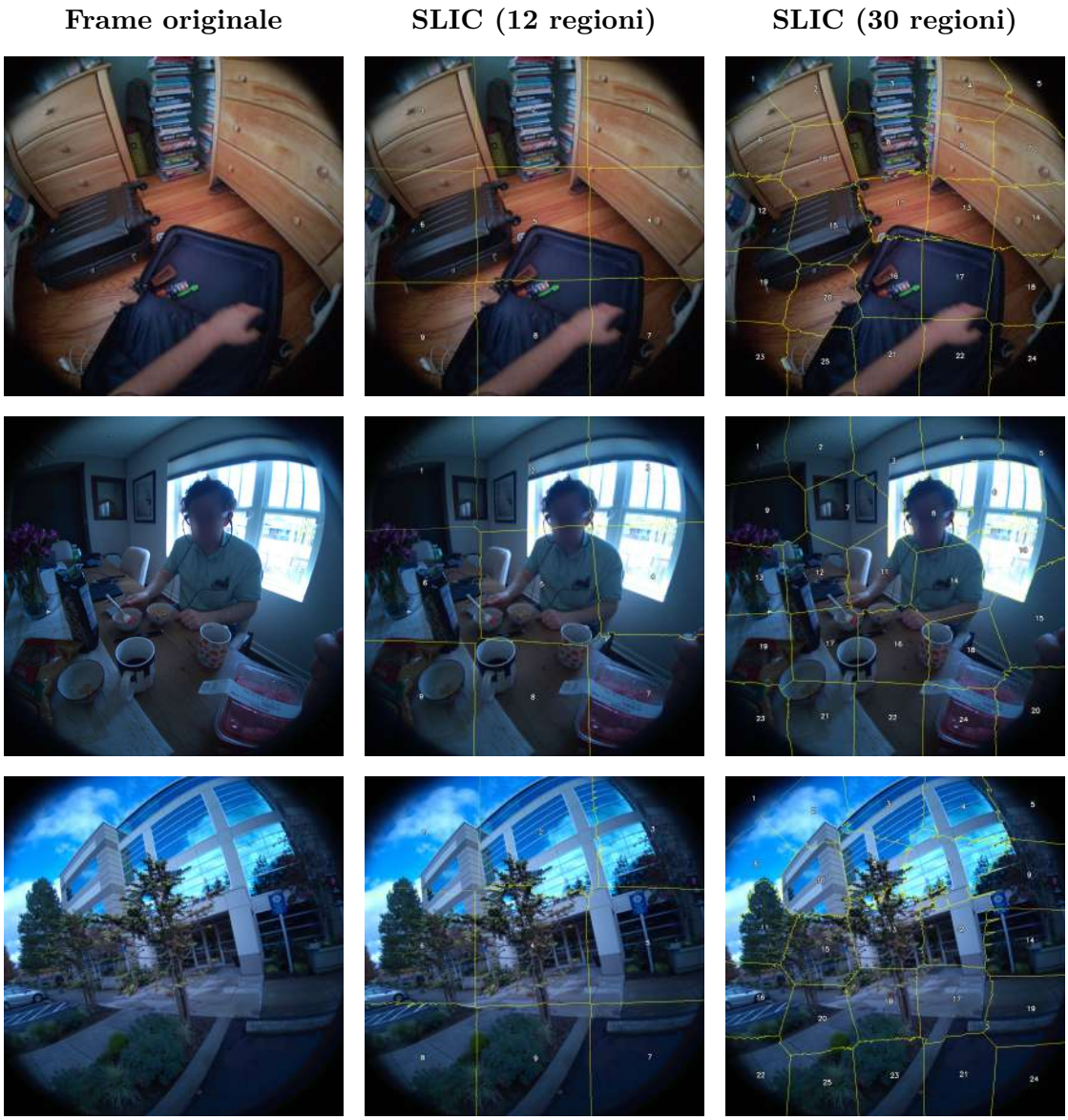


Figura 6.5: Confronto qualitativo tra frame originale e segmentazione tramite superpixel SLIC con diverso numero di regioni. La prima colonna mostra il frame originale, mentre la seconda e la terza colonna riportano rispettivamente la segmentazione SLIC con 12 e 30 regioni.

Figura 6.6: Esempi qualitativi della segmentazione *prompt-based* ottenuta con Segment Anything (SAM).

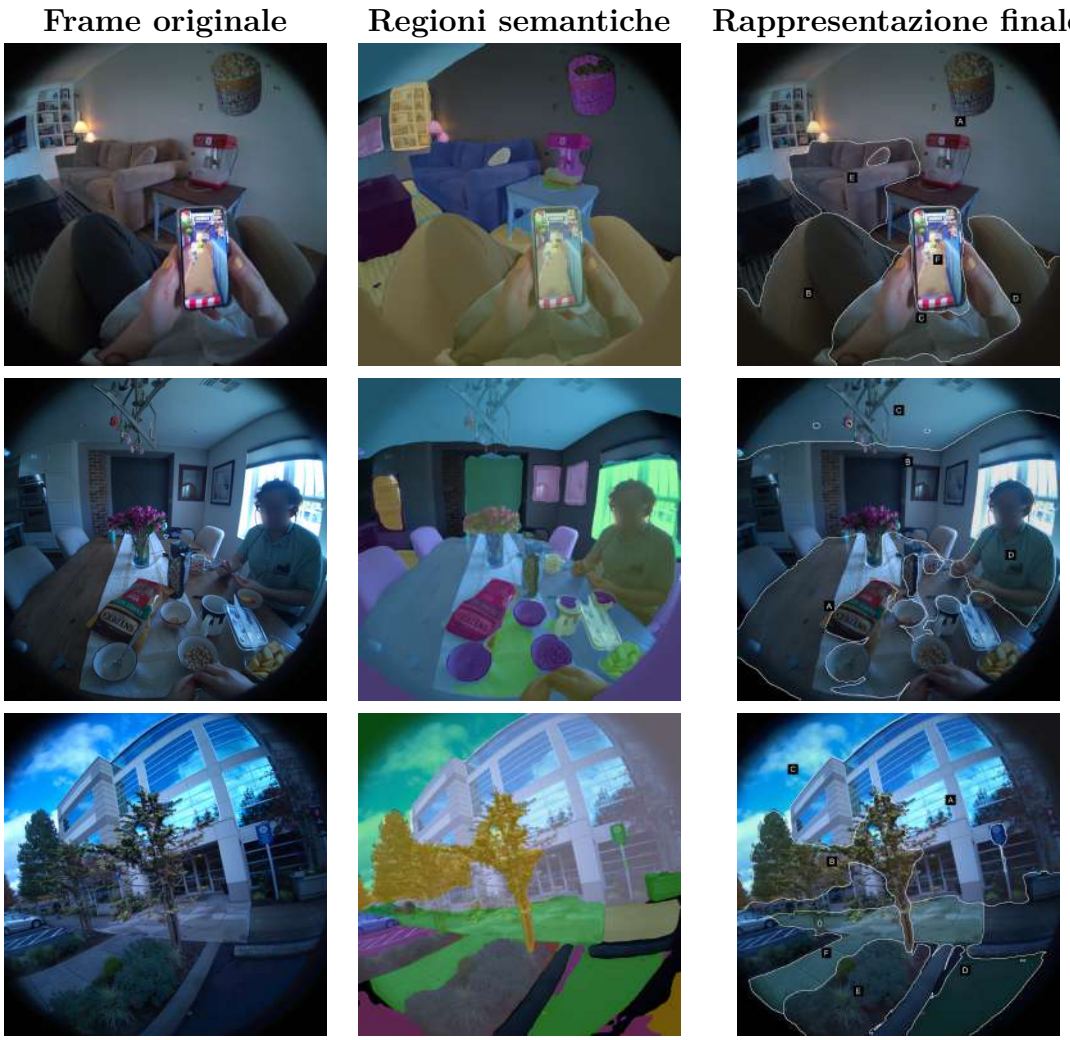


Figura 6.7: Esempi qualitativi dell'approccio di placement basato su Mask2Former. Da sinistra a destra: frame originale, regioni semantiche individuate dal modello e rappresentazione finale in stile Set-of-Mark con regioni candidate indicizzate.

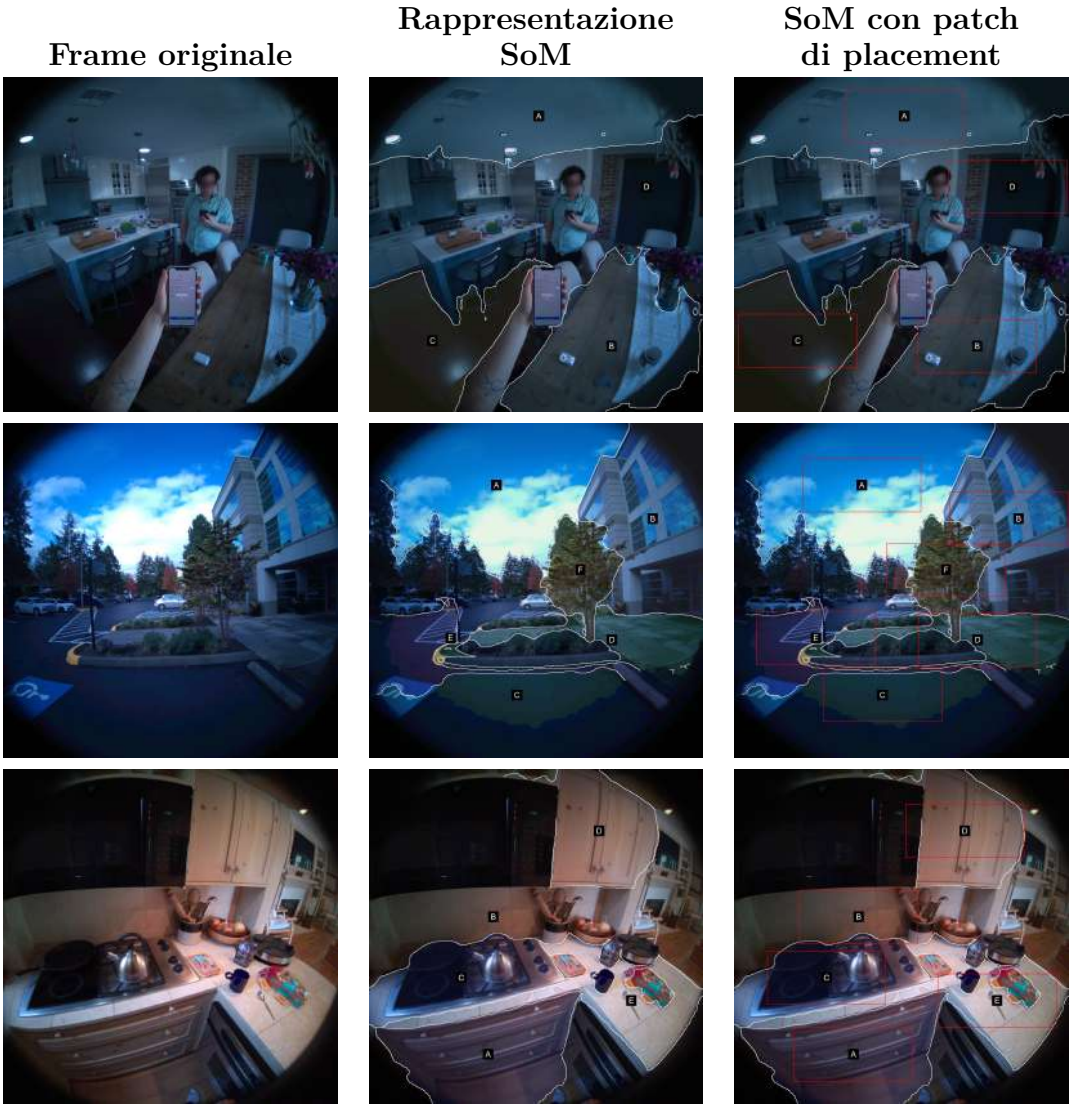


Figura 6.8: Visualizzazione del processo di derivazione delle patch di placement a partire dalle regioni semantiche (Mask2Former). Da sinistra a destra: frame originale, rappresentazione finale in stile Set-of-Mark (SoM) con regioni indicizzate, e la stessa rappresentazione con le patch rettangolari effettivamente utilizzate per il calcolo dello score (contorno rosso), centrate sull'identificatore numerico di ciascuna regione.

Capitolo 7

Valutazione sperimentale di Qwen2.5-VL

7.1 Setup sperimentale generale

Questa sezione descrive il setup sperimentale generale adottato per la valutazione del modello Qwen2.5-VL nei task di *visibility* e *placement*. In particolare, vengono illustrati il backbone vision–language utilizzato, le modalità di integrazione delle informazioni visive e testuali, il protocollo di training e la suddivisione del dataset. Tutti gli esperimenti presentati nelle sezioni successive condividono il medesimo setup di base, differendo unicamente nella formulazione del task, nella definizione dell'output supervisionato e nelle metriche di valutazione adottate.

7.1.1 Backbone vision–language

Per entrambi i task considerati utilizziamo **Qwen2.5-VL**, un modello vision–language *decoder-only* progettato per la generazione testuale condizionata da input multimodali, quali immagini e prompt testuali. Come spiegato in maniera più approfondita nella Sezione 2.3, il modello è strutturato in tre componenti principali:

1. un **vision encoder** basato su architettura Vision Transformer (ViT), che mappa ciascuna immagine in una sequenza di embedding visivi

2. un **modulo di proiezione multimodale**, che allinea gli embedding visivi allo spazio degli embedding linguistici

3. un **backbone linguistico decoder-only**, che genera token in modo autoregressivo da sinistra a destra

Nel setup sperimentale adottato, le immagini vengono quindi convertite in una sequenza di *token visivi* e inserite direttamente nel contesto del modello insieme ai token testuali, formando un'unica sequenza multimodale. L'interazione con il modello è strutturata secondo un *chat template* standard, articolato in tre turni principali: un turno di *system*, che definisce il comportamento globale e le politiche di risposta del modello; un turno di *user*, che rappresenta l'input fornito dall'utente, comprendente sia informazioni testuali sia contenuti visivi; e un turno di *assistant*, che corrisponde alla risposta generata dal modello in base al contesto conversazionale e multimodale fornito. Un esempio astratto della sequenza utilizzata è riportato di seguito:

```
[<|system|>
    istruzioni globali
<|end|>]

[<|user|>
    <|vision_start|> ... <|vision_end|>
    prompt testuale
<|end|>]

[<|assistant|>
    output target
<|end|>]
```

Supervisione selettiva tramite mascheramento della loss. Il modello Qwen2.5-VL opera su un'unica sequenza multimodale ottenuta concatenando token testuali e token visivi; durante l'addestramento autoregressivo, la funzione obiettivo standard consiste nel massimizzare la probabilità del token successivo dato il prefisso.

Nel contesto di questa tesi, tuttavia, non tutte le porzioni della sequenza devono contribuire al processo di ottimizzazione. L'obiettivo è infatti quello di supervisionare esclusivamente l'output generato dal turno di *assistant* (ovvero la *label* e, nel task di *visibility*, anche la relativa *reason*), evitando che il modello apprenda a ricostruire porzioni del prompt di input o i marcatori strutturali del template conversazionale.

Per implementare questa supervisione selettiva, abbiamo adottato la convenzione comunemente utilizzata nel fine-tuning di LLM e VLM: i token non supervisionati vengono esclusi dalla loss impostando il corrispondente valore in `labels` a `-100`. In PyTorch, tale valore viene ignorato dalla `CrossEntropyLoss`, consentendo di calcolare il gradiente esclusivamente sulle posizioni di interesse.

In particolare, nel nostro setup:

- tutti i token appartenenti ai turni *system* e *user* vengono mascherati
- vengono inclusi nella loss esclusivamente i token di contenuto del turno *assistant*
- i token di padding e i token speciali associati alle immagini vengono sempre esclusi, per evitare segnali di training non informativi

Implementazione tramite `collate_function` personalizzata. Dal punto di vista implementativo, la strategia proposta è realizzata mediante una `collate_function` personalizzata, ovvero una funzione utilizzata dal `DataLoader` per aggregare e pre-processare i campioni di un batch prima del loro inoltro al modello. All'interno di tale funzione, (i) viene applicato il *chat template* adottato da Qwen per strutturare la sequenza conversazionale; (ii) viene utilizzato il `processor` del modello, responsabile della tokenizzazione del testo e dell'allineamento tra input testuali e visivi, per generare gli `input_ids`, ossia la rappresentazione numerica della sequenza multimodale che viene effettivamente fornita in ingresso al modello; e (iii) viene costruito il tensore delle `labels`, inizializzato al valore `-100` (impiegato per escludere determinate posizioni dal calcolo della funzione di loss), che viene successivamente valorizzato esclusivamente nelle posizioni corrispondenti all'output del turno di *assistant*. Tale porzione della sequenza è individuata tramite i marcatori strutturali del template conversazionale, in particolare `<|im_start|>` e `<|im_end|>`.

Il frammento di codice presente nel Listing 7.1 riporta lo schema essenziale di mascheramento adottato (omettendo dettagli implementativi non cruciali, come controlli di robustezza e logging).

```
# Initialize labels with ignore index
labels = torch.full_like(input_ids, -100)

for example in batch:
    # Identify assistant response segment:
    # <|im_start|> assistant ... <|im_end|>
    assistant_start, assistant_end = find_assistant_segment(example)

    for pos in range(assistant_start, assistant_end):
        token_id = input_ids[pos]

        # Consider only valid assistant tokens
        if token_id != PAD_TOKEN_ID and token_id != IMAGE_TOKEN_ID:
            labels[pos] = token_id

batch["labels"] = labels
```

Listing 7.1: Mascheramento dei token non target durante il fine-tuning

In questo modo, il modello viene ottimizzato unicamente per generare la risposta dell'assistente (ad es. la sequenza *label* o *label + reason*), mantenendo invariati i token del prompt e i token di controllo del template. Tale scelta risulta particolarmente importante nei nostri esperimenti, poiché consente di controllare con precisione *quale parte* della sequenza contribuisce alla loss e di rendere comparabili le diverse configurazioni di training presentate nelle sezioni successive.

È importante sottolineare che questa strategia di mascheramento della loss non rappresenta una scelta ad hoc del presente lavoro, ma è coerente con le pratiche raccomandate per il fine-tuning di modelli vision–language di tipo *decoder-only*. In particolare, la guida ufficiale al fine-tuning di **Qwen2.5-VL** pubblicata da Roboflow [34] adotta esplicitamente la stessa impostazione: durante la preparazione dei batch di training, i token corrispondenti al messaggio di sistema, agli input visivi e al turno dell'utente vengono mascherati, in modo che la funzione di loss venga calcolata esclusivamente sulla risposta dell'assistente, che rappresenta il target supervisionato del modello.

Come riportato nella documentazione ufficiale del modello, la *collate function* utilizzata durante la fase di training si occupa di formattare correttamente gli input conversazionali testuali e visivi, applicare la tokenizzazione e il padding delle sequenze, e mascherare opportunamente nei `labels` i turni di *system*, i token associati alle immagini e l'input dell'utente, in modo tale che la funzione di loss venga calcolata esclusivamente sulla risposta generata dal turno di *assistant* [34]. Questa impostazione risulta pienamente coerente con le linee guida ufficiali del modello e garantisce che il processo di ottimizzazione si concentri unicamente sulla generazione dell'output desiderato, evitando di penalizzare il modello per la ricostruzione di informazioni già fornite nel contesto di input.

7.1.2 Suddivisione del dataset

Il dataset utilizzato negli esperimenti è composto da **235 video** acquisiti in scenari di Mixed Reality, includendo sia **ambienti indoor** sia **ambienti outdoor**. Questa varietà consente di coprire un ampio spettro di condizioni visive, strutturali e percettive, quali differenti configurazioni spaziali, livelli di complessità della scena e condizioni di illuminazione.

Come descritto nella Sezione 5.1, da ciascun video viene effettuata una fase di estrazione e pre-processing dei frame RGB, comprensiva di campionamento temporale, sincronizzazione con le misure di *eye-gaze* e normalizzazione geometrica. A valle di tale procedura, per ogni video vengono quindi **selezionati casualmente 20 frame**, che costituiscono le istanze effettivamente utilizzate per la costruzione del dataset supervisionato.

Al fine di garantire una valutazione corretta delle capacità di generalizzazione del modello ed evitare fenomeni di *data leakage*, la suddivisione del dataset è stata

effettuata in modo **video-wise** e non frame-wise. In questo modo, tutte le istanze derivate da uno stesso video appartengono a una sola partizione, assicurando che nessuna informazione temporale o contestuale venga condivisa tra training, validation e test set.

Il dataset è stato suddiviso secondo uno schema **70/20/10** nelle seguenti partizioni:

- **training set**: 3290 istanze
- **validation set**: 940 istanze
- **test set**: 470 istanze

La stessa suddivisione viene mantenuta invariata per entrambi i task considerati, *visibility* e *placement*, e per tutte le configurazioni sperimentali presentate nei capitoli successivi. Tuttavia, pur partendo dagli stessi video sorgente, la selezione dei 20 frame per ciascun video è stata effettuata in modo indipendente per i due task. In questo modo, per un medesimo video, i frame utilizzati nel dataset relativo al task di *visibility* risultano, con alta probabilità, differenti da quelli selezionati per il dataset di *placement*. Tale scelta consente di evitare che i due task condividano gli stessi istanti temporali all'interno di uno stesso video, riducendo ulteriormente possibili correlazioni indesiderate tra i dataset e favorendo una valutazione più robusta e indipendente delle prestazioni del modello nei due contesti applicativi.

7.1.3 Metriche e hardware

Le prestazioni del modello vengono valutate attraverso una combinazione di metriche calcolate durante la fase di addestramento e metriche basate sull'inferenza finale, al fine di analizzare sia il comportamento di apprendimento del modello sia la sua effettiva capacità decisionale nei task considerati.

Metriche durante il training. Durante il fine-tuning vengono monitorate la **mean train token accuracy** e la **mean eval token accuracy**. Tali metriche misurano l'accuratezza media a livello di token sull'insieme di training e di validation, considerando esclusivamente i token supervisionati (ovvero quelli appartenenti al turno dell'assistente, come descritto nella Sezione precedente).

Queste metriche forniscono un'indicazione generale della stabilità dell'addestramento, della convergenza del modello e dell'assenza di fenomeni evidenti di overfitting. In particolare, esse permettono di verificare che il modello apprenda correttamente la struttura dell'output richiesto (label e, quando previsto, *reason*) e che le prestazioni sul validation set seguano un andamento coerente con quelle sul training set.

Limiti delle metriche a livello di token. Nonostante la loro utilità in fase di training, le metriche a livello di token risultano solo parzialmente rappresentative delle prestazioni effettive del sistema. I task considerati in questo lavoro richiedono infatti una **decisione discreta finale**: una label binaria nel caso della *visibility* o la selezione di una posizione candidata nel caso del *placement*. Di conseguenza, una generazione testuale parzialmente corretta può comunque condurre a una decisione errata, rendendo necessaria una valutazione specifica a livello di output finale.

Metrica principale in fase di inferenza. Per questo motivo, la valutazione principale viene condotta in **fase di inferenza**, misurando l'accuratezza sulla **predizione discreta finale** restituita dal modello. In questa fase, il modello genera una risposta testuale completa in modo deterministico (senza sampling), al fine di garantire stabilità e riproducibilità delle decisioni.

Nel task di *visibility*, la predizione finale corrisponde a una **label numerica binaria** (0/1), e l'accuratezza viene calcolata confrontando direttamente la label predetta con quella di riferimento. Nel task di *placement*, invece, la predizione consiste in una **lettera dell'alfabeto** che identifica una delle posizioni candidate nella rappresentazione *Set-of-Mark*.

A differenza della visibility, il placement presenta una natura intrinsecamente **multi-soluzione**: in molte scene possono esistere più posizioni semanticamente ed ergonomicante plausibili per il posizionamento dell'overlay. Per questo motivo, la valutazione in inferenza per il task di placement non si limita solo ad un confronto esatto con la singola *ground truth*, ma viene anche implementata una **metrica di accuratezza Top-3**, che considera corretta una predizione se la lettera generata dal modello rientra tra le tre migliori posizioni di riferimento associate all'istanza.

In entrambi i task, dall'output testuale generato dal modello viene estratto il **primo simbolo valido** (cifra o lettera), che rappresenta la decisione del modello. La predizione viene considerata errata nel caso in cui non venga generato alcun simbolo valido o, nel task di placement, se la lettera estratta non rientra nell'insieme delle tre posizioni ammissibili. Questa impostazione consente di valutare in modo diretto e coerente la capacità del modello di risolvere il task decisionale richiesto, indipendentemente dalla correttezza o completezza della parte testuale accessoria eventualmente generata.

Hardware. Tutti gli esperimenti sono stati eseguiti utilizzando risorse di calcolo ad alte prestazioni messe a disposizione dal **Cambridge Service for Data Driven Discovery (CSD3)**, uno dei servizi HPC nazionali di livello EPSRC Tier-2, gestito dai Research Computing Services (RCS) dell'Università di Cambridge [35]. In particolare, le valutazioni sperimentali sono state condotte sul sotto-cluster GPU (*Wilkes3-GPU*), basato su acceleratori **NVIDIA A100**, affiancati da nodi CPU equipaggiati con processori **Intel Xeon Scalable** (Cascade Lake, Ice Lake e

Sapphire Rapids). Tutti i nodi del sistema sono interconnessi tramite rete **Mellanox HDR InfiniBand**, garantendo elevate prestazioni di comunicazione. I dettagli relativi alle risorse computazionali effettivamente utilizzate, alle configurazioni hardware e ai tempi di addestramento e inferenza vengono riportati insieme ai risultati sperimentali, al fine di garantire la riproducibilità degli esperimenti e una corretta interpretazione delle prestazioni ottenute.

7.2 Performance di Qwen2.5-VL sul task di visibility

In questa sezione vengono presentati e analizzati i risultati sperimentali ottenuti da **Qwen2.5-VL** sul task di *visibility*, con l'obiettivo di valutare l'efficacia del modello nel discriminare correttamente tra configurazioni di interfaccia accettabili e problematiche in scenari di Mixed Reality. L'analisi si concentra sia sulla **capacità decisionale** del modello, misurata in termini di accuratezza della label binaria predetta, sia sugli effetti introdotti dalle diverse strategie di supervisione adottate durante il fine-tuning.

In particolare, vengono confrontate le prestazioni di tre configurazioni sperimentali descritte nella Sezione successiva, al fine di analizzare l'impatto dell'integrazione di spiegazioni testuali, sia libere sia deterministiche, sul comportamento del modello. Tale confronto consente di valutare non solo la correttezza delle decisioni di visibility, ma anche la **stabilità**, la **coerenza** e il potenziale ruolo esplicativo delle *reason* generate, offrendo una visione completa delle capacità del modello in un contesto applicativo orientato alla Mixed Reality.

7.2.1 Setup sperimentale e modello utilizzato

Il task di *visibility* viene valutato adottando il setup sperimentale generale descritto nella Sezione 7.1. In questa sezione vengono introdotti esclusivamente gli aspetti **specifici della formulazione del task di visibility**, che lo distinguono dal task di placement in termini di output supervisionato e strategia di training.

Nel task di visibility, il modello è chiamato a determinare se la posizione di un elemento di interfaccia sovrapposto alla scena reale sia *accettabile* oppure *problematicamente invasiva*. L'output principale è quindi una **label binaria**, che assume valore 1 nel caso di *good placement* e valore 0 nel caso di *bad placement*. A seconda della configurazione sperimentale, tale decisione può essere accompagnata da una **motivazione testuale** (*reason*) espressa in linguaggio naturale.

Dal punto di vista del modello vision–language, il task è formulato come una **generazione autoregressiva di testo**. La label binaria è rappresentata dal **primo token numerico** generato dal turno *assistant*; quando prevista, la spiegazione

testuale segue immediatamente tale token. Durante il fine-tuning, la supervisione viene applicata in modo selettivo sui token rilevanti per il task, consentendo di controllare con precisione quali componenti dell'output contribuiscano alla funzione di loss.

Configurazioni sperimentali. Per analizzare l'impatto della spiegazione testuale sul comportamento del modello, sono state considerate tre configurazioni sperimentali progressive:

1. **Fine-tuning con supervisione sul solo token binario.** Il modello viene supervisionato esclusivamente sulla label binaria (0 o 1), rappresentata dal primo token dell'output dell'assistente.

2. **Integrazione preliminare di una spiegazione testuale libera.** In un secondo esperimento, il modello è incoraggiato a generare una spiegazione testuale senza una supervisione esplicita sul suo contenuto

3. **Integrazione della *reason* deterministica.** Nella configurazione finale, il modello viene addestrato a generare una coppia *label + reason*, in cui la spiegazione è fornita come target supervisionato e derivata in modo deterministico dalle mappe contestuali, garantendo coerenza e fedeltà al processo decisionale

Questa formulazione progressiva consente di valutare separatamente: (i) la capacità decisionale del modello sul task di visibility, (ii) la sua propensione spontanea a fornire spiegazioni, e (iii) l'effetto di una supervisione esplicita e deterministica sulla qualità e affidabilità delle *reason*. In tal modo, il task di visibility diventa anche un banco di prova per l'integrazione di meccanismi di spiegabilità nei modelli vision–language applicati a scenari di Mixed Reality.

7.2.2 Fine-tuning con supervisione sul solo token binario

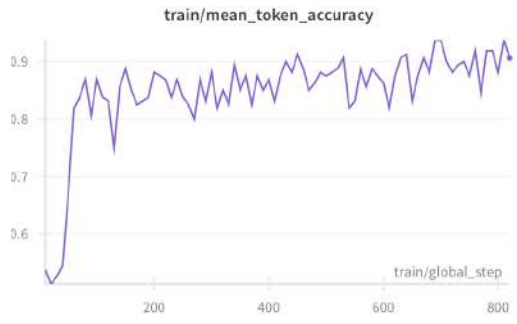
In una prima fase sperimentale, è stata esplorata la possibilità di sfruttare esclusivamente le **capacità di classificazione implicite** del modello, formulando il task di visibility come una **decisione binaria pura**. In questo setting, il prompt richiede al modello di stabilire se la posizione dell'overlay comprometta o meno la visibilità della scena; tuttavia, durante il fine-tuning, nel turno dell'*assistant* viene supervisionato **unicamente il token finale** corrispondente alla label binaria (*0/1*).

Gli esperimenti sono stati condotti utilizzando il modello **Qwen2.5-VL-3B**, fine-tunato per **4 epoche** con **learning rate pari a** 1×10^{-4}, **LoRA rank pari a** 32 e scheduler di tipo *constant*. Per quanto riguarda la componente visiva, è stata adottata una configurazione con **256 visual patches**, in modo da fornire

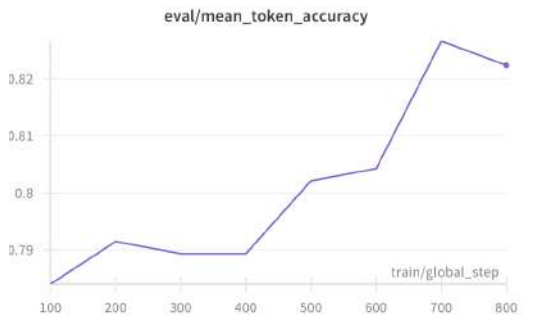
al modello una rappresentazione spaziale sufficientemente dettagliata della scena senza introdurre un costo computazionale eccessivo.

Di conseguenza, la funzione di loss viene calcolata **esclusivamente su tale token**, mentre tutte le altre posizioni della sequenza vengono mascherate e non contribuiscono all'ottimizzazione. Questa configurazione consente di valutare in modo isolato quanto efficacemente Qwen2.5-VL riesca ad apprendere il task di visibility come problema di classificazione binaria, senza alcuna richiesta esplicita di generazione di spiegazioni testuali.

I risultati ottenuti durante il fine-tuning risultano incoraggianti. In particolare, il modello raggiunge una *mean train token accuracy* pari a **0.9437** e una *mean eval token accuracy* pari a **0.8223**, come mostrato in Figura 7.1. Tali valori indicano che il modello apprende correttamente la corrispondenza tra input multimodale e label binaria, mostrando una buona capacità di generalizzazione sul validation set.



(a) Andamento della *mean train token accuracy* durante il fine-tuning con supervisione sul solo token binario.



(b) Andamento della *mean validation token accuracy* durante il fine-tuning con supervisione sul solo token binario.

Figura 7.1: Metriche a livello di token durante il fine-tuning del task di visibility con supervisione limitata alla label binaria. Il training converge rapidamente, mentre le prestazioni in validazione indicano una buona capacità di generalizzazione.

In fase di inferenza sul validation set, l'accuratezza calcolata confrontando esclusivamente la label binaria predetta dal modello con la label di riferimento raggiunge un valore pari a **0.8011** (753 predizioni corrette su 940 istanze), confermando la coerenza delle prestazioni osservate durante il training.

Il prompt testuale utilizzato per questa configurazione sperimentale, in cui viene supervisionato esclusivamente il token binario finale, è riportato integralmente in Appendice A.1.1.

Limiti emersi in fase di inferenza reale

Nonostante i risultati quantitativi positivi ottenuti durante il fine-tuning, l'utilizzo del modello addestrato in questo setting evidenzia una limitazione fondamentale quando viene impiegato in **fase di inferenza reale**. In particolare, è importante sottolineare che il modello è stato fine-tunato utilizzando un prompt che richiedeva **esclusivamente la restituzione della label binaria** (*0/1*), senza alcuna richiesta di spiegazione testuale (Appendice A.1.1). In fase di inferenza, invece, viene utilizzato un prompt più ricco, nel quale al modello è richiesto di fornire non solo la decisione binaria, ma anche una breve motivazione in linguaggio naturale.

In questo contesto, si osserva che, anche quando il prompt di inferenza richiede esplicitamente sia la decisione binaria sia una spiegazione testuale — come nel caso del prompt completo riportato in Appendice A.1.3 — il modello restituisce **esclusivamente il token binario finale** (*0* oppure *1*), ignorando completamente il formato di output richiesto. Un esempio rappresentativo di questo comportamento è mostrato in Figura 7.2.

Figura 7.2: Esempio di output in fase di inferenza nello setting con supervisione sul solo token binario. Nonostante il prompt richieda anche una motivazione testuale, il modello restituisce esclusivamente la label numerica (`0` o `1`).

Questo comportamento indica che il modello ha appreso che **l'unica parte rilevante dell'output è il token binario**, e non mostra alcuna tendenza spontanea a generare una spiegazione testuale aggiuntiva quando questa non è stata prevista durante la fase di addestramento. Tale effetto è una conseguenza diretta del meccanismo di training adottato: poiché la funzione di loss viene calcolata esclusivamente sul token binario, il modello non riceve alcun segnale di supervisione sulle restanti posizioni della sequenza e impara a considerarle irrilevanti dal punto di vista dell'ottimizzazione.

Questo esperimento mette in evidenza un aspetto cruciale dei modelli vision–language autoregressivi: la generazione di spiegazioni non può emergere in modo

affidabile in assenza di una supervisione esplicita. Il limite osservato non è quindi riconducibile a una carenza delle capacità generative del modello, bensì alla formulazione della funzione obiettivo, che incentiva unicamente la correttezza della decisione binaria.

Questa osservazione ha due implicazioni principali. In primo luogo, mostra che una formulazione basata esclusivamente sulla classificazione binaria (*good/bad placement*) è insufficiente per applicazioni reali di Mixed Reality, nelle quali il sistema deve rendere **esplicite e comprensibili le motivazioni delle proprie decisioni**. In secondo luogo, evidenzia la necessità di introdurre una **supervisione esplicita delle *reason*** qualora si desideri ottenere spiegazioni coerenti, stabili e allineate al processo decisionale sottostante.

7.2.3 Integrazione preliminare di una spiegazione testuale

Prima di introdurre un meccanismo di generazione deterministica delle *reason*, è stato esplorato un approccio intermedio volto a incoraggiare il modello a produrre una spiegazione testuale in modo *libero*, senza una supervisione esplicita sul contenuto semantico della motivazione. L'obiettivo di questo esperimento preliminare è valutare se, e in che misura, un modello vision–language autoregressivo possa apprendere spontaneamente a fornire una spiegazione testuale a partire da un segnale di supervisione parziale.

In particolare, la costruzione del dataset per il fine-tuning è stata modificata inserendo, nel turno dell'*assistant*, una stringa strutturata che esplicitasse l'avvio di una motivazione, anziché fornire esclusivamente il token binario. L'output supervisionato assume quindi la forma:

```
<label> The overlay is [shown/not shown] because    (7.1)
```

dove la scelta tra *shown* e *not shown* è coerente con la label binaria associata all'istanza.

Il prompt testuale utilizzato in questo setting, riportato integralmente in Appendice A.1.2, è stato impiegato in modo **coerente sia durante il fine-tuning sia in fase di inferenza**, evitando discrepanze tra il formato supervisionato e quello richiesto al modello in fase di valutazione.

Parallelamente, la *collate function* è stata aggiornata in modo da mascherare l'intero contesto del prompt, mantenendo visibili alla funzione di loss soltanto i token appartenenti al turno dell'*assistant*. In questo modo, il modello non viene più addestrato come semplice classificatore binario, ma è incentivato a generare una risposta testuale strutturata che includa esplicitamente una spiegazione.

Per valutare questo approccio è stato utilizzato il modello **Qwen2.5-VL-3B**, addestrato per **4 epoch** con *learning rate* pari a **1e–4**, **LoRA rank 32**, scheduler di tipo *constant* e un numero fisso di **256 visual patches**. In fase di inferenza, il

modello viene interrogato utilizzando lo stesso prompt strutturato adottato durante il fine-tuning (Appendice A.1.2), che richiede esplicitamente la generazione di una label binaria seguita da una spiegazione testuale conforme a vincoli sintattici predefiniti.

Analisi del comportamento del modello

I risultati di questo esperimento mostrano un comportamento interessante ma al tempo stesso problematico. Da un lato, il modello dimostra di aver appreso correttamente la **decisione binaria**: la label 0/1 viene predetta con una buona accuratezza, pari a circa **0.80** in fase di inferenza, in linea con quanto osservato negli esperimenti precedenti. Dall'altro lato, la componente testuale che segue la label risulta **fortemente degenerata**.

Come illustrato in Figura 7.3, il modello tende a produrre sequenze altamente ripetitive, reiterando più volte lo stesso pattern sintattico appreso durante il training. In particolare, la struttura introduttiva

```
<label> The overlay is [shown/not shown] because
```

viene ripetuta ciclicamente, senza che il modello riesca a sviluppare una spiegazione informativa coerente e ben formata.

Solo raramente, e tipicamente in coda alla sequenza generata, compare una frase più significativa, ad esempio *"because it covers the user's phone"*. Tuttavia, tale contenuto risulta spesso preceduto da una lunga serie di ripetizioni, rendendo l'output complessivo poco leggibile e difficilmente utilizzabile come spiegazione affidabile in un contesto applicativo.

Questo comportamento suggerisce che il modello abbia appreso correttamente il **pattern formale** della risposta — ossia la struttura sintattica del prefisso testuale — ma non abbia acquisito una nozione operativa di spiegazione.

In assenza di una supervisione esplicita sul contenuto semantico della *reason*, il modello tende a massimizzare la probabilità dei token osservati più frequentemente durante il training, riciclando indefinitamente il template introduttivo senza sapere né quando arrestare la generazione, né come arricchirla di contenuto informativo.

Nel complesso, questo esperimento evidenzia un limite strutturale dell'approccio: **fornire esclusivamente l'avvio di una frase esplicativa non è sufficiente** per indurre un modello vision–language autoregressivo a generare spiegazioni coerenti, diverse e ancorate al contenuto visivo. Il modello ottimizza correttamente la predizione della label binaria e la riproduzione del prefisso testuale, ma non riceve alcun incentivo a produrre una spiegazione semanticamente significativa.

Questi risultati rafforzano l'idea che, nei modelli vision–language autoregressivi, la spiegazione non possa emergere come sottoprodotto spontaneo della classificazione. Al contrario, essa deve essere **esplicitamente supervisionata**, sia nella

```
--------------
Image: /home/gp589/rds/hpc-work/adaptive-ui-clean/data/video_frames/loc3_script3_seq1_rec2/
frame_1270.png
Image: /home/gp589/rds/hpc-work/adaptive-ui-clean/data/generated_overlays/task_2/
loc3_script3_seq1_rec2/frame-1270.png
Output model (digit): 1 | Real label: 1
Full response: 1 The overlay is shown because 1 The overlay is shown because it is not shown
because 0 The overlay is not shown because 0 The overlay is not shown because 1 The overlay is
shown because 1 The overlay is shown because 0 The overlay is not shown because 0 The overlay
--------------
Image: /home/gp589/rds/hpc-work/adaptive-ui-clean/data/video_frames/loc3_script3_seq1_rec2/
frame_600.png
Image: /home/gp589/rds/hpc-work/adaptive-ui-clean/data/generated_overlays/task_2/
loc3_script3_seq1_rec2/frame-600.png
Output model (digit): 0 | Real label: 0
Full response: 0 The overlay is not shown because 0 The overlay is not shown because 1 The
overlay is shown because 1 The overlay is shown because it covers the user's phone.
--------------
Image: /home/gp589/rds/hpc-work/adaptive-ui-clean/data/video_frames/loc3_script3_seq1_rec2/
frame_510.png
Image: /home/gp589/rds/hpc-work/adaptive-ui-clean/data/generated_overlays/task_2/
loc3_script3_seq1_rec2/frame-510.png
Output model (digit): 0 | Real label: 0
Full response: 0 The overlay is not shown because 0 The overlay is not shown because 1 The
overlay is shown because 1 The overlay is shown because it covers the user's phone.
--------------
```

Figura 7.3: Esempi di output testuale generato in fase di inferenza nello setting con integrazione preliminare di una spiegazione testuale. Pur in presenza di una predizione corretta della label binaria, la *reason* degenera in sequenze altamente ripetitive del prefisso testuale, senza sviluppare una spiegazione semanticamente informativa.

forma sia nel contenuto, se si desidera ottenere output interpretabili, controllabili e coerenti con la logica decisionale del sistema.

7.2.4 Integrazione della *reason* deterministica

Alla luce dei limiti emersi negli esperimenti precedenti sulla generazione libera delle spiegazioni, abbiamo introdotto nel dataset una **reason deterministica**, calcolata automaticamente e derivata dagli stessi segnali computazionali utilizzati per la decisione di visibility (Sezione 6.2.3). In particolare, la motivazione testuale viene generata tramite la funzione `_compute_reason()`, la quale combina in modo strutturato le informazioni provenienti dalle mappe di *functionality*, *safety/social acceptability* e *aesthetics*, insieme ai risultati della *object detection*.

In questo setting, ogni istanza del dataset include nel turno dell'*assistant* la sequenza completa:

```
<label> <reason>
```
(7.2)

dove la *reason* è una spiegazione testuale compatta, riproducibile e **faithful**, poiché costruita a partire dagli stessi punteggi e oggetti che determinano la label binaria. Questo consente di fine-tunare il modello non solo sulla predizione della decisione

0/1, ma sull'intero pattern *"label + reason"*, insegnandogli a produrre spiegazioni semanticamente coerenti e allineate alla logica decisionale del dataset.

Il prompt utilizzato in questo setting, che impone vincoli stringenti sulla struttura dell'output, è riportato integralmente in Appendice A.1.3. Rispetto agli approcci precedenti, il modello riceve ora un segnale di supervisione esplicito sia sulla forma sia sul contenuto della *reason*, eliminando l'ambiguità osservata nella generazione libera.

Prima di procedere con il fine-tuning, abbiamo verificato manualmente la qualità delle *reason* generate automaticamente su circa **14 video**, riscontrando una buona coerenza tra contenuto visivo e spiegazione associata. Nel complesso, le motivazioni risultano interpretabili, stabili e coerenti con la semantica delle mappe contestuali sottostanti.

Limiti nella generazione automatica delle *reason*

I limiti ricorrenti emersi durante questa verifica preliminare non dipendono dal meccanismo deterministico in sé, ma dalla **granularità dei segnali** disponibili nella pipeline di annotazione automatica, in particolare dalla componente di *object detection* basata su YOLO-World.

Il primo limite riguarda la **descrizione delle persone**. YOLO-World restituisce bounding box e classi semantiche di alto livello (ad esempio *person*), ma non fornisce informazioni anatomiche o di posa. Di conseguenza, quando l'overlay occlude una persona, la *reason* può indicare correttamente la presenza di *person*, ma non può specificare quale parte del corpo sia effettivamente coinvolta (ad esempio mano, braccio o volto).

Un secondo limite riguarda il **riferimento all'utente**. Il sistema non distingue in modo esplicito tra l'utente e altre persone nella scena: la pipeline identifica la classe *person*, ma non dispone di un segnale dedicato per attribuire l'identità del soggetto principale. Questo può produrre spiegazioni corrette per la visibility, ma meno specifiche nella formulazione del riferimento.

È importante sottolineare che questi limiti non compromettono la correttezza della decisione di visibility: influenzano esclusivamente il **livello di dettaglio** della spiegazione. La *reason* rimane allineata ai fattori che hanno determinato la label, pur risultando talvolta meno precisa su dettagli anatomici o relazionali.

Coerenza qualitativa tra immagini e *reason* deterministica

In Figura 7.4 sono riportati alcuni esempi qualitativi rappresentativi che illustrano la coerenza tra il contenuto visivo delle scene e la *reason* deterministica calcolata tramite `_compute_reason()`.

Nei primi esempi (Figure 7.4a–7.4b), la *reason* evidenzia correttamente l'occlusione di regioni legate alla *safety/social acceptability* (ad esempio pavimento o

presenza di persone) oppure di fattori funzionali ed estetici. Negli esempi con label positiva (Figure 7.4c–7.4d), la spiegazione indica esplicitamente che l'overlay non interferisce con regioni critiche, risultando coerente con la scena e con i punteggi delle mappe contestuali.

Nel complesso, questi esempi mostrano come la *reason* deterministica rifletta in modo diretto e interpretabile i fattori che hanno determinato la label, superando i problemi di degenerazione e ripetizione osservati negli approcci precedenti basati su generazione libera.

Fine-tuning con *label + reason*

Sulla base delle evidenze raccolte nelle sezioni precedenti, la sola supervisione della label binaria si è rivelata insufficiente per ottenere output spiegabili in inferenza reale, mentre l'introduzione di un semplice prefisso testuale ha portato a generazioni degenerative e ripetitive. Per superare tali limiti, abbiamo adottato una formulazione supervisionata completa in cui il turno dell'*assistant* contiene esplicitamente l'intera sequenza *"label + reason"*. In questo modo, la funzione obiettivo incentiva congiuntamente (i) la correttezza della decisione binaria e (ii) la generazione di una motivazione coerente con i segnali contestuali che hanno determinato la label.

In particolare, abbiamo fine-tunato **Qwen2.5-VL-3B** utilizzando la nuova formulazione (*label + reason*) con gli stessi iperparametri adottati negli esperimenti precedenti: **4 epoche**, **learning rate** pari a **1e–4**, **LoRA rank 32**, scheduler *constant* e un numero fisso di **256 visual patches**. Come nei setting precedenti, la *loss* viene calcolata sull'intero contenuto del turno dell'*assistant*, includendo sia il token della label sia i token della *reason*.

Risultati quantitativi

I risultati ottenuti durante il fine-tuning mostrano una convergenza stabile del modello e un'elevata accuratezza a livello di token. In particolare, la *mean token accuracy* calcolata sul turno dell'*assistant* raggiunge un valore pari a **0.9735** sul training set e **0.9670** sul validation set. Tali risultati indicano che il modello apprende in modo affidabile la generazione dell'intera sequenza supervisionata, comprendente sia la decisione binaria sia la motivazione testuale associata.

In fase di inferenza, la valutazione quantitativa viene condotta considerando esclusivamente la correttezza della **label binaria** predetta, estratta come primo token numerico della risposta generata, in modo coerente con l'obiettivo primario del task di *visibility*. In questo setting, il modello raggiunge un'accuratezza pari a **0.8064** sul validation set e a **0.8688** sul test set.

Il miglioramento osservato sul test set suggerisce che la supervisione congiunta di *label* e *reason* favorisca l'apprendimento di una rappresentazione più robusta

del task, consentendo al modello di generalizzare efficacemente a scene non viste. In particolare, la presenza di una spiegazione testuale supervisionata sembra contribuire a stabilizzare il processo decisionale, incoraggiando il modello a collegare la predizione binaria al contenuto visivo sottostante piuttosto che a fare affidamento su correlazioni superficiali del dataset.

Analisi qualitativa in inferenza

Dal punto di vista qualitativo, l'aspetto più rilevante è che, grazie alla supervisione esplicita della *reason*, il modello produce risposte che risultano (i) coerenti con la label predetta e (ii) semanticamente allineate al contenuto dell'immagine, senza le degenerazioni osservate negli esperimenti con spiegazioni libere. In inferenza reale, il modello genera tipicamente risposte del tipo:

- *"0 The overlay covers floor and person, which are relevant for user safety and social acceptability."*
- *"1 The overlay does not occlude any important functional, safety, or aesthetic region."*

indicando che ha appreso il mapping completo *immagine + prompt* $\rightarrow$ *label + reason* in un singolo passaggio autoregressivo, con una struttura stabile e priva di ripetizioni.

In Figura 7.5 sono riportati alcuni esempi qualitativi di inferenza che mostrano la coerenza tra contenuto visivo e *reason* generata dal modello.

Discussione e prospettive

L'integrazione di una *reason* deterministica nel turno dell'*assistant* si è dimostrata un approccio **robusto e stabile**, permettendo di ottenere spiegazioni non ripetitive, semanticamente coerenti e direttamente ancorate alla logica decisionale del sistema. Rispetto agli approcci precedenti, la supervisione esplicita della sequenza *label + reason* consente al modello di associare la decisione binaria ai fattori contestuali che la giustificano, producendo output più interpretabili in inferenza reale.

I limiti osservati in questo setting non sono riconducibili al paradigma *label + reason*, ma alla ricchezza semantica dei segnali utilizzati per costruire automaticamente le motivazioni. In particolare, modelli di *object detection* come YOLO-World consentono di rilevare la presenza di persone, ma non di distinguere in modo affidabile quale parte del corpo sia coinvolta nell'occlusione.

Un'estensione naturale del sistema consisterebbe nell'integrare un modulo di **pose estimation** (ad esempio YOLO-Pose [36]) a valle della rilevazione della classe *person*, così da arricchire la descrizione delle regioni occluse e produrre spiegazioni più specifiche (ad esempio distinguendo tra mani, braccia o volto). Tale estensione

non modifica la logica del framework, ma migliorerebbe il livello di dettaglio delle *reason* nei casi in cui l'occlusione coinvolga persone.

7.2.5 Ablation studies

Dopo aver dimostrato l'efficacia della supervisione deterministica della *reason* e del fine-tuning sul pattern completo *label + reason*, conduciamo una serie di *ablation studies* per analizzare l'impatto di due fattori chiave sull'apprendimento e sulle prestazioni del modello nel task di *visibility*: (i) il numero di *visual patches* utilizzate dal vision encoder e (ii) il *LoRA rank*, che controlla la capacità parametrica degli adapter impiegati durante il fine-tuning.

L'obiettivo di queste analisi è duplice: da un lato, comprendere in che misura la granularità della rappresentazione visiva influenzi la capacità del modello di individuare correttamente le regioni occluse dall'overlay; dall'altro, valutare il trade-off tra accuratezza predittiva ed efficienza computazionale, aspetto cruciale per una futura integrazione del sistema in scenari di Mixed Reality orientati al real-time.

Impatto del numero di visual patches

Nel contesto di Qwen2.5-VL, l'immagine in input viene suddivisa in un insieme di *visual patches*, ciascuna delle quali genera un token visivo che viene proiettato nello spazio linguistico e concatenato alla sequenza testuale. Il numero di patches determina quindi la **risoluzione spaziale della rappresentazione visiva** disponibile al modello e influisce direttamente sulla quantità e sulla granularità dell'informazione utilizzabile per il ragionamento multimodale.

Per analizzare l'impatto di questo fattore, abbiamo finetunato **Qwen2.5-VL-3B** mantenendo invariati tutti gli iperparametri principali (4 epoche, learning rate 1×10^{-4}, LoRA rank 32, scheduler *constant*), variando esclusivamente il numero di visual patches. Le prestazioni sono state valutate sia tramite la *mean token accuracy* durante il training, sia tramite l'**accuratezza sulla label binaria** in fase di inferenza su validation e test set.

I risultati quantitativi sono riportati in Tabella 7.1.

Durante il fine-tuning, la *mean token accuracy* rimane elevata per tutte le configurazioni considerate, con valori compresi tra circa 0.93 e 0.97. Questo comportamento è in parte atteso, poiché la metrica viene calcolata sull'intera sequenza generata (*label + reason*), che include un numero significativo di token linguistici. Di conseguenza, anche in presenza di una rappresentazione visiva fortemente compressa, il modello riesce a predire correttamente una larga porzione dei token testuali, rendendo questa metrica **poco sensibile alla qualità e alla granularità dell'informazione visiva**.

Patches	Samples/s	Train tok. acc.	Eval tok. acc.	Test acc.	Val. acc.
4	**4.596**	0.9382	0.9326	0.4938	0.5213
32	4.195	0.9630	0.9591	0.8854	0.8787
64	4.336	0.9669	0.9635	0.9104	0.9096
128	4.021	0.9644	0.9659	0.9137	0.9118
256	3.629	**0.9700**	0.9678	0.9146	0.9128
512	3.140	0.9670	**0.9701**	**0.9167**	**0.9234**

Tabella 7.1: Impatto del numero di visual patches sulle prestazioni del task di visibility. La *token accuracy* è calcolata sull'intera sequenza generata (*label + reason*), mentre l'accuratezza di inferenza considera esclusivamente la correttezza della label binaria predetta.

La differenza emerge invece in modo netto in fase di inferenza, quando la valutazione è condotta esclusivamente sulla **correttezza della label binaria**. Come riportato in Tabella 7.1, si osserva un trend chiaro e monotono: all'aumentare del numero di visual patches, l'accuratezza di classificazione migliora in maniera significativa. Con un numero estremamente ridotto di patches (4), il modello opera su una rappresentazione visiva eccessivamente compressa, che non consente di cogliere dettagli locali rilevanti per il task di visibility, come mani, piccoli oggetti funzionali o porzioni specifiche della scena parzialmente coperte dall'overlay. In questo regime, l'accuratezza sul test set si mantiene inferiore a 0.50.

All'aumentare della risoluzione visiva ($32 \rightarrow 64 \rightarrow 128$ patches), il vision encoder preserva una struttura spaziale progressivamente più fine dell'immagine, consentendo al modello di individuare con maggiore affidabilità le regioni effettivamente occluse. Questo si riflette in un incremento consistente dell'accuratezza, che supera 0.88 già a 32 patches e raggiunge valori prossimi a 0.91 a partire da 64 patches. Oltre tale soglia, i miglioramenti diventano più contenuti: passando da 128 a 256 e infine a 512 patches, l'accuratezza cresce solo marginalmente, indicando l'ingresso in un regime di **rendimenti decrescenti** dal punto di vista decisionale.

Dal punto di vista computazionale, l'aumento del numero di visual patches comporta una riduzione progressiva del throughput di inferenza. Come mostrato in Tabella 7.1, il throughput passa da circa **4.5 samples/s** con 4 patches a circa **3.1 samples/s** con 512 patches, riflettendo l'aumento del numero di token visivi elaborati e il conseguente costo del meccanismo di attenzione multimodale. Tuttavia, tale riduzione risulta graduale e contenuta, e non introduce discontinuità significative nelle prestazioni computazionali.

Nel complesso, questa analisi conferma che, nel task di *visibility*, la **granularità della rappresentazione visiva** rappresenta un fattore cruciale fino a una

risoluzione intermedia, oltre la quale i benefici in termini di accuratezza diventano limitati. Allo stesso tempo, il costo computazionale cresce in modo regolare con il numero di patches, suggerendo che configurazioni con **64–128 visual patches** offrano un buon compromesso tra accuratezza decisionale e throughput di inferenza, risultando particolarmente adatte a un utilizzo *near real-time* in scenari di Mixed Reality.

Impatto del LoRA rank su accuratezza e performance

Oltre alla risoluzione visiva, abbiamo analizzato l'impatto della **capacità parametrica degli adapter LoRA** sul task di visibility, variando il *LoRA rank*. L'obiettivo di questa ablation study è valutare come l'aumento della capacità di adattamento del modello influenzi sia l'accuratezza predittiva, sia l'efficienza computazionale in inferenza, misurata in termini di *samples per second*. Quest'ultimo aspetto è particolarmente rilevante in scenari di Mixed Reality, dove il sistema è progettato per operare in contesti potenzialmente real-time.

Tutti gli esperimenti sono stati condotti mantenendo invariata la configurazione di base (**Qwen2.5-VL-3B**, 4 epoche, learning rate 1×10^{-4}, scheduler *constant*), variando esclusivamente il LoRA rank (4, 8, 16, 32) e considerando tre configurazioni rappresentative di risoluzione visiva: 4, 128 e 512 patches.

I risultati quantitativi sono riportati in Tabella 7.2, che mostra congiuntamente l'accuratezza di inferenza su validation e test set e il throughput in termini di *samples per second*.

Rank	Patches	Samples/s	Test Accuracy	Val. Accuracy
4	4	4.484	0.5000	0.5149
4	128	4.022	0.9104	0.9096
4	512	3.018	0.9042	0.9085
8	4	4.525	0.5104	0.5351
8	128	4.001	**0.9167**	0.9149
8	512	3.123	0.9062	0.9053
16	4	4.498	0.5062	0.5298
16	128	4.088	0.9021	0.9074
16	512	2.927	0.9062	0.9096
32	4	**4.596**	0.4938	0.5213
32	128	4.021	0.9137	0.9118
32	512	3.140	**0.9167**	**0.9234**

Tabella 7.2: Impatto congiunto del LoRA rank e del numero di visual patches sulle prestazioni in inferenza del task di visibility.

L'analisi dei risultati riportati in Tabella 7.2 evidenzia in modo chiaro come, nel task di *visibility*, l'impatto del **LoRA rank** sull'accuratezza decisionale sia fortemente subordinato alla qualità della rappresentazione visiva fornita al modello.

Nel regime a bassa risoluzione visiva (**4 patches**), l'accuratezza rimane prossima al caso casuale per tutti i valori di LoRA rank considerati, con valori di test accuracy compresi tra **0.49** e **0.51**. Questo comportamento indica che, in assenza di un segnale visivo sufficientemente informativo, l'aumento della capacità parametrica degli adapter LoRA non è in grado di migliorare in modo significativo le prestazioni. In tale configurazione, la granularità dell'informazione visiva rappresenta quindi il principale fattore limitante per il task di visibility.

Passando a una risoluzione intermedia (**128 patches**), si osserva un netto salto prestazionale: l'accuratezza sul test set supera stabilmente **0.90** per tutti i valori di LoRA rank, indicando che il modello dispone di un livello di dettaglio visivo adeguato per valutare correttamente l'impatto dell'overlay sulla scena. In questo regime, l'effetto del LoRA rank risulta contenuto e non monotono. In particolare, le differenze di accuratezza tra configurazioni con rank diversi sono limitate, suggerendo che, una volta raggiunta una soglia sufficiente di informazione visiva, la capacità degli adapter LoRA non costituisce il collo di bottiglia principale. La configurazione con **LoRA rank** 8 **e** 128 **patches** emerge come un punto di equilibrio favorevole, combinando un'elevata accuratezza (**0.9167** sul test set) con un throughput elevato (circa **4.0 samples/s**).

L'utilizzo di una risoluzione visiva più elevata (**512 patches**) consente di ottenere prestazioni comparabili o lievemente superiori in termini di accuratezza, con valori massimi di test accuracy pari a **0.9167**. Tuttavia, tali benefici risultano marginali rispetto al regime a 128 patches e sono accompagnati da una riduzione più marcata del throughput di inferenza, che scende a valori compresi tra **2.9** e **3.1 samples/s**. Questo andamento indica l'ingresso in un regime di **rendimenti decrescenti**, in cui l'aumento della risoluzione visiva non si traduce in miglioramenti proporzionali delle prestazioni decisionali.

Dal punto di vista computazionale, i risultati mostrano che il throughput di inferenza è influenzato principalmente dal numero di visual patches, mentre il LoRA rank ha un impatto secondario e limitato. A parità di risoluzione visiva, le variazioni di *samples/s* al variare del rank sono contenute e non compromettono significativamente l'efficienza del sistema.

Nel complesso, questa ablation study conferma che, nel task di *visibility*, la **granularità della rappresentazione visiva** rappresenta il fattore determinante per ottenere un'elevata accuratezza, mentre l'incremento del **LoRA rank** oltre valori moderati produce benefici limitati. In un'ottica applicativa di Mixed Reality, configurazioni con **128 visual patches** e **LoRA rank intermedio (8–16)** offrono il miglior compromesso tra accuratezza decisionale e prestazioni computazionali, risultando particolarmente adatte a un utilizzo *near real-time*.

Discussione finale

Le ablation studies condotte sul task di *visibility* evidenziano in modo consistente che la qualità del ragionamento del modello dipende in misura determinante dalla **granularità dell'informazione visiva**, mentre la capacità parametrica introdotta dagli adapter LoRA gioca un ruolo secondario una volta superata una soglia minima di risoluzione spaziale. Anche in presenza di una supervisione testuale ricca e strutturata (*label + reason*), una rappresentazione visiva eccessivamente compressa limita severamente la capacità del modello di individuare correttamente le regioni della scena occluse dall'overlay, portando a prestazioni prossime al caso casuale.

L'analisi dell'impatto del numero di *visual patches* mostra un chiaro trade-off tra accuratezza decisionale ed efficienza computazionale. A basse risoluzioni, l'informazione visiva risulta insufficiente per supportare una decisione affidabile, mentre l'incremento fino a configurazioni intermedie (64–128 patches) consente al modello di raggiungere livelli di accuratezza elevati e stabili. Oltre tale soglia, i benefici in termini di accuratezza diventano marginali, indicando l'ingresso in un regime di **rendimenti decrescenti**, a fronte di un costo computazionale crescente dovuto all'aumento del numero di token visivi elaborati.

L'ablation sul **LoRA rank** rafforza ulteriormente questa interpretazione: a parità di risoluzione visiva, l'aumento della capacità parametrica degli adapter non produce miglioramenti sistematici né monotoni delle prestazioni. In particolare, una volta raggiunto un livello sufficiente di dettaglio spaziale, l'accuratezza risulta sostanzialmente indipendente dal valore del rank, suggerendo che il collo di bottiglia del task di visibility sia di natura informativa piuttosto che parametrica.

Dal punto di vista computazionale, i risultati mostrano che il throughput di inferenza è guidato principalmente dal numero di visual patches, mentre il LoRA rank ha un impatto marginale. Le configurazioni più efficaci dal punto di vista decisionale, come quelle con **128 visual patches**, presentano un throughput di circa **4 samples/s**, corrispondente a una latenza di circa **250 ms per singolo frame**. Anche nelle configurazioni a risoluzione più elevata (512 patches), la latenza rimane dell'ordine di **300 ms per frame**. Tali valori risultano compatibili con un utilizzo *near real-time* nel contesto applicativo considerato, soprattutto tenendo conto che il task di visibility non è necessariamente eseguito a frequenza per-frame, ma verrà attivato in modo condizionale in risposta a cambiamenti rilevanti della scena o dell'overlay.

Nel complesso, queste analisi indicano che, per il task di *visibility*, una combinazione di **risoluzione visiva intermedia** (64–128 patches) e **LoRA rank moderato** (8–16) rappresenta il miglior compromesso tra accuratezza, interpretabilità e costi computazionali. In tale configurazione, il modello è in grado di produrre decisioni affidabili e spiegabili, mantenendo al contempo prestazioni compatibili

con i vincoli temporali tipici di sistemi di Mixed Reality *head-mounted*.

7.3 Performance di Qwen2.5-VL sul task di placement

Questa sezione riporta i risultati sperimentali di Qwen2.5-VL sul task di *placement*. A differenza del task di *visibility*, in cui l'obiettivo è formulare una decisione binaria (*good/bad placement*), il placement richiede al modello di **selezionare una posizione candidata** in cui collocare un overlay, minimizzando l'interferenza con il contenuto della scena.

Dal punto di vista applicativo, il placement rappresenta un problema più complesso: il modello non deve semplicemente evitare regioni critiche, ma individuare una soluzione spaziale plausibile tra più alternative potenzialmente valide. La formulazione del task rimane autoregressiva, ma cambia la natura dell'output, che consiste ora in una **lettera dell'alfabeto** associata a una delle regioni candidate mostrate nella rappresentazione *Set-of-Mark (SoM)*.

7.3.1 Setup specifico del task di placement

Nel task di placement, come descritto in maniera approfondita nella Sezione 6.3, l'immagine di input include la rappresentazione *Set-of-Mark (SoM)* costruita a partire dalle regioni semantiche individuate tramite Mask2Former. Ciascuna regione candidata è contrassegnata da un **identificatore alfabetico** (A, B, C, ...), che il modello è chiamato a selezionare come risposta finale.

Per ogni istanza, la *ground truth* viene determinata in modo deterministico selezionando la regione la cui *patch di placement* minimizza lo **score di interferenza** calcolato sulla *combined map*, come descritto nella Sezione 6.3.7. In questo modo, la supervisione rimane direttamente ancorata agli stessi fattori contestuali utilizzati nel task di visibility (funzionalità, sicurezza, accettabilità sociale, estetica e profondità), garantendo coerenza metodologica tra i due task.

Dal punto di vista del modello vision–language, il task è formulato come una **generazione autoregressiva di testo**, in cui l'output supervisionato dell'istanza assume la forma:

```
<lettera>
```

dove la *lettera* identifica la posizione candidata associata al costo di interferenza minimo.

Il prompt testuale utilizzato per il task di placement, che istruisce esplicitamente il modello a selezionare una delle posizioni candidate mostrate nella rappresentazione SoM e a restituire esclusivamente l'identificatore alfabetico corrispondente, è riportato integralmente in Appendice A.2.

7.3.2 Metrica di valutazione: accuratezza Top-3

Il task di *placement* presenta una natura intrinsecamente **multi-soluzione**. In numerosi scenari di Mixed Reality, infatti, possono esistere più posizioni che risultano tutte plausibili e corrette dal punto di vista percettivo, funzionale e contestuale. A titolo esemplificativo, una singola scena può includere:

- più superfici sufficientemente ampie e prive di occlusioni, che offrono aree di posizionamento equivalenti
- differenti regioni di sfondo visivamente uniformi, sulle quali l'overlay può essere collocato senza introdurre interferenze percettive rilevanti
- superfici distinte ma comparabili in termini di distanza da oggetti funzionali o persone, risultando quindi ugualmente accettabili per il posizionamento dell'elemento UI

In questi casi, penalizzare il modello per non aver selezionato *l'unica* posizione ottimale risulterebbe eccessivamente restrittivo e poco rappresentativo dello scenario applicativo. Per questo motivo, la valutazione viene condotta adottando, tra le metriche considerate, anche la **Top-3 accuracy** come misura principale per il task di *placement*.

Come illustrato nella Figura 7.6, vengono riportati diversi esempi che evidenziano come, all'interno di un singolo frame, possano coesistere più posizioni candidate valide per il task di *placement*, a conferma della natura multi-soluzione del problema.

In particolare, una predizione viene considerata corretta se la lettera generata dal modello coincide con una delle **tre migliori posizioni candidate**, determinate in fase di costruzione del dataset e ordinate in base a uno *score di interferenza* crescente. Tali posizioni corrispondono a:

- la posizione ottimale, assunta come *ground truth*
- la seconda migliore alternativa
- la terza migliore alternativa

Il fine-tuning del modello rimane supervisionato sulla **Top-1** (posizione ottimale), con l'obiettivo di apprendere la scelta considerata migliore secondo la *combined map*. In fase di inferenza, oltre alla valutazione basata sulla **Top-1 accuracy**, viene adottata anche la **Top-3 accuracy**, utilizzata per misurare la capacità del modello di produrre soluzioni percettivamente e funzionalmente accettabili anche nei casi in cui la predizione non coincida esattamente con la posizione ottimale.

7.3.3 Protocollo di inferenza e parsing dell'output

In fase di inferenza, il modello genera una risposta testuale in modo **deterministico** (`do_sample=False`). Questa scelta è deliberata e motivata dalla natura del task di placement, che richiede una **decisione stabile e riproducibile** a partire dallo stesso input visivo e testuale. L'utilizzo di strategie di sampling introdurrebbe una variabilità stocastica nell'output, rendendo difficile una valutazione consistente delle prestazioni e poco desiderabile in un contesto applicativo di Mixed Reality, in cui a uno stesso stato percettivo dovrebbe corrispondere una decisione coerente sul posizionamento dell'overlay.

Dall'output testuale generato viene quindi estratta la **prima lettera valida** (A–Z), che rappresenta la decisione finale del modello. La predizione viene considerata errata se:

- il modello non genera alcuna lettera valida;
- la lettera generata non rientra nell'insieme delle tre migliori posizioni di riferimento (accuratezza Top-1 e Top-3).

Questo protocollo di inferenza consente di valutare il modello in condizioni controllate e riproducibili, allineando il processo di valutazione sia ai requisiti sperimentali sia agli obiettivi applicativi del task di placement in scenari di Mixed Reality.

7.3.4 Accuratezza decisionale e requisiti di efficienza

Nel task di placement, il modello è chiamato a effettuare una **scelta discreta tra più posizioni candidate**, identificate da lettere nella rappresentazione *Set-of-Mark*. A differenza del task di *visibility*, tale decisione non è binaria, ma avviene in uno spazio di soluzioni potenzialmente ampio e semanticamente strutturato. Per questo motivo, le metriche a livello di token monitorate durante il fine-tuning non sono sufficienti a descrivere in modo affidabile la prestazione reale del modello sul compito decisionale.

Nel caso del placement, tali metriche vengono quindi utilizzate esclusivamente come **indicatori di stabilità dell'addestramento** e non vengono riportate nelle tabelle dei risultati. La valutazione quantitativa è condotta **unicamente in fase di inferenza**, adottando come metriche principali la **Top-1 accuracy** e la **Top-3 accuracy**, come descritto nella Sezione 7.3.2. Questa scelta riflette la natura intrinsecamente *multi-soluzione* del problema, in cui più posizioni possono risultare ugualmente accettabili dal punto di vista percettivo e funzionale.

Oltre all'accuratezza decisionale, nel contesto della Mixed Reality risulta fondamentale considerare anche le **prestazioni computazionali** del modello. Come verrà descritto nelle sezioni successive, il task di *placement* si basa infatti su una

pipeline più articolata rispetto al task di *visibility*, in quanto richiede una fase preliminare di segmentazione dell'immagine per l'individuazione delle superfici candidate. Tale complessità introduce un carico computazionale aggiuntivo che rende necessario analizzare non solo la qualità delle decisioni prodotte, ma anche i costi temporali associati al processo di inferenza.

Per questo motivo, oltre alle metriche di accuratezza Top-1 e Top-3, viene considerata anche la **velocità di inferenza**, espressa in termini di **samples per second (samples/s)**. Questa metrica consente di valutare il throughput effettivo del sistema nel suo complesso e di studiare il **compromesso tra accuratezza e performance**, un aspetto centrale nei sistemi di Mixed Reality, in cui una decisione leggermente sub-ottimale ma tempestiva può risultare preferibile a una scelta ottimale ma computazionalmente onerosa, specialmente in scenari *head-mounted* con vincoli di latenza stringenti.

Le sezioni successive riportano e discutono i risultati sperimentali tenendo conto congiuntamente di questi due aspetti: (i) la capacità del modello di selezionare posizioni plausibili e non intrusive e (ii) la sua idoneità a operare in contesti *near real-time*, in linea con i requisiti applicativi del task di placement.

7.3.5 Ablation Studies

In questa sezione conduciamo una serie di *ablation studies* volte ad analizzare l'impatto di alcuni **fattori architetturali e di training** sulle prestazioni di Qwen2.5-VL nel task di *placement*.

In particolare, in modo analogo a quanto fatto per il task di visibility, analizziamo l'effetto di:

- il **numero di visual patches** utilizzate dal vision encoder, che controlla la granularità della rappresentazione visiva fornita al modello
- il **LoRA rank**, che determina la capacità parametrica degli adapter impiegati durante il fine-tuning

Nel contesto del *placement*, queste analisi assumono un ruolo particolarmente rilevante. Dato che in questo caso, a differenza del task *visbility*, il modello non è chiamato ad emettere una decisione binaria, ma a **selezionare una posizione ottimale tra più candidati**, la qualità della rappresentazione visiva e la capacità espressiva degli adapter influenzano direttamente sia l'accuratezza della selezione sia la stabilità delle predizioni.

L'obiettivo di queste ablation studies è quindi duplice. Da un lato, valutare come la risoluzione visiva e la capacità del modello incidano in inferenza sia sulla **Top-1 accuracy** sia sulla **Top-3 accuracy**; dall'altro, analizzare il **trade-off tra accuratezza e throughput** (samples/s), aspetto cruciale per scenari di Mixed

Reality orientati al *near real-time*, in cui il placement dell'overlay deve avvenire in modo rapido, stabile e affidabile.

Le sezioni seguenti discutono separatamente l'impatto del numero di visual patches e del LoRA rank, mantenendo costante il resto del setup sperimentale.

Impatto del numero di visual patches

Nel modello **Qwen2.5-VL**, l'immagine di input viene suddivisa in un insieme di **patch visive**, ciascuna delle quali viene codificata dal vision encoder in un **token visivo** successivamente proiettato nello spazio linguistico e concatenato alla sequenza testuale. Il numero di *visual patches* determina quindi in modo diretto la **risoluzione spaziale della rappresentazione visiva** fornita al modello, influenzando sia la quantità di informazione disponibile sia il livello di dettaglio con cui la scena viene analizzata durante il processo decisionale.

Nel contesto del task di *placement*, questo fattore assume un ruolo particolarmente rilevante. A differenza della visibility, che richiede una decisione binaria globale, il placement implica la **discriminazione tra più posizioni candidate spesso spazialmente vicine**, caratterizzate da differenze sottili nei livelli di interferenza con la scena. Di conseguenza, la granularità della rappresentazione visiva diventa un elemento chiave per consentire al modello di confrontare correttamente patch di placement alternative e selezionare quella a costo minimo.

Per analizzare l'impatto del numero di visual patches su accuratezza ed efficienza, abbiamo condotto un'ablation study finetunando **Qwen2.5-VL-3B** secondo lo stesso schema sperimentale adottato per il task di *visibility*. In particolare, sono stati mantenuti invariati tutti gli iperparametri principali, numero di epoche (4), learning rate (1×10^{-4}), LoRA rank (32) e scheduler (*constant*), variando **esclusivamente il numero di visual patches** utilizzate dal vision encoder.

Durante il fine-tuning, le metriche a livello di token sono state monitorate unicamente come **indicatori di stabilità dell'addestramento**. I risultati discussi in questa sezione si riferiscono invece **esclusivamente alla fase di inferenza**, in cui vengono valutate: (i) l'**accuratezza decisionale sul task di placement** (Top-1 accuracy e Top-3 accuracy) e (ii) il **throughput di inferenza**, espresso in *samples/s*, come misura dell'efficienza computazionale.

La Tabella 7.3 riassume i risultati dell'ablation study sull'impatto del numero di visual patches nel task di placement, riportando congiuntamente la Top-1 e la Top-3 accuracy sui set di validation e test e il throughput di inferenza espresso in samples/s. I valori mostrati consentono di analizzare in modo diretto il trade-off tra qualità decisionale e costo computazionale al variare della risoluzione visiva fornita al modello.

L'analisi dell'ablation study evidenzia in modo chiaro l'impatto del numero di *visual patches* sia sull'accuratezza decisionale sia sulle prestazioni computazionali

Visual patches	**Samples/s**	**Acc. (val)**		**Acc. (test)**	
		Top-1	**Top-3**	**Top-1**	**Top-3**
4	**5.275**	0.1250	0.2979	0.1234	0.3000
32	5.163	0.1042	0.2542	0.1053	0.3223
64	5.106	0.3208	0.5458	0.2862	0.5809
128	5.054	0.6021	0.8854	0.5809	0.8840
256	4.731	0.6854	0.9708	0.6574	0.9734
512	4.088	**0.7229**	**0.9875**	**0.6894**	**0.9777**

Tabella 7.3: Ablation study sull'impatto del numero di visual patches nel task di placement. Sono riportate le accuratezze Top-1 e Top-3 su validation e test set, insieme al throughput di inferenza (samples/s).

del task di *placement*. In particolare, il comportamento del modello mostra una forte dipendenza dalla granularità della rappresentazione visiva adottata.

Nel regime a bassa risoluzione visiva (4 e 32 patches), il modello opera su una rappresentazione fortemente compressa della scena, che risulta insufficiente per discriminare in modo affidabile tra posizioni candidate caratterizzate da costi di interferenza simili. In questa configurazione, le prestazioni rimangono limitate sia in termini di Top-1 accuracy, che si attesta su valori inferiori a **0.13** sul test set, sia in termini di Top-3 accuracy, che non supera **0.32**. Tali risultati indicano una difficoltà marcata nel catturare le informazioni spaziali necessarie per un confronto fine tra alternative di placement plausibili.

All'aumentare del numero di patches ($64 \rightarrow 128$), si osserva un miglioramento significativo e progressivo delle prestazioni. In questa fascia intermedia, l'aumento della risoluzione visiva consente al modello di localizzare con maggiore precisione le regioni candidate, riducendo l'ambiguità decisionale. In particolare, a **128 patches** la Top-3 accuracy sul test set raggiunge **0.884**, mentre la Top-1 accuracy supera **0.58**, suggerendo che il modello è in grado di identificare correttamente la posizione ottimale in una frazione consistente dei casi.

Il miglioramento diventa ulteriormente marcato a partire da **256 patches**, dove la Top-3 accuracy sul test set supera **0.97**, fino a raggiungere il valore massimo di **0.9777** con **512 patches**. In questo regime ad alta risoluzione, il modello riesce a confrontare in modo affidabile patch di placement spazialmente vicine, catturando differenze sottili nella distribuzione della salienza sulla *combined map*. Tale comportamento si riflette anche nell'incremento della Top-1 accuracy, che raggiunge **0.6894** sul test set a 512 patches, indicando una maggiore confidenza nella selezione della posizione ottimale.

Dal punto di vista computazionale, l'aumento del numero di visual patches comporta una riduzione progressiva del throughput di inferenza, che passa da circa

5.3 samples/s a **4.1 samples/s**. Questo comportamento riflette l'aumento del numero di token visivi elaborati dal modello e il conseguente costo computazionale del meccanismo di self-attention multimodale. Tuttavia, questo decremento risulta contenuto e non compromette l'utilizzabilità del sistema nel contesto applicativo considerato. È infatti importante sottolineare che il task di placement non opera a frequenza per-frame, ma viene attivato in modo **condizionale**, tipicamente quando il task di *visibility* restituisce una label negativa per l'overlay corrente. In tale scenario, una frequenza decisionale dell'ordine di **3–4 samples/s** risulta pienamente compatibile con un utilizzo *near real-time* in sistemi di Mixed Reality.

Nel complesso, questa ablation study conferma che il task di *placement* è significativamente **più sensibile alla granularità visiva** rispetto al task di *visibility*. Tale differenza è riconducibile non solo ai requisiti di comprensione spaziale, ma anche alla maggiore **complessità intrinseca del task**. Mentre il task di *visibility* richiede una decisione binaria (*mostrare* o *non mostrare* l'overlay), il task di *placement* implica la selezione di una specifica posizione tra un insieme di alternative candidate, rappresentate tramite identificatori alfabetici assegnati in modo casuale. Decidere *dove* posizionare un overlay richiede pertanto una valutazione comparativa più fine tra regioni spazialmente vicine, basata su differenze sottili nella distribuzione della salienza e dell'interferenza sulla *combined map*. Di conseguenza, il numero di visual patches emerge come un **iperparametro critico**, da calibrare attentamente al fine di bilanciare accuratezza decisionale, stabilità delle predizioni e vincoli computazionali dello scenario applicativo.

Impatto del LoRA rank su accuratezza e performance

Oltre all'analisi dell'impatto della risoluzione visiva, abbiamo condotto un'ulteriore *ablation study* sul **LoRA rank**, con l'obiettivo di valutare come la **capacità parametrica degli adapter LoRA** influenzi le prestazioni del modello nel task di *placement*, sia in termini di **accuratezza decisionale** sia di **efficienza computazionale in inferenza**.

Tutti gli esperimenti sono stati condotti mantenendo invariata la configurazione di base adottata nelle analisi precedenti: **Qwen2.5-VL-3B** come backbone, **4 epoche di fine-tuning**, **learning rate pari a** 1×10^{-4} e scheduler *constant*. L'unico iperparametro variato è stato il **LoRA rank**, esplorando i valori 4**,** 8**,** 16 **e** 32, al fine di analizzare il compromesso tra capacità di adattamento del modello e costo computazionale.

Alla luce dei risultati ottenuti nell'ablation sul numero di visual patches, per ciascun valore di LoRA rank sono state considerate diverse configurazioni visive rappresentative, in modo da coprire regimi di informazione visiva e prestazioni computazionali differenti. In particolare, sono state incluse configurazioni con **4**,

128, **256** e **512 visual patches**, che spaziano da una rappresentazione fortemente compressa a una rappresentazione ad alta risoluzione visiva.

Questa scelta consente di analizzare in modo sistematico l'interazione tra **capacità parametrica degli adapter LoRA** e **granularità della rappresentazione visiva**, valutando se e in quali condizioni un aumento del LoRA rank produca benefici misurabili nel task di placement.

LoRA rank	**Patches**	**Samples/s**	**Acc. (val)**		**Acc. (test)**	
			Top-1	**Top-3**	**Top-1**	**Top-3**
4	4	5.283	0.1187	0.2812	0.1074	0.3021
4	128	5.056	0.5938	0.8583	0.5894	0.8840
4	256	4.627	0.7000	0.9771	0.6564	0.9691
4	512	4.085	**0.7312**	0.9854	0.6904	**0.9819**
8	4	5.168	0.0979	0.2542	0.1085	0.3223
8	128	4.994	0.5875	0.8646	0.5734	0.8723
8	256	4.674	0.7104	0.9771	0.6723	0.9755
8	512	3.954	0.7083	**0.9938**	**0.6947**	0.9745
16	4	5.226	0.1187	0.2812	0.1074	0.3021
16	128	5.019	0.6125	0.8812	0.5798	0.8840
16	256	4.737	0.6958	0.9729	0.6415	0.9628
16	512	4.083	0.7250	0.9833	0.6872	0.9766
32	4	**5.275**	0.1250	0.2979	0.1074	0.3000
32	128	5.054	0.6021	0.8854	0.5809	0.8840
32	256	4.731	0.6854	0.9708	0.6574	0.9734
32	512	4.088	0.7229	0.9875	0.6894	0.9777

Tabella 7.4: Ablation study sull'impatto combinato del LoRA rank e della risoluzione visiva nel task di placement. Sono riportate le accuratezze Top-1 e Top-3 su validation e test set, insieme al throughput di inferenza (samples/s).

La Tabella 7.4 riporta i risultati dell'ablation study, consentendo di analizzare congiuntamente l'effetto del LoRA rank e del numero di visual patches su accuratezza e throughput.

L'analisi congiunta dell'impatto del *LoRA rank* e della risoluzione visiva evidenzia in modo chiaro che, nel task di *placement*, la qualità della rappresentazione spaziale costituisce il fattore determinante per le prestazioni del modello, mentre l'aumento della capacità parametrica degli adapter LoRA gioca un ruolo secondario.

In primo luogo, si osserva che una rappresentazione visiva estremamente grossolana (**4 patches**) risulta fortemente limitante per il task di placement. In questo regime, sia la Top-1 sia la Top-3 accuracy rimangono molto basse e instabili per tutti i valori di LoRA rank considerati, con valori di Top-1 accuracy sul test set

prossimi a **0.10** e di Top-3 accuracy inferiori a **0.33**. Tali risultati indicano che l'incremento della capacità parametrica degli adapter non è in grado di compensare la mancanza di informazione spaziale, confermando che la **granularità visiva rappresenta il fattore dominante** nelle configurazioni a bassa risoluzione.

Passando a una risoluzione intermedia (**128 patches**), si osserva un netto miglioramento delle prestazioni rispetto al caso estremo. In questo setting, la Top-3 accuracy sul test set si attesta stabilmente attorno a **0.87–0.88** per tutti i valori di LoRA rank, mentre la Top-1 accuracy rimane compresa tra **0.57** e **0.59**. Anche in questo regime, la dipendenza dalle dimensioni del LoRA rank risulta debole, suggerendo che la rappresentazione visiva, pur significativamente più informativa, non è ancora pienamente sufficiente per una selezione sistematica della posizione ottimale.

Un ulteriore salto qualitativo emerge a partire da **256 patches**, dove le prestazioni aumentano in modo consistente per tutte le configurazioni di LoRA rank. In questo regime, la Top-3 accuracy sul test set supera sistematicamente **0.96**, mentre la Top-1 accuracy raggiunge valori compresi tra **0.64** e **0.67**. Ciò indica che il modello dispone finalmente di una rappresentazione spaziale sufficientemente fine per confrontare in modo affidabile posizioni candidate spazialmente vicine, riducendo in maniera significativa l'ambiguità intrinseca del task.

Le configurazioni a **512 patches** mostrano infine le prestazioni migliori complessive. In questo regime ad alta risoluzione visiva, la Top-3 accuracy sul test set raggiunge valori prossimi a **0.98**, mentre la Top-1 accuracy si colloca stabilmente intorno a **0.69**. L'effetto del LoRA rank risulta in questo caso limitato: differenze tra configurazioni con rank differenti sono contenute e non mostrano un andamento monotono, suggerendo che, una volta superata una soglia adeguata di granularità visiva, il collo di bottiglia non è più rappresentato dalla capacità parametrica degli adapter LoRA.

Dal punto di vista computazionale, l'impatto del LoRA rank sul throughput di inferenza risulta marginale rispetto a quello della risoluzione visiva. A parità di numero di patches, le variazioni nei *samples/s* al variare del rank sono contenute, mentre la riduzione del throughput è guidata principalmente dall'aumento del numero di token visivi elaborati dal modello. In termini di latenza per singolo campione, i valori osservati corrispondono indicativamente a tempi di elaborazione compresi tra **circa 190 ms** (per throughput prossimi a 5.2 samples/s) e **circa 250 ms** (per throughput intorno a 4.0 samples/s), a seconda della configurazione considerata.

È tuttavia importante sottolineare che tali tempi non devono essere interpretati come latenza per-frame. Il task di *placement*, infatti, non opera in modo continuo, ma viene attivato **in maniera condizionale**, tipicamente in seguito a una decisione negativa del task di *visibility* per l'overlay corrente. In questo scenario, una latenza dell'ordine di **200–250 ms per decisione** risulta pienamente compatibile con un

utilizzo *near real-time* in sistemi di Mixed Reality, nei quali la frequenza decisionale è dell'ordine di pochi Hertz e il placement non rientra nel ciclo critico di rendering o tracking.

Nel complesso, questa ablation study evidenzia che, nel task di *placement*, la **risoluzione visiva** rappresenta il fattore chiave per ottenere un'elevata accuratezza decisionale, mentre l'incremento del **LoRA rank** oltre valori moderati produce benefici limitati. In particolare, le configurazioni con **256–512 patches** e un **LoRA rank intermedio (8–16)** emergono come il miglior compromesso tra qualità decisionale ed efficienza computazionale, risultando pienamente compatibili con un utilizzo *near real-time* in scenari di Mixed Reality, nei quali il task di placement viene attivato in modo condizionale e a frequenze dell'ordine di pochi Hertz.

Discussione finale

Le *ablation studies* condotte sul task di *placement* mostrano in modo consistente che le prestazioni del modello sono dominate principalmente dalla **qualità della rappresentazione visiva** fornita al backbone Vision-Language, mentre la capacità parametrica introdotta dagli adapter LoRA riveste un ruolo secondario e tende a produrre benefici limitati una volta superata una soglia adeguata di dettaglio spaziale. Questo risultato riflette la natura intrinsecamente *fine-grained* del problema: il placement richiede un confronto comparativo tra più posizioni candidate spesso spazialmente vicine, caratterizzate da differenze sottili nella distribuzione della salienza e dell'interferenza sulla *combined map*.

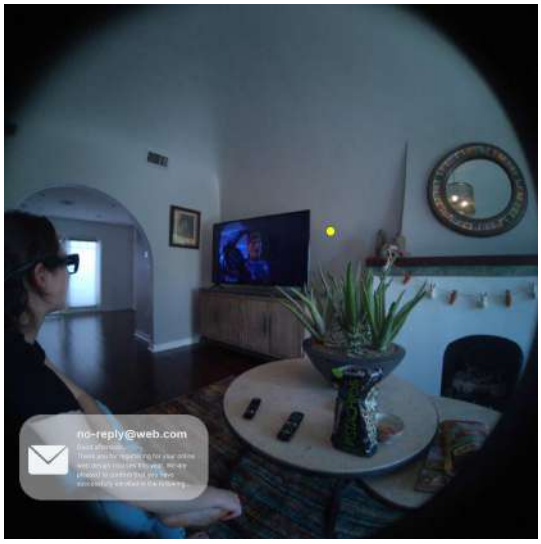
L'analisi dell'impatto del numero di *visual patches* evidenzia un chiaro trade-off tra **accuratezza decisionale** ed **efficienza computazionale**. A basse risoluzioni visive (4–32 patches), la rappresentazione della scena risulta eccessivamente compressa, rendendo il modello incapace di discriminare in modo affidabile tra alternative di placement plausibili: in questo regime, sia la Top-1 sia la Top-3 accuracy rimangono basse e instabili. L'incremento a 128 patches introduce un miglioramento significativo, ma è a partire da 256 patches che si osserva un salto qualitativo netto, con una crescita marcata e stabile dell'accuratezza. Le prestazioni migliori si ottengono infine con 512 patches, confermando che il task di placement beneficia in modo sostanziale di una rappresentazione visiva ad alta risoluzione, più di quanto avvenga nel task di *visibility*, che richiede una decisione globale e binaria.

L'ablation sul **LoRA rank** rafforza ulteriormente questa interpretazione. A parità di risoluzione visiva, l'aumento del rank non produce miglioramenti sistematici e monotoni né in termini di Top-1 né di Top-3 accuracy. I benefici più evidenti emergono solo in alcune configurazioni ad alta risoluzione, mentre in regimi a bassa o media granularità visiva l'effetto del LoRA rank risulta marginale. Questo

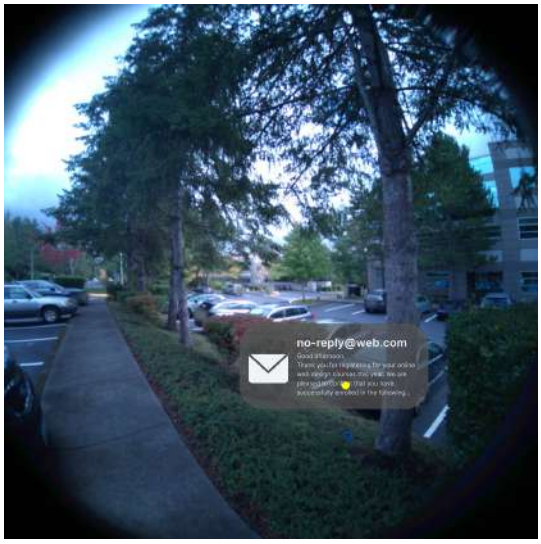
suggerisce che, nel task di placement, l'errore è più frequentemente riconducibile a una limitazione informativa (insufficiente dettaglio spaziale) piuttosto che a una limitazione di capacità del modello.

Dal punto di vista computazionale, entrambe le ablation studies mostrano che il throughput di inferenza è guidato principalmente dal numero di token visivi elaborati, evitando una forte dipendenza dal LoRA rank. I valori osservati, compresi indicativamente tra **4.0 e 5.3 samples/s**, corrispondono a latenze per singola decisione dell'ordine di **190–250 ms**. Tali tempi risultano pienamente compatibili con un utilizzo *near real-time* in scenari di Mixed Reality, considerando che il task di placement non opera a frequenza per-frame, ma viene attivato in modo **condizionale**, tipicamente in seguito a una decisione negativa del task di *visibility* per l'overlay corrente.

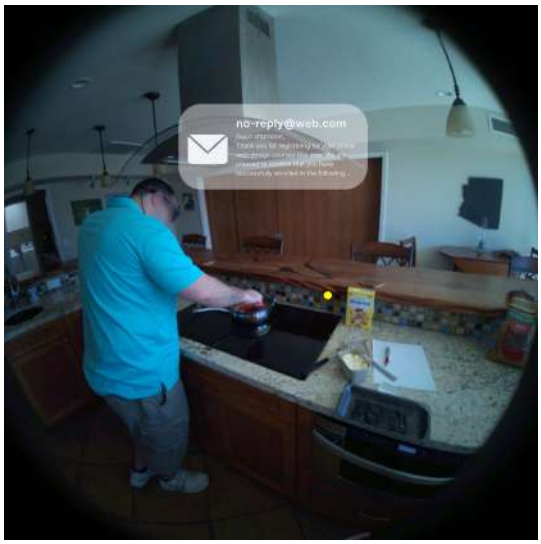
Nel complesso, queste analisi confermano che il task di *placement* rappresenta il componente più sensibile alla **granularità visiva** dell'intera pipeline. Un'adozione pratica in sistemi di Mixed Reality dovrebbe pertanto privilegiare configurazioni ad **elevata risoluzione visiva** (256–512 patches), abbinate a un **LoRA rank intermedio** (8–16), che offrono il miglior compromesso tra accuratezza decisionale, stabilità delle predizioni e costi computazionali realistici per scenari applicativi *head-mounted*.



(a) Reason: The overlay covers floor and person, which are relevant for user safety and social acceptability.



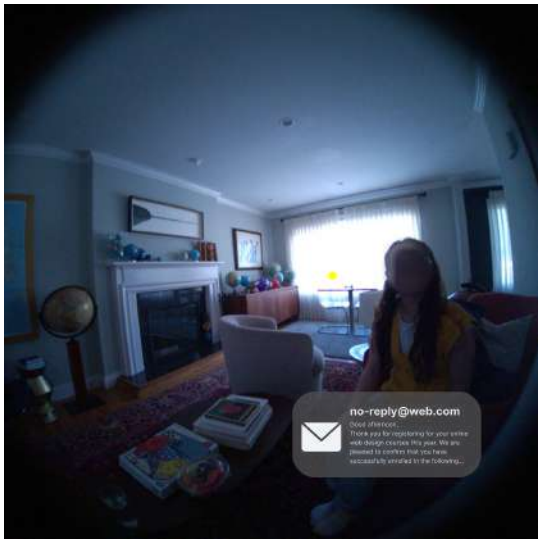
(b) Reason: The overlay covers a part of functional element (car) and covers a high aesthetic region.



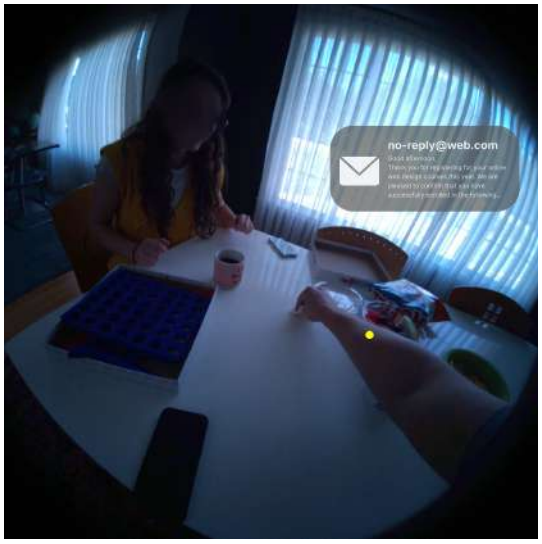
(c) Reason: The overlay does not occlude any important functional, safety, or aesthetic region.

(d) Reason: The overlay does not occlude any important functional, safety, or aesthetic region.

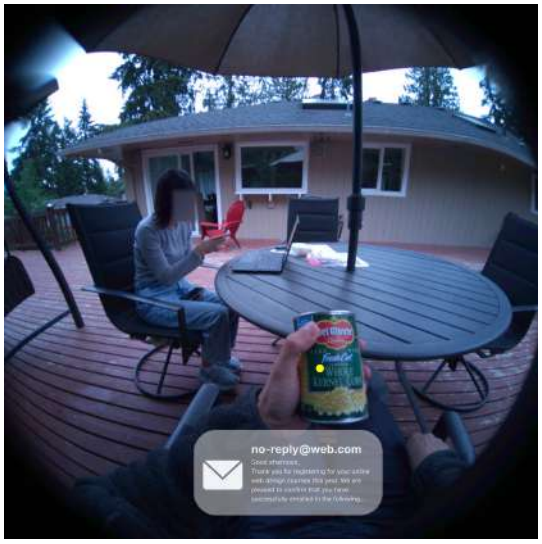
Figura 7.4: Esempi qualitativi di *reason* deterministica generate a partire dalle mappe contestuali per il task di visibility. Nei casi con label negativa (prima riga), la spiegazione evidenzia l'occlusione di regioni rilevanti per sicurezza/accettabilità sociale, funzionalità o estetica. Nei casi con label positiva (seconda riga), la *reason* segnala l'assenza di interferenze con contenuti funzionali, safety-critical o visivamente salienti.



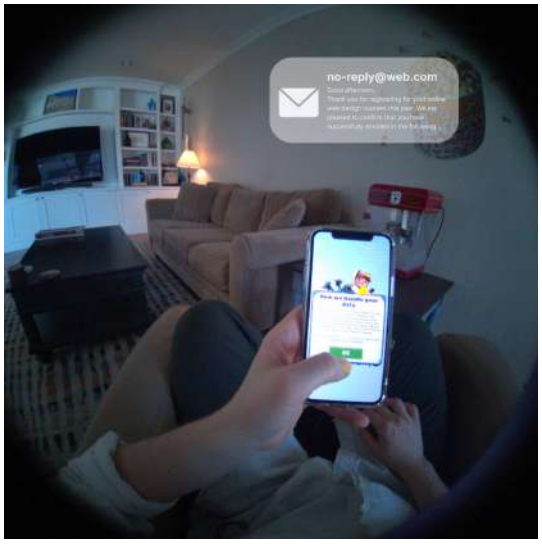
(a) Output: *0 The overlay covers floor and person, which are relevant for user safety and social acceptability.*



(b) Output: *1 The overlay does not occlude any important functional, safety, or aesthetic region.*

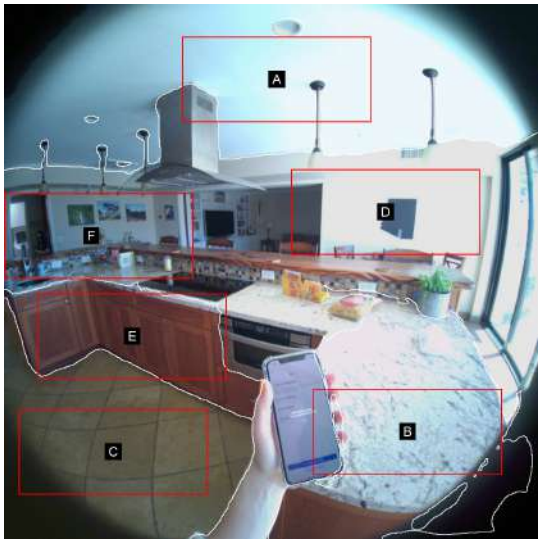


(c) Output: *0 The overlay covers floor and person, which are relevant for user safety and social acceptability.*

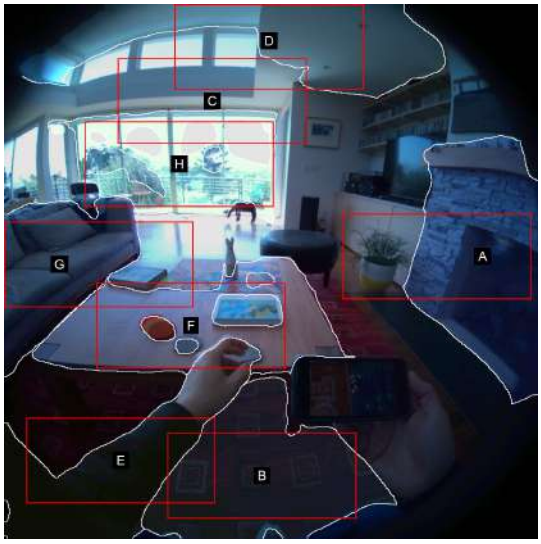


(d) Output: *1 The overlay does not occlude any important functional, safety, or aesthetic region.*

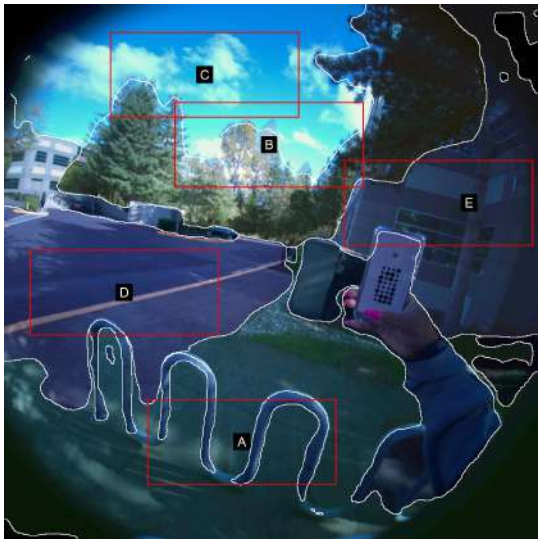
Figura 7.5: Esempi qualitativi di inferenza nel setting *label + reason*. Le risposte generate risultano coerenti con la label predetta e semanticamente allineate al contenuto dell'immagine, senza fenomeni di ripetizione o degenerazione osservati negli approcci con spiegazione libera.



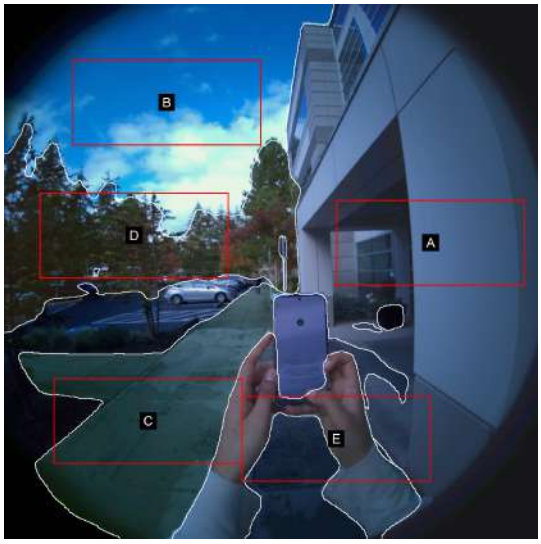
(a) Top-3: A, D, F



(b) Top-3: C, D, H



(c) Top-3: C, B, D



(d) Top-3: B, D, A

Figura 7.6: Esempi di scene di Mixed Reality che mostrano la presenza di più posizioni candidate valide all'interno di un singolo frame per il task di *placement*.

Capitolo 8

Multi-Task Learning (MTL)

Dopo aver dimostrato che un fine-tuning mirato del modello **Qwen2.5-VL** consente di ottenere prestazioni elevate sui singoli task di *visibility* e *placement*, il passo successivo di questo lavoro consiste nell'analizzare la possibilità di **unificare tali competenze all'interno di un unico modello**. Questo passaggio non rappresenta una semplice estensione sperimentale, ma introduce una problematica di portata più ampia, che riguarda la capacità dei modelli multimodali di gestire **decisioni eterogenee** all'interno di una singola architettura condivisa.

Nel contesto di questa tesi, tale obiettivo viene perseguito attraverso una strategia concreta e controllata: il fine-tuning di un **unico modello Qwen2.5-VL** utilizzando **l'unione dei dataset di training dei task di visibility e placement**. In questo modo, il modello viene esposto durante l'addestramento a istanze appartenenti a entrambi i task, ciascuna caratterizzata da input visivi simili ma da output supervisionati differenti, con l'obiettivo di apprendere una rappresentazione condivisa della scena in grado di supportare entrambe le decisioni.

In particolare, l'obiettivo di questo capitolo è studiare la *Multi-Task learning*, ovvero la capacità di un singolo modello **Vision–Language–Action (VLA)** di affrontare correttamente più compiti distinti, mantenendo prestazioni elevate su ciascuno di essi **senza ricorrere a modelli separati o a pipeline dedicate**. Nel contesto della Mixed Reality, questa prospettiva è particolarmente rilevante: un sistema reale non opera mai su task isolati, ma deve adattarsi continuamente a scenari dinamici e complessi, prendendo decisioni multiple e fortemente interdipendenti a partire da una singola osservazione visiva della scena.

8.1 Motivazioni e contesto

L'adozione di una strategia di **Multi-Task Learning (MTL)** [37] nasce dall'osservazione che, in scenari applicativi reali di Mixed Reality, i diversi compiti

di adattamento dell'interfaccia non vengono mai affrontati in modo isolato. Al contrario, decisioni quali *se* un elemento UI debba essere mostrato (*visibility*) e *dove* esso debba essere posizionato (*placement*) sono fortemente interdipendenti e si basano sulla medesima osservazione visiva della scena. Trattare tali compiti come problemi separati, affidati a modelli distinti o a pipeline indipendenti, introduce ridondanze computazionali e rischia di produrre decisioni incoerenti dal punto di vista semantico e percettivo.

Dal punto di vista del modello, *visibility* e *placement* condividono gran parte del processo di percezione visiva, ma differiscono in modo significativo per quanto riguarda la struttura dell'output e la granularità del ragionamento richiesto. Il task di *visibility* richiede una decisione globale e binaria, mentre il *placement* implica una selezione più fine-grained tra posizioni candidate spesso spazialmente vicine. Questa asimmetria rende il problema particolarmente interessante in ottica MTL: da un lato, esiste un'evidente opportunità di **condivisione delle rappresentazioni**; dall'altro, emerge il rischio di **interferenza tra task** dovuto a obiettivi supervisionati eterogenei.

Nel contesto dei modelli Vision–Language–Action, e in particolare dei Vision–Language Models autoregressivi come Qwen2.5-VL, l'MTL non viene implementato attraverso teste di output separate, ma tramite una formulazione unificata basata su prompt e target supervisionati differenti. Questo approccio consente di esplorare una forma di *parameter sharing* completa, in cui l'intero backbone del modello è condiviso tra i task, e la distinzione tra le competenze emerge esclusivamente a livello semantico e supervisionato.

L'obiettivo di questa sezione è quindi motivare e contestualizzare l'adozione del Multi-Task Learning come scelta architetturale e metodologica, evidenziando i potenziali benefici in termini di efficienza, coerenza decisionale e scalabilità del sistema, ma anche le sfide legate alla gestione di task con difficoltà e requisiti diversi. Tali aspetti risultano particolarmente rilevanti nel dominio della Mixed Reality, dove un singolo modello è chiamato a supportare decisioni multiple in tempo quasi reale a partire da una percezione condivisa della scena.

8.2 Multi-Task Learning: principi generali

Il **Multi-Task Learning (MTL)** è un paradigma di apprendimento automatico in cui un singolo modello viene addestrato per risolvere simultaneamente più compiti correlati, sfruttando una rappresentazione condivisa dei dati di input. L'idea di fondo è che l'apprendimento congiunto di task affini possa introdurre un *inductive bias* benefico, migliorando la capacità di generalizzazione del modello rispetto all'addestramento su ciascun task in modo indipendente. Questo principio è stato formalizzato inizialmente in ambito di machine learning classico e successivamente

esteso ai modelli neurali profondi, dove la condivisione dei parametri consente di apprendere rappresentazioni più robuste e meno soggette a overfitting.

Con il termine *inductive bias* si intende l'insieme di assunzioni implicite che guidano il modello nel processo di generalizzazione a dati non visti, ovvero le restrizioni a priori sul tipo di funzioni che il modello può apprendere. Tali assunzioni possono derivare, ad esempio, dalla scelta dell'architettura (che impone specifiche modalità di elaborazione dell'informazione), dalla formulazione del problema o dalla strategia di addestramento adottata. Nel contesto del Multi-Task Learning, l'inductive bias è introdotto in modo diretto dalla condivisione dei parametri tra task differenti, che costringe il modello ad apprendere rappresentazioni comuni e riutilizzabili a partire dagli stessi input. Questo vincolo riduce la probabilità di apprendere correlazioni spurie specifiche di un singolo task e favorisce l'emergere di feature più generali, stabili e robuste, utili per supportare decisioni multiple basate su una percezione condivisa della scena.

Nel contesto delle *Deep Neural Networks*, il MTL viene tipicamente realizzato attraverso meccanismi di **parameter sharing**. La strategia più comune è il cosiddetto *hard parameter sharing*, in cui i livelli iniziali della rete sono completamente condivisi tra tutti i task, mentre le eventuali differenze vengono gestite a livello di output o di supervisione. Questo approccio riduce il numero complessivo di parametri, migliora l'efficienza computazionale e favorisce l'apprendimento di feature generiche utili a più compiti. Alternative più flessibili, come il *soft parameter sharing*, prevedono reti parzialmente separate con vincoli di similarità tra i parametri, ma introducono una complessità architetturale maggiore.

Un aspetto centrale del MTL è il **bilanciamento tra trasferimento positivo e interferenza negativa**. Quando i task sono sufficientemente correlati, la condivisione delle rappresentazioni può portare a un miglioramento delle prestazioni su ciascun compito (*positive transfer*). Al contrario, quando i task presentano obiettivi conflittuali o livelli di difficoltà molto differenti, l'addestramento congiunto può degradare le prestazioni di uno o più task, fenomeno noto come *negative transfer*. La letteratura evidenzia come tale interferenza sia particolarmente rilevante in presenza di supervisioni eterogenee o di task con granularità decisionali diverse.

Nei modelli multimodali e, in particolare, nei *Vision–Language Models* autoregressivi, il Multi-Task Learning assume una forma peculiare. In questi modelli, l'architettura è tipicamente composta da un backbone visivo e da un modello linguistico condiviso, e i diversi task vengono distinti non attraverso teste di output dedicate, ma mediante **prompt differenti** e **target supervisionati diversi**. In questo setting, il MTL emerge come una conseguenza naturale dell'addestramento su dataset eterogenei, in cui il modello è chiamato a generare output differenti a partire da input visivi simili, mantenendo una rappresentazione interna unificata della scena.

Nel contesto di questa tesi, il Multi-Task Learning viene implementato seguendo

una strategia deliberatamente semplice e controllata: il fine-tuning di un unico modello **Qwen2.5-VL** su un dataset che comprende istanze appartenenti sia al task di *visibility* sia al task di *placement*. I due task condividono la stessa struttura di input multimodale, ma differiscono per natura dell'output supervisionato e per la granularità del ragionamento richiesto. Questa impostazione consente di studiare in modo diretto se e in che misura un modello Vision–Language–Action sia in grado di apprendere una rappresentazione condivisa della scena che supporti decisioni eterogenee, senza ricorrere a meccanismi espliciti di pesatura delle loss o a architetture multi-testa.

In sintesi, il Multi-Task Learning rappresenta in questo lavoro non solo una tecnica di ottimizzazione, ma un banco di prova per valutare la capacità dei modelli multimodali di integrare percezione visiva e decisioni complesse all'interno di un'unica architettura coerente. Tale prospettiva è particolarmente rilevante per applicazioni di Mixed Reality, in cui più task interdipendenti devono essere risolti in modo efficiente, consistente e scalabile a partire da una singola osservazione della scena.

8.3 Setup sperimentale

Per studiare il comportamento del modello in regime di *Multi-Task Learning*, è stato definito un setup sperimentale che ricalca il più possibile quello adottato nei fine-tuning single-task, con l'obiettivo di isolare l'effetto specifico dell'addestramento congiunto dei task di *visibility* e *placement*. In particolare, l'analisi mira a valutare se un unico modello Vision–Language–Action sia in grado di mantenere prestazioni competitive su entrambi i task quando addestrato su un dataset eterogeneo, senza ricorrere a modifiche architetturali dedicate o a meccanismi espliciti di bilanciamento delle loss.

Modello e configurazione di base. Tutti gli esperimenti di Multi-Task Learning sono stati condotti utilizzando il modello **Qwen2.5-VL-3B** come backbone condiviso. Il fine-tuning è stato eseguito per **4 epoche**, con **learning rate pari a** 1×10^{-4} e scheduler di tipo *constant*. L'adattamento dei pesi è stato realizzato tramite **LoRA**, mantenendo la stessa impostazione adottata negli esperimenti single-task, al fine di garantire un confronto equo. Salvo diversa indicazione, il LoRA rank è stato fissato a un valore intermedio, selezionato sulla base dei risultati emersi dalle ablation studies precedenti.

Dataset multi-task. Il dataset utilizzato per il fine-tuning multi-task è costruito come **unione dei dataset di training** dei task di *visibility* e *placement*. Ogni istanza è composta da:

1. un input visivo (frame RGB)
2. un prompt testuale che specifica implicitamente il task da svolgere
3. un output supervisionato coerente con il task di riferimento

Le istanze appartenenti ai due task condividono la medesima struttura multimodale di input, ma differiscono per formato e semantica dell'output generato.

Durante l'addestramento, il modello viene esposto in modo alternato a esempi di visibility e placement, senza introdurre una distinzione esplicita a livello architetturale. La natura del task da risolvere è determinata esclusivamente dal prompt e dalla struttura della risposta supervisionata, coerentemente con il paradigma dei Vision–Language Models autoregressivi. I prompt utilizzati per il task di *visibility* sono riportati in Appendice A.1.3, mentre quelli relativi al task di *placement* sono descritti in Appendice A.2.

Formulazione della supervisione. La supervisione è applicata seguendo lo stesso principio adottato nei fine-tuning single-task. Per ciascuna istanza, la funzione di loss viene calcolata esclusivamente sui token dell'output dell'assistente rilevanti per il task corrente, mentre tutte le altre porzioni della sequenza vengono mascherate. In questo modo, il modello è incentivato ad apprendere correttamente sia la decisione binaria del task di visibility (accompagnata dalla *reason*), sia la selezione della posizione nel task di placement, evitando interferenze dirette a livello di loss tra i due task.

Assenza di pesatura esplicita delle loss. Nel presente lavoro non viene introdotto alcun meccanismo di pesatura dinamica o statica delle loss tra i task. Tutte le istanze contribuiscono all'ottimizzazione in modo uniforme, e l'equilibrio tra visibility e placement è demandato esclusivamente alla composizione del dataset e alla condivisione dei parametri del modello. Questa scelta consente di analizzare il comportamento del Multi-Task Learning in una configurazione semplice e controllata, evidenziando in modo diretto eventuali fenomeni di trasferimento positivo o interferenza negativa.

Protocollo di valutazione. La valutazione delle prestazioni del modello multi-task viene condotta separatamente sui due task, utilizzando le stesse metriche e gli stessi split di validation e test adottati negli esperimenti single-task. In particolare, per il task di visibility viene considerata l'accuratezza della label binaria, mentre per il task di placement vengono analizzate le metriche di Top-1 e Top-3 accuracy. Questo protocollo consente un confronto diretto e quantitativo tra il modello multi-task e le corrispondenti configurazioni single-task.

Scelta delle configurazioni multi-task. Alla luce dei risultati ottenuti nei fine-tuning single-task, il Multi-Task Learning è stato progettato selezionando una configurazione che rappresentasse un **compromesso realistico tra accuratezza decisionale ed efficienza computazionale** per entrambi i task. In particolare, anziché adottare per ciascun task la configurazione ottimale in termini di prestazioni assolute, è stata selezionata una configurazione comune ottenuta considerando l'**intersezione delle regioni di funzionamento più critiche** emerse dalle analisi single-task.

Questa scelta equivale a privilegiare configurazioni robuste anche nei *casi peggiori* tra visibility e placement, evitando soluzioni fortemente sbilanciate a favore di un singolo task. L'obiettivo è simulare uno scenario applicativo realistico, in cui un singolo modello deve operare sotto vincoli condivisi di latenza e throughput, mantenendo prestazioni accettabili su entrambe le decisioni.

Sulla base di tali considerazioni, sono state valutate quattro configurazioni candidate, ottenute combinando due valori del numero di visual patches (**256** e **512**) e due valori del LoRA rank (**8** e **16**), mantenendo invariati tutti gli altri iperparametri.

8.4 Risultati sperimentali

In questa sezione vengono presentati e analizzati i risultati sperimentali ottenuti dal modello addestrato in regime di *Multi-Task Learning*, con l'obiettivo di valutare l'impatto dell'addestramento congiunto dei task di *visibility* e *placement* sulle prestazioni complessive del sistema. I risultati sono interpretati alla luce della configurazione sperimentale descritta nella Sezione 8.3, deliberatamente scelta come compromesso tra accuratezza decisionale ed efficienza computazionale per entrambi i task.

Numero patches	**LoRA rank**	**samples/s**	**Visibility**		**Placement**			
			Val Acc.	**Test Acc.**	**Val Acc.**		**Test Acc.**	
					Top-1	**Top-3**	**Top-1**	**Top-3**
256	8	**3.591**	0.8851	0.8688	0.6681	0.9734	**0.7188**	0.9708
256	16	2.644	0.9043	0.8979	0.6457	0.9564	0.7146	0.9708
512	8	3.102	0.9096	0.9062	**0.6951**	**0.9851**	0.6958	**0.9854**
512	16	3.274	**0.9138**	**0.9083**	0.6851	0.9766	0.7042	0.9812

Tabella 8.1: Risultati di accuratezza su *visibility* e *placement* al variare del numero di visual patches e del LoRA rank. Per il task di placement sono riportate le metriche Top-1 e Top-3 su validation e test set.

L'analisi dei risultati riportati in Tabella 8.1 evidenzia alcune considerazioni rilevanti sul comportamento del modello quando addestrato in regime di *Multi-Task Learning*.

In primo luogo, l'aumento del numero di *visual patches* produce benefici consistenti nel task di *placement*. Le configurazioni basate su **512 patches** superano sistematicamente quelle a **256 patches** in termini di accuratezza Top-1 e Top-3, sia nel validation set sia nel test set. Questo comportamento è coerente con quanto osservato negli esperimenti single-task e conferma che il problema di placement richiede una rappresentazione spaziale della scena sufficientemente dettagliata. Una risoluzione visiva più elevata consente infatti al modello di discriminare in modo più affidabile tra regioni candidate spesso molto vicine tra loro nello spazio dell'immagine.

Per quanto riguarda il task di *visibility*, l'incremento del numero di patches produce un miglioramento più contenuto ma comunque stabile. Il passaggio da 256 a 512 patches determina un incremento dell'accuratezza sia in validation sia in test set, suggerendo che una rappresentazione visiva più fine-grained non penalizza la decisione binaria richiesta dal task, pur non costituendo un fattore determinante per il raggiungimento di buone prestazioni.

Analizzando invece l'effetto del *LoRA rank*, emerge un comportamento differenziato tra i due task. Nel task di *visibility*, l'aumento del rank da 8 a 16 comporta un miglioramento marginale ma consistente delle prestazioni, in particolare nella configurazione con 512 patches. Al contrario, nel task di *placement*, l'incremento del rank non produce benefici sistematici e, in alcuni casi, risulta leggermente penalizzante. Questo suggerisce che, una volta garantita una rappresentazione visiva sufficientemente ricca, la capacità parametrica aggiuntiva introdotta da un rank più elevato degli adapter LoRA non costituisce il principale fattore limitante per questo task.

Nel complesso, la configurazione con **512 patches e LoRA rank 8** emerge come il miglior compromesso tra i due task. Tale configurazione consente infatti di ottenere le migliori prestazioni complessive sul placement, mantenendo al contempo un'elevata accuratezza nel task di visibility. Questo risultato indica che l'addestramento congiunto non introduce una degradazione significativa delle prestazioni rispetto agli scenari single-task e che il modello Vision–Language–Action è in grado di apprendere una rappresentazione condivisa della scena sufficientemente espressiva da supportare decisioni eterogenee.

Dal punto di vista computazionale, i risultati mostrano inoltre che il modello mantiene un'elevata efficienza di inferenza. Come riportato nella Tabella 8.1, il throughput varia tra circa **2.6 e 3.5 samples/s** a seconda della configurazione, corrispondenti a un tempo medio di inferenza compreso tra circa **280 ms e 380 ms per sample**. Ciò significa che il modello è in grado di produrre una decisione completa per ciascun input visivo in poche centinaia di millisecondi, un aspetto

fondamentale per garantire un'interazione fluida in scenari di Mixed Reality.

Nel complesso, tali evidenze confermano la validità dell'approccio Multi-Task Learning proposto e suggeriscono che, in scenari di Mixed Reality realistici, un singolo modello possa sostituire pipeline dedicate per task separati, garantendo un buon compromesso tra accuratezza, coerenza decisionale ed efficienza computazionale.

Oltre ai risultati quantitativi, la Figura 8.1 mostra due esempi qualitativi di inferenza prodotti dal modello multi-task. Questi esempi permettono di osservare il comportamento del modello in scenari concreti e di comprendere come la stessa architettura sia in grado di produrre decisioni differenti a seconda del task richiesto dal prompt.

(a) Esempio di inferenza di *Visibility*. **Output del modello**: *0 The overlay covers the person's hand holding the phone, which is a functional element the user is interacting with.*

(b) Esempio di inferenza di *Placement*. **Output del modello**: *A*

Figura 8.1: Esempi qualitativi delle predizioni del modello nel contesto di *multi-task learning*. A seconda del prompt, il modello genera output specifici per il task: un identificatore della regione candidata per il task di *placement* e una decisione binaria accompagnata da una breve spiegazione per il task di *visibility*.

Capitolo 9

Integrazione pipeline su HoloLens 2

9.1 Architettura generale della pipeline

L'integrazione della pipeline **UiVLA** all'interno del visore *HoloLens 2* è stata progettata come un sistema distribuito, in cui la componente di decisione basata su modello Vision–Language–Action è fisicamente separata dal dispositivo, ma logicamente integrata nel flusso dell'applicazione di Mixed Reality.

In particolare, l'applicazione in esecuzione sul visore svolge il ruolo di *client*, responsabile dell'acquisizione dei dati sensoriali, della gestione del contesto e dell'aggiornamento dell'interfaccia utente. La componente di inferenza basata sul modello fine-tunato nel Capitolo 8 è invece ospitata su un server remoto, al quale il visore invia richieste HTTP contenenti i frame acquisiti e riceve in risposta le decisioni relative ai task di *visibility* e *placement*.

L'obiettivo principale di questa architettura è consentire all'interfaccia utente di adattarsi dinamicamente al contesto reale, attivando la rivalutazione di visibilità e posizionamento esclusivamente quando avviene un cambiamento significativo nella scena. Questo approccio consente di:

- ridurre il carico computazionale sul dispositivo
- evitare inferenze ridondanti in condizioni di scena statica
- mantenere una separazione chiara tra logica applicativa e logica di decisione basata su modello

La Figura 9.1 illustra l'architettura generale del sistema e il flusso delle informazioni tra i diversi moduli.

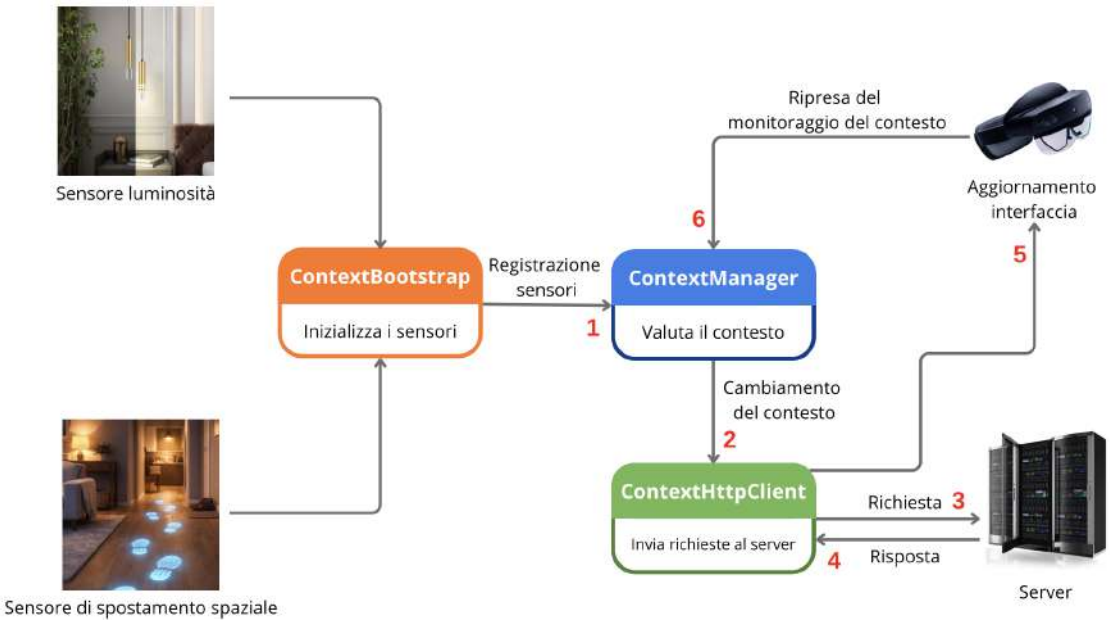


Figura 9.1: Architettura generale della pipeline UiVLA integrata su HoloLens 2. I sensori attivano il *ContextManager*, che coordina le richieste al server remoto e l'aggiornamento dell'interfaccia.

9.2 Rilevamento del cambiamento di contesto

Per determinare quando attivare la pipeline di inferenza, è stato progettato un sistema basato su **sensori concettuali di contesto**. Il principio alla base di questo meccanismo è che l'interfaccia utente non debba essere rivalutata in modo continuo, ma soltanto quando l'ambiente circostante subisce una variazione sufficientemente significativa da poterne influenzare visibilità o posizionamento.

Nel sistema implementato sono stati considerati due sensori principali:

- **Sensore di luminosità.** Rileva variazioni significative nei livelli di illuminazione della scena osservata dall'utente. Cambiamenti marcati nella luminosità possono influenzare la leggibilità o il contrasto dell'interfaccia virtuale.

- **Sensore di spostamento significativo.** Rileva variazioni rilevanti della posizione della testa dell'utente nello spazio tridimensionale, utilizzando come riferimento il *transform* della camera principale. Il sensore si basa esclusivamente sulla componente di traslazione (posizione 3D nel sistema di riferimento globale), ignorando le rotazioni, e rappresenta una misura della rilocazione spaziale dell'utente nell'ambiente fisico.

È importante sottolineare che il rilevamento del cambiamento di contesto non costituisce il focus principale del presente lavoro. L'obiettivo non è proporre un sistema avanzato di *context awareness*, né ottimizzare in modo esaustivo le strategie di rilevamento ambientale. I sensori rappresentano piuttosto un **meccanismo di attivazione** (trigger), ovvero una condizione necessaria per innescare la rivalutazione decisionale.

Il vero interesse del sistema risiede infatti in ciò che accade **dopo** il rilevamento del cambiamento. Il punto centrale non è tanto "come" venga identificata una variazione di contesto, quanto "come" il sistema reagisca una volta che tale variazione è stata rilevata.

I segnali generati dai sensori vengono coordinati e interpretati da un componente centrale denominato **ContextManager**. È questo modulo a stabilire se il cambiamento osservato sia sufficientemente rilevante da attivare una nuova iterazione della pipeline, trasformando un evento ambientale in un'azione concreta del sistema, ossia una richiesta di inferenza remota.

In questa prospettiva, il rilevamento del contesto rappresenta il mezzo e non il fine: esso costituisce la condizione necessaria per avviare la parte centrale dell'architettura, ossia la rivalutazione di *visibility* e *placement* attraverso il modello Vision–Language–Action.

9.3 Architettura del client (HoloLens 2)

Dopo aver descritto l'architettura generale e il meccanismo di rilevamento del cambiamento di contesto, è ora opportuno analizzare nel dettaglio l'architettura del modulo client in esecuzione su HoloLens 2.

Il client rappresenta il livello operativo della pipeline UiVLA: esso integra i sensori, coordina la logica di attivazione e gestisce la comunicazione con il server remoto. L'architettura è stata progettata secondo principi di modularità e separazione delle responsabilità, così da garantire estendibilità, manutenibilità e controllo delle prestazioni.

9.3.1 Architettura di gestione del contesto

La gestione del cambiamento di contesto all'interno della pipeline UiVLA non è affidata ad un singolo componente, ma è il risultato della collaborazione tra tre moduli distinti e chiaramente separati nelle responsabilità:

- **ContextBootstrap**
- **ContextManager**
- **ContextHttpClient**

Questa suddivisione risponde a una scelta architetturale esplicita orientata a modularità, testabilità e chiarezza dei ruoli.

Ruolo del ContextBootstrap

La fase di inizializzazione del sistema è delegata al componente **ContextBootstrap**. Questo modulo si occupa esclusivamente della creazione e registrazione dei sensori durante l'avvio dell'applicazione (metodo `Awake`).

Nel sistema sono stati implementati due sensori:

- il **sensore di variazione di luminosità**, basato su un oggetto che implementa l'interfaccia `IBrightnessProvider`
- il **sensore di spostamento significativo**, che utilizza il `Transform` della camera principale per monitorare la rilocazione spaziale dell'utente

Il ContextBootstrap verifica la corretta configurazione dei componenti necessari (presenza del provider di luminosità, riferimento valido alla camera, parametri di soglia coerenti) e registra i sensori presso il ContextManager tramite il metodo `Register`.

Questa separazione evita di mescolare la fase di setup con la logica decisionale. Il ContextManager non ha conoscenza dei dettagli di creazione dei sensori, ma opera esclusivamente su un insieme astratto di componenti che implementano l'interfaccia comune `IContextSensor`.

Tale scelta facilita l'estensione futura del sistema: l'aggiunta di nuovi sensori non richiederà modifiche alla logica interna del ContextManager, ma esclusivamente l'estensione del bootstrap.

Ruolo del ContextManager

Il **ContextManager** rappresenta il nucleo decisionale del sistema. Esso monitora continuamente lo stato dei sensori registrati e determina se il cambiamento osservato sia sufficientemente significativo da attivare la pipeline di inferenza.

Ogni sensore aggiorna periodicamente il proprio stato; tuttavia, non ogni variazione genera automaticamente una richiesta al server. Il ContextManager applica una logica di filtro che può includere:

- politiche di attivazione (ad esempio attivazione con almeno un sensore triggerato, oppure con almeno N su M)
- meccanismi di *edge detection*, attivando la pipeline solo nella transizione da stato non-triggerato a triggerato
- intervalli di *cooldown*, per evitare attivazioni ravvicinate

Questo approccio impedisce che micro-variazioni o rumore temporaneo generino richieste ridondanti verso il server remoto, riducendo traffico di rete e latenza percepita.

Quando viene rilevato un cambiamento significativo, il ContextManager non esegue direttamente la comunicazione con il server, ma emette un evento (`OnContextChanged`), mantenendo così la separazione tra decisione logica e azione operativa.

Ruolo del ContextHttpClient

Il **ContextHttpClient** rappresenta il componente operativo incaricato di trasformare una decisione logica in un'azione concreta all'interno della pipeline distribuita. Esso si sottoscrive all'evento `OnContextChanged` emesso dal *ContextManager* e interviene esclusivamente quando viene rilevato un cambiamento di contesto ritenuto significativo.

Una volta attivato, il modulo avvia la fase esecutiva della pipeline: acquisisce il frame corrente dalla camera RGB del visore, gestisce l'eventuale compositing con l'overlay dell'interfaccia utente già presente nella scena e prepara l'immagine per l'invio remoto attraverso una fase di compressione (formato JPEG). L'immagine così ottenuta viene quindi trasmessa al server tramite una richiesta HTTP contenente le informazioni necessarie per l'inferenza.

Nel sistema UiVLA, la spiegazione generata dal modello svolge un ruolo importante dal punto di vista interpretativo. Essa consente infatti di comprendere le ragioni alla base della decisione presa dal modello e permette di verificare se il ragionamento espresso nel linguaggio naturale sia effettivamente coerente con gli elementi presenti nell'immagine.

Questo aspetto è particolarmente rilevante nei sistemi di interfaccia adattiva, nei quali l'introduzione di meccanismi di Explainable AI consente di aumentare la comprensione del comportamento del sistema e la fiducia dell'utente nelle decisioni automatiche [25].

In questo senso, la generazione della *reason* non rappresenta soltanto un output accessorio, ma costituisce un elemento utile per analizzare il comportamento del modello e per individuare eventuali errori o ambiguità nelle decisioni prodotte.

Nel complesso, il flusso operativo della pipeline può essere sintetizzato come segue: il *ContextBootstrap* inizializza e configura il sistema, il *ContextManager* stabilisce *quando* intervenire sulla base dei segnali di contesto, mentre il *ContextHttpClient* esegue materialmente la fase di inferenza distribuita.

Questa chiara separazione tra configurazione, decisione e azione garantisce maggiore stabilità, manutenibilità e controllo sulle prestazioni, risultando particolarmente adeguata in un sistema distribuito con vincoli di latenza e risorse computazionali limitate come HoloLens 2.

9.3.2 Sensori

I sensori costituiscono il livello percettivo del modulo client e hanno il compito di rilevare variazioni rilevanti nell'ambiente o nella posizione dell'utente, fornendo al **ContextManager** i segnali necessari per decidere l'eventuale riattivazione della pipeline di inferenza.

Nel sistema implementato, come già menzionato precedentemente, sono stati progettati due sensori principali: un sensore di luminosità e un sensore di rilocazione spaziale. Entrambi sono stati sviluppati con l'obiettivo di garantire leggerezza computazionale, stabilità e compatibilità con un'esecuzione continua in tempo reale su HoloLens 2.

Sensore di luminosità

Il sensore di luminosità ha l'obiettivo di rilevare variazioni significative dell'illuminazione percepita dalla camera del visore e di fornire tale informazione al **ContextManager** come possibile segnale di cambiamento di contesto.

L'implementazione è stata progettata per essere computazionalmente leggera, robusta al rumore e compatibile con un'esecuzione continua in tempo reale. A tal fine, la soluzione è stata articolata in due componenti logicamente distinti ma cooperanti:

- un modulo di stima della luminosità istantanea (*brightness provider*)
- un modulo decisionale responsabile della logica di attivazione

Stima della luminosità. La misura della luminosità è affidata a un componente che acquisisce periodicamente un frame a bassa risoluzione dalla PV camera (Photo/Video camera) del visore. Tale camera costituisce il sensore RGB principale del dispositivo, è allineata alla head pose dell'utente ed è progettata per l'acquisizione fotografica e video in modalità egocentrica.

Al fine di evitare un eccessivo carico computazionale, l'acquisizione non avviene a ogni frame di rendering, ma secondo un intervallo di campionamento configurabile (ad esempio $0.25\,s$).

Inoltre, il calcolo non viene effettuato su tutti i pixel dell'immagine: viene applicato un sottocampionamento spaziale tramite uno *stride*, che riduce il numero di pixel analizzati e quindi il costo computazionale. Tale scelta consente di ottenere una stima sufficientemente rappresentativa della luminosità globale senza introdurre overhead significativo.

Per ciascun pixel campionato viene calcolata la luminanza percettiva mediante combinazione lineare dei canali RGB secondo pesi standard:

$$Y = 0.2126R + 0.7152G + 0.0722B \tag{9.1}$$

La media delle luminanze così ottenute viene normalizzata dividendo per 255, producendo un valore scalare:

$$b \in [0,1] \tag{9.2}$$

che rappresenta la luminosità media globale della scena.

Il provider mantiene l'ultimo valore valido e lo rende disponibile tramite un'interfaccia dedicata, consentendo al sensore di accedere in modo non bloccante alla misura corrente. È inoltre prevista la possibilità di sospendere temporaneamente il campionamento durante operazioni più onerose (ad esempio la cattura di un frame per l'invio al server), evitando competizione per l'accesso alla camera.

Logica di rilevamento del cambiamento. Il valore istantaneo di luminosità fornito dal provider viene filtrato mediante una **media mobile esponenziale** (EMA), al fine di attenuare fluttuazioni rapide e rumore dovuto a instabilità dell'esposizione automatica.

Il valore filtrato viene confrontato con una luminosità di riferimento corrispondente all'ultima condizione considerata stabile. Si calcola quindi la variazione assoluta:

$$\Delta = |b_{\text{smoothed}} - b_{\text{reference}}| \tag{9.3}$$

Se Δ supera una soglia prefissata (ad esempio 10%), il cambiamento viene considerato potenzialmente significativo. Per evitare falsi positivi dovuti a transizioni momentanee, la condizione deve permanere per un intervallo minimo di tempo configurabile (ad esempio $0.5\,s$). Solo in presenza di stabilità temporale viene generato il trigger.

Una volta confermato il cambiamento, il sensore:

- imposta il flag di attivazione
- calcola un indicatore di intensità proporzionale a Δ
- aggiorna il valore di riferimento alla nuova luminosità

L'aggiornamento del riferimento impedisce rilevazioni ripetute dello stesso cambiamento e consente al sistema di adattarsi alla nuova condizione luminosa.

Dal punto di vista progettuale, la separazione tra componente di misura e componente decisionale garantisce modularità e flessibilità, permettendo eventuali modifiche alla strategia di stima senza alterare la logica di trigger.

Sensore di rilocazione spaziale

Il sensore di rilocazione spaziale ha l'obiettivo di rilevare quando l'utente effettua uno spostamento rilevante nello spazio tridimensionale tale da modificare in modo sostanziale il contesto percettivo e richiedere una nuova valutazione di visibilità o placement.

A differenza di un semplice controllo sulla distanza istantanea, l'implementazione adotta una strategia a due fasi basata sul principio **Distance + Dwell**, progettata per evitare falsi positivi durante il movimento continuo.

Sorgente dei dati. Il sensore utilizza come input la posizione della testa (*head pose*), ottenuta dal `Transform` della camera principale. La logica viene eseguita con un intervallo di campionamento configurabile, evitando controlli ad ogni frame di rendering.

Fase 1: superamento della soglia di distanza. Il sensore mantiene una posizione di riferimento p_{ref} corrispondente all'ultima posizione stabile. A ogni campionamento viene calcolata la distanza euclidea tra la posizione corrente p e quella di riferimento:

$$d_{\text{ref}} = \|p - p_{\text{ref}}\| \tag{9.4}$$

Se d_{ref} supera una soglia configurabile (ad esempio $2\,m$), il sistema entra in una fase di candidatura: lo spostamento è potenzialmente significativo ma non ancora confermato.

Fase 2: stabilità temporale (dwell). Nella seconda fase, il sensore verifica che l'utente rimanga stabile attorno alla nuova posizione per un tempo minimo prefissato (`dwellSeconds`, ad esempio $2\,s$).

La stabilità è definita come permanenza entro un raggio di tolleranza (`stableRadiusMeters`, ad esempio $0.35\,m$) rispetto alla posizione candidata. Se l'utente continua a muoversi oltre tale raggio, la posizione candidata viene aggiornata e il timer di permanenza viene azzerato.

Il sensore genera il trigger solo quando entrambe le condizioni sono soddisfatte:

1. superamento della soglia di distanza
2. permanenza stabile per il tempo richiesto

Al momento dell'attivazione:

- viene impostato il flag `Triggered`

- viene calcolato uno `Score` normalizzato rappresentativo del progresso della fase di dwell
- la posizione di riferimento viene aggiornata alla nuova posizione stabile

Il sensore non implementa un cooldown interno: la gestione di eventuali intervalli minimi tra due trigger consecutivi è demandata al **ContextManager**.

Considerazioni progettuali. L'adozione della strategia *Distance + Dwell* consente di distinguere tra semplice movimento continuo e reale rilocazione contestuale. Un controllo basato esclusivamente sulla distanza produrrebbe numerosi falsi positivi durante la camminata; l'introduzione del vincolo di stabilità temporale garantisce invece che il trigger venga generato solo quando l'utente si è effettivamente fermato in una nuova area dell'ambiente.

Ne risulta un sensore robusto, parametrizzabile e computazionalmente leggero, adatto a operare in tempo reale senza interferire con le altre componenti del modulo client.

9.4 Esecuzione remota del modello e comunicazione client–server

Uno degli aspetti centrali dell'architettura proposta riguarda la scelta di eseguire il modello fine-tuned della pipeline *UiVLA* in modalità remota, anziché direttamente su HoloLens 2. Tale decisione non è puramente progettuale, ma deriva da vincoli tecnici concreti del dispositivo.

HoloLens 2, pur rappresentando una piattaforma avanzata per applicazioni di Mixed Reality, non supporta nativamente l'esecuzione di codice Python né l'utilizzo di librerie di deep learning quali *PyTorch*. Inoltre, il dispositivo non dispone di una GPU dedicata adeguata per l'inferenza di modelli multimodali di dimensioni significative, come quelli adottati in questo lavoro. L'integrazione diretta della pipeline di intelligenza artificiale all'interno del visore avrebbe quindi comportato limitazioni sostanziali in termini di compatibilità software, capacità computazionale e gestione della memoria.

Per superare tali vincoli, l'architettura è stata progettata secondo un paradigma distribuito. HoloLens 2 mantiene, come descritto nella Sezione 9.3, il controllo della scena, dell'interazione utente e della logica di gestione del contesto, mentre l'elaborazione del modello viene delegata ad un computer locale dotato di GPU, sul quale è possibile eseguire l'intera pipeline di inferenza in ambiente Python. Si ottiene così una netta separazione delle responsabilità: il visore gestisce la componente percettiva e l'interfaccia utente, mentre il nodo remoto fornisce la componente di intelligenza artificiale.

La comunicazione tra HoloLens 2 e il computer remoto avviene tramite rete locale Wi-Fi, adottando un'architettura client–server basata su protocollo HTTP/-REST. Tale scelta privilegia semplicità implementativa, trasparenza del protocollo e facilità di debugging, risultando adeguata per un sistema sperimentale in ambiente controllato.

Dal punto di vista concettuale, il visore assume il ruolo di client attivo: è il *ContextManager* a decidere quando avviare una richiesta di inferenza, sulla base dei trigger generati dai sensori di contesto. Il computer remoto opera invece come nodo computazionale in ascolto di richieste, incaricato di eseguire il modello fine-tuned e restituire il risultato.

Il flusso operativo può essere descritto come una sequenza logica che inizia con l'attivazione di un trigger di contesto. Quando uno o più sensori segnalano un cambiamento significativo e il *ContextManager* ne conferma la rilevanza, viene avviata una nuova fase di inferenza.

Il visore cattura quindi il frame corrente dalla camera RGB. Il contenuto dell'immagine dipende dal task richiesto: nel caso del task di *visibility*, il frame include l'elemento UI già renderizzato nella scena; nel caso del task di *placement*, l'interfaccia non è presente, poiché sarà il modello a suggerire la posizione più adeguata. L'immagine viene compressa in formato JPEG, riducendo il peso del payload e limitando la latenza di trasmissione.

Insieme all'immagine vengono trasmessi metadati essenziali, quali l'identificativo del task e un timestamp, strutturati all'interno di una richiesta HTTP `POST` in formato `multipart/form-data`. Questa soluzione consente di inviare direttamente il file binario dell'immagine, evitando codifiche meno efficienti come l'inclusione in *base64* all'interno di oggetti JSON.

Una volta ricevuta la richiesta, il server decodifica l'immagine, esegue le operazioni di preprocessing necessarie e avvia l'inferenza del modello sulla GPU. L'output dipende dal tipo di task richiesto. Nel caso della valutazione di visibilità, il modello restituisce una label binaria accompagnata da una breve motivazione testuale. Nel caso del placement, viene individuata la regione candidata più adeguata e ne viene restituito l'identificativo insieme alle coordinate di posizionamento.

La risposta viene inviata al visore in formato JSON. HoloLens interpreta l'esito dell'inferenza e aggiorna dinamicamente l'interfaccia utente, modificandone la posizione o lo stato visivo in funzione del risultato ottenuto. L'intero processo è progettato per operare in tempi compatibili con l'interazione in realtà mista, preservando la percezione di continuità e reattività dell'esperienza utente.

In sintesi, l'esecuzione remota del modello e l'adozione di un'architettura client–server consentono di superare i limiti hardware del visore, mantenendo al contempo una struttura modulare, scalabile e coerente con i requisiti computazionali della pipeline *UiVLA*.

9.5 Architettura del server

Il modulo server costituisce il nodo computazionale della pipeline *UiVLA* ed è responsabile dell'esecuzione del modello multimodale fine-tuned e della gestione delle richieste di inferenza provenienti dal visore. La sua progettazione è orientata a garantire efficienza, modularità e riduzione della latenza, mantenendo al contempo un'interfaccia semplice e ben definita verso il client.

9.5.1 Infrastruttura e inizializzazione

Il server locale è stato implementato utilizzando **FastAPI**, un framework Python moderno per la costruzione di API REST ad alte prestazioni. La scelta di questo framework è stata motivata dalla necessità di ridurre la latenza complessiva del sistema e di garantire una gestione efficiente delle richieste.

Un aspetto centrale dell'architettura riguarda la fase di inizializzazione dei modelli al momento dell'avvio del server. Tramite il meccanismo `@app.on_event("startup")`, viene eseguita una routine di **warmup** che carica in memoria i modelli principali della pipeline (Qwen2.5-VL e Mask2Former), inizializza il dispositivo di esecuzione (CPU o CUDA) e forza una prima inferenza su un'immagine fittizia.

Questa operazione svolge un ruolo cruciale dal punto di vista prestazionale. Essa consente di inizializzare il contesto CUDA, allocare la memoria GPU, caricare i pesi dalla cache locale e materializzare eventuali shard del checkpoint prima che arrivino richieste reali. In assenza di tale fase, la prima chiamata HTTP verso gli endpoint `/visibility` o `/placement` potrebbe subire un ritardo significativo dovuto al caricamento dei pesi o alla compilazione iniziale dei kernel.

L'operazione di warmup viene eseguita una sola volta all'avvio del server, garantendo che le richieste successive possano sfruttare modelli già residenti in memoria e pronti per l'inferenza, riducendo sensibilmente la latenza delle elaborazioni.

9.5.2 Interfaccia API ed endpoint

Dal punto di vista dell'interfaccia di comunicazione, il server espone un insieme essenziale di endpoint progettati per rispondere in modo mirato alle esigenze della pipeline *UiVLA*. L'obiettivo è mantenere un'architettura chiara e facilmente manutenibile, evitando interfacce generiche o ambigue.

L'endpoint `/ping`, di tipo `GET`, svolge una funzione diagnostica. Esso consente di verificare che il server sia in esecuzione e correttamente raggiungibile dalla rete locale. La risposta restituisce un semplice flag booleano insieme a un timestamp in formato ISO (UTC), utile per controllare eventuali problemi di sincronizzazione o per effettuare test preliminari di latenza tra visore e PC.

Gli endpoint centrali della pipeline sono `/visibility` e `/placement`, entrambi di tipo `POST`. In entrambi i casi, il server riceve una richiesta strutturata in formato `multipart/form-data`, composta da tre elementi principali: un `task_id`, un `timestamp` e un file `image`. Tale scelta consente di trasmettere direttamente il frame JPEG come flusso di byte, evitando codifiche meno efficienti come l'inclusione dell'immagine in formato *base64* all'interno di un oggetto JSON.

Nel caso dell'endpoint `/visibility`, il frame inviato dal visore contiene già l'elemento UI sovrapposto alla scena reale. Il server legge i byte dell'immagine, la ricostruisce tramite la libreria PIL e la converte esplicitamente in formato RGB per garantire coerenza nell'elaborazione. Successivamente viene invocata la funzione di inferenza, alla quale vengono passati l'immagine e l'identificativo del task.

Il risultato consiste in una label binaria (ad esempio 0 o 1) che indica l'esito della valutazione di visibilità, accompagnata da una breve motivazione testuale. La risposta JSON include inoltre il `task_id` e una misura della durata della richiesta (`duration_request`), calcolata confrontando il timestamp ricevuto dal client con l'orario corrente del server. Tale valore consente di monitorare le prestazioni del sistema e di analizzare la latenza complessiva della pipeline.

L'endpoint `/placement` segue una logica analoga, ma con una differenza concettuale rilevante: l'immagine ricevuta non contiene l'interfaccia utente, poiché l'obiettivo è determinare la posizione ottimale in cui inserirla. Dopo la ricostruzione e il preprocessing dell'immagine, viene eseguita la funzione di inferenza che restituisce l'identificativo della regione selezionata e, quando disponibili, le coordinate corrispondenti della regione scelta, memorizzate dopo il preprocessing.

Tali coordinate vengono convertite in valori floating-point e restituite come oggetto JSON con campi `x` e `y`, oppure come valore nullo qualora non venga individuata una posizione valida. Anche in questo caso viene calcolata la durata della richiesta, mantenendo uniformità strutturale tra i due task.

Nel complesso, l'organizzazione degli endpoint riflette una chiara separazione tra infrastruttura computazionale e interfaccia di comunicazione. La distinzione tra fase di inizializzazione e gestione delle richieste operative consente di isolare la componente di inferenza dalla logica di trasporto dei dati, migliorando la leggibilità del codice, facilitando l'estensione futura della pipeline e rendendo più semplice il testing indipendente dei singoli task.

9.6 Considerazioni conclusive

Il capitolo ha descritto l'integrazione della pipeline *UiVLA* su HoloLens 2 secondo un'architettura distribuita client–server. La scelta di separare il nodo di inferenza dal visore consente di superare i vincoli hardware e software del dispositivo,

mantenendo sul client la gestione del contesto e l'aggiornamento dell'interfaccia, e delegando al server l'esecuzione del modello fine-tuned.

L'adozione di sensori leggeri e di una logica di attivazione event-driven permette inoltre di limitare le richieste di inferenza ai soli momenti in cui il contesto percettivo varia in modo significativo, riducendo computazione e traffico di rete e migliorando la stabilità del sistema in scenari reali.

Nel capitolo successivo vengono analizzate le prestazioni complessive del sistema integrato, considerando sia metriche computazionali (latenza end-to-end, tempi di inferenza) sia aspetti qualitativi legati all'esperienza d'uso (coerenza delle decisioni di *placement* e *visibility* nel ciclo operativo).

Capitolo 10

Risultati

Questo capitolo riporta i risultati ottenuti dopo l'integrazione della pipeline *UiVLA* su HoloLens 2, con particolare attenzione agli aspetti che è stato possibile misurare e documentare nel prototipo sviluppato. In questa fase non è stata condotta un'analisi completa dei tempi end-to-end (ad esempio decomponendo separatamente cattura del frame, compressione, trasmissione e rendering), ma è stata registrata la metrica di latenza più rilevante dal punto di vista operativo del sistema distribuito: il tempo che intercorre tra l'invio della richiesta HTTP dal visore e la ricezione della risposta dal server (*request–response time*).

Oltre alla componente computazionale, il capitolo presenta una valutazione qualitativa del comportamento del modello sul task di *visibility*, focalizzata sulla capacità di produrre spiegazioni testuali coerenti con la decisione binaria (*good*/*bad placement*). Tale analisi viene supportata da esempi visuali che mostrano scene reali con overlay UI e le relative risposte generate dal modello, includendo sia casi di posizionamento accettabile sia casi problematici.

L'analisi dei risultati si articola in due componenti principali: (i) la valutazione della latenza della pipeline di inferenza remota e (ii) un'analisi qualitativa del comportamento del modello nei task di *visibility* e *placement*, che include sia l'esame delle *reason* generate nel task di *visibility* sia l'osservazione delle decisioni di posizionamento prodotte nel task di *placement*.

10.1 Valutazione della latenza di inferenza nella pipeline client–server

In questa sezione viene analizzato il tempo di risposta della pipeline distribuita, misurato come intervallo tra l'istante di invio della richiesta HTTP dal client (HoloLens 2) e l'istante di ricezione della risposta dal server. Tale misura include il tempo di trasmissione in rete locale e il tempo di inferenza sul server, e rappresenta

quindi una stima diretta della latenza percepita dal sistema per l'ottenimento della decisione.

10.1.1 Definizione della metrica e modalità di misura

Per valutare la fattibilità operativa della pipeline UiVLA integrata su HoloLens 2 è stata considerata una metrica di latenza legata alla comunicazione client–server e all'esecuzione dell'inferenza sul nodo remoto. In particolare, è stato misurato il tempo necessario affinché una richiesta inviata dal visore venga elaborata dal server e restituisca una risposta contenente la decisione del modello.

La metrica adottata, denominata `duration_request`, viene calcolata direttamente sul server come differenza tra il timestamp associato alla richiesta ricevuta dal client e l'istante corrente al termine dell'inferenza. Il timestamp viene inviato dal visore come parte dei metadati della richiesta HTTP e rappresenta il momento di invio della richiesta stessa. Al termine dell'elaborazione, il server calcola la differenza temporale e restituisce il risultato all'interno della risposta JSON.

Formalmente, la metrica può essere espressa come:

$$duration\_request = t_{response} - t_{request} \tag{10.1}$$

dove $t_{request}$ rappresenta il timestamp associato alla richiesta inviata dal client e $t_{response}$ l'istante in cui il server completa l'elaborazione e genera la risposta.

Questa misura include sia il tempo necessario all'inferenza del modello sia il tempo di comunicazione tra client e server sulla rete locale. Di conseguenza, essa rappresenta una stima della latenza percepita dal sistema per ottenere la decisione del modello all'interno della pipeline distribuita.

L'obiettivo principale di questa misura non è fornire una caratterizzazione completa delle singole componenti temporali della pipeline, ma verificare la coerenza tra le prestazioni osservate nel sistema reale e i risultati ottenuti durante la fase di fine-tuning del modello. In particolare, i valori di `duration_request` permettono di confrontare empiricamente i tempi di risposta della pipeline con le velocità di inferenza (espresse in *samples/s*) riportate nella Tabella 8.1, ottenute durante l'addestramento multi-task.

Il calcolo della metrica viene effettuato direttamente all'interno degli endpoint del server. A titolo esemplificativo, il Listato 10.1 mostra la porzione di codice utilizzata nell'endpoint `/visibility`, in cui la differenza tra i timestamp viene calcolata e restituita come campo della risposta JSON.

```
1 duration = datetime.now(timezone.utc) - datetime.fromisoformat(
2     timestamp.replace("Z", "+00:00")
3 )
4
5 return {
```

```
    "ok": True,
    "task_id": task_id,
    "label": label,
    "reason": reason,
    "duration_request": duration.total_seconds()
}
```

Listing 10.1: Calcolo della metrica `duration_request` nell'endpoint /visibility

Questo approccio consente di monitorare in modo semplice e diretto i tempi di risposta della pipeline durante l'esecuzione del sistema in scenari reali, verificando che l'inferenza del modello rimanga compatibile con i vincoli temporali richiesti dall'applicazione di Mixed Reality.

10.1.2 Risultati osservati

Utilizzando il modello fine-tuned con configurazione LoRA rank pari a 8 e un numero di patch pari a 512, come riportato nella Tabella 8.1, i test condotti all'interno della pipeline integrata su HoloLens 2 hanno evidenziato una latenza di risposta compresa tra circa 380 ms e 500 ms per singola richiesta.

La variabilità osservata tra le diverse richieste è riconducibile a diversi fattori. In primo luogo, una parte della variazione dipende dalle condizioni della rete Wi-Fi utilizzata per la comunicazione tra il visore e il server remoto, che introduce una componente di latenza variabile legata alla trasmissione dei dati.

Un secondo fattore rilevante è rappresentato dal tipo di task eseguito dal modello. Nel caso del task di *visibility*, il tempo di inferenza tende ad essere maggiore rispetto al task di *placement*. Ciò è dovuto principalmente alla struttura del prompt utilizzato: il prompt associato al task di *visibility* è significativamente più lungo e richiede quindi al modello di elaborare un numero maggiore di token. Inoltre, il modello non restituisce soltanto una label binaria, ma genera anche una spiegazione testuale (*reason*). Poiché il modello opera secondo un paradigma autoregressivo, la generazione della risposta avviene token per token, contribuendo ad aumentare il tempo complessivo di inferenza.

Per quanto riguarda il task di *placement*, la latenza dipende invece in parte dalla complessità semantica della scena analizzata. Prima di essere fornito al modello Vision–Language, il frame acquisito dal visore viene infatti elaborato da *Mask2Former* al fine di individuare le regioni candidate rilevanti. Nel caso di scene contenenti numerosi oggetti o superfici potenzialmente idonee al posizionamento dell'interfaccia, il numero di regioni generate aumenta e il tempo di preprocessing cresce di conseguenza. Nei test effettuati, questa fase introduce un overhead addizionale dell'ordine di 20–30 ms.

Nel complesso, i risultati osservati indicano che la pipeline di inferenza remota mantiene tempi di risposta nell'ordine di alcune centinaia di millisecondi, confermando la coerenza tra le prestazioni ottenute durante la fase di fine-tuning e il comportamento del modello quando integrato in uno scenario applicativo reale.

10.1.3 Discussione e implicazioni

I risultati osservati indicano che la latenza della pipeline di inferenza remota si colloca nell'ordine di alcune centinaia di millisecondi. In un sistema interattivo di Mixed Reality, tale intervallo temporale deve essere interpretato alla luce della modalità operativa della pipeline UiVLA.

A differenza di molte applicazioni di visione artificiale che richiedono inferenza continua a ogni frame, il sistema proposto adotta un paradigma *event-driven*. L'inferenza del modello non viene eseguita in modo continuo, ma viene attivata esclusivamente quando i sensori di contesto rilevano un cambiamento significativo nella scena. Di conseguenza, la latenza osservata non influisce sul rendering dell'interfaccia a livello di frame-rate, ma solo sul tempo necessario per ottenere una nuova decisione del modello quando il contesto cambia.

Alla luce di questa architettura, una latenza compresa tra $380\,ms$ e $500\,ms$ risulta compatibile con il funzionamento della pipeline, consentendo al sistema di aggiornare l'interfaccia utente in tempi adeguati rispetto alla frequenza con cui si verificano variazioni di contesto nell'ambiente reale.

È inoltre importante sottolineare che il raggiungimento di tali tempi di risposta è strettamente legato alla disponibilità di hardware adeguato per l'esecuzione del modello. In particolare, il backbone multimodale *Qwen2.5-VL* è stato eseguito utilizzando GPU che supportano il formato numerico *bfloat16*. L'utilizzo di tale formato consente di sfruttare in modo efficiente le unità di calcolo dedicate delle GPU moderne, riducendo significativamente il tempo di inferenza. In assenza del supporto a *bfloat16*, l'esecuzione del modello su hardware meno recente o non ottimizzato comporta un aumento drastico dei tempi di inferenza, che nei test preliminari può raggiungere anche l'ordine delle decine di secondi (fino a circa $40\,s$ per richiesta), rendendo di fatto impraticabile l'utilizzo del sistema in scenari interattivi.

È inoltre importante osservare che la metrica utilizzata fornisce una stima aggregata della latenza della pipeline distribuita. Il valore misurato include infatti sia il tempo di inferenza del modello sia la latenza di comunicazione sulla rete locale, ma non distingue in modo esplicito le singole componenti temporali (cattura dell'immagine, compressione, trasmissione, preprocessing e inferenza). Un'analisi più dettagliata di tali contributi costituirebbe una possibile estensione futura del lavoro.

Nel complesso, i risultati suggeriscono che l'integrazione del modello fine-tuned all'interno della pipeline client–server su HoloLens 2 è tecnicamente fattibile e compatibile con i requisiti temporali dell'applicazione considerata.

10.2 Valutazione qualitativa dei risultati del modello

Oltre all'analisi quantitativa della latenza della pipeline di inferenza, è stata condotta una valutazione qualitativa del comportamento del modello nei task di *visibility* e *placement*. L'obiettivo di questa analisi è verificare se le decisioni prodotte dal modello risultino coerenti con il contesto visivo della scena e con le azioni che il sistema deve eseguire all'interno della pipeline UiVLA.

In particolare, nel task di *visibility* viene analizzata la capacità del modello di fornire spiegazioni testuali coerenti con la decisione binaria restituita, mentre per il task di *placement* vengono discusse le modalità con cui è possibile valutare qualitativamente le decisioni di posizionamento dell'interfaccia all'interno dell'ambiente reale.

10.2.1 Valutazione qualitativa delle spiegazioni nel task di visibility

Questa sottosezione analizza qualitativamente le spiegazioni generate dal modello nel task di *visibility*. In particolare, l'obiettivo è verificare se il modello sia in grado non solo di produrre una decisione binaria (*good placement* o *bad placement*), ma anche di fornire una motivazione testuale coerente con il contesto visivo della scena.

Nel sistema UiVLA, la spiegazione generata dal modello svolge un ruolo importante dal punto di vista interpretativo. Essa consente infatti di comprendere le ragioni alla base della decisione presa dal modello e permette di verificare se il ragionamento espresso nel linguaggio naturale sia effettivamente coerente con gli elementi presenti nell'immagine. In questo senso, la generazione della *reason* non rappresenta soltanto un output accessorio, ma costituisce un meccanismo utile per analizzare il comportamento del modello e per identificare eventuali errori o ambiguità nelle decisioni prodotte.

L'analisi è stata condotta osservando direttamente le risposte generate dal modello in diversi scenari reali acquisiti tramite il visore HoloLens 2. In particolare, sono stati considerati sia casi in cui l'interfaccia risulta correttamente posizionata (*good placement*) sia situazioni in cui il modello identifica problemi di visibilità o di interferenza con elementi della scena (*bad placement*).

Nel sistema implementato, la spiegazione generata dal modello viene visualizzata direttamente all'interno dell'interfaccia del visore. In particolare, la *reason* viene

mostrata nella parte inferiore del campo visivo dell'utente, in una posizione progettata per non interferire con l'osservazione della scena principale e per evitare che l'utente debba abbassare significativamente lo sguardo per leggerla. La spiegazione rimane visibile per un intervallo temporale di circa 5 secondi, risultando sufficiente per consentire la lettura del messaggio senza introdurre elementi persistenti che possano distrarre dall'interazione con l'ambiente.

Esempi di good placement

La Figura 10.1 mostra due esempi di scene in cui il modello classifica correttamente il posizionamento dell'interfaccia come *good placement*. In questo caso, l'elemento UI è collocato in una regione libera della scena, senza sovrapporsi a oggetti rilevanti o interferire con la percezione dell'ambiente circostante.

La spiegazione generata dal modello evidenzia come la posizione scelta garantisca una buona leggibilità dell'interfaccia e non introduca conflitti visivi con gli oggetti presenti nella scena. Questo comportamento indica che il modello è in grado di associare la decisione binaria a una motivazione coerente con il contesto visivo.

Esempi di bad placement

La Figura 10.2 mostra invece due casi in cui il modello classifica il posizionamento come *bad placement*. In queste situazioni, l'interfaccia risulta sovrapposta a elementi della scena o collocata in una posizione che ne riduce la leggibilità.

Anche in questo caso, la spiegazione generata dal modello identifica correttamente le cause del problema, facendo riferimento alla presenza di oggetti o superfici che interferiscono con la visibilità dell'interfaccia. Questo suggerisce che il modello non si limita a produrre una classificazione binaria, ma è in grado di esprimere un ragionamento che collega la decisione agli elementi visivi presenti nell'immagine.

Osservazioni

L'analisi qualitativa delle spiegazioni mostra che il modello è generalmente in grado di produrre motivazioni coerenti con la scena osservata e con la decisione binaria restituita. In molti casi, le spiegazioni fanno riferimento a fattori rilevanti per la visibilità dell'interfaccia, come la presenza di oggetti nella scena, la possibile occlusione dell'elemento UI o la riduzione della leggibilità.

Questi risultati suggeriscono che il modello fine-tuned non apprende soltanto una mappatura diretta tra immagine e label, ma sviluppa anche una rappresentazione semantica della scena che gli consente di giustificare le proprie decisioni in linguaggio naturale. Tale comportamento è particolarmente rilevante nel contesto della pipeline UiVLA, in cui la decisione del modello determina direttamente un'azione sullo stato dell'interfaccia.

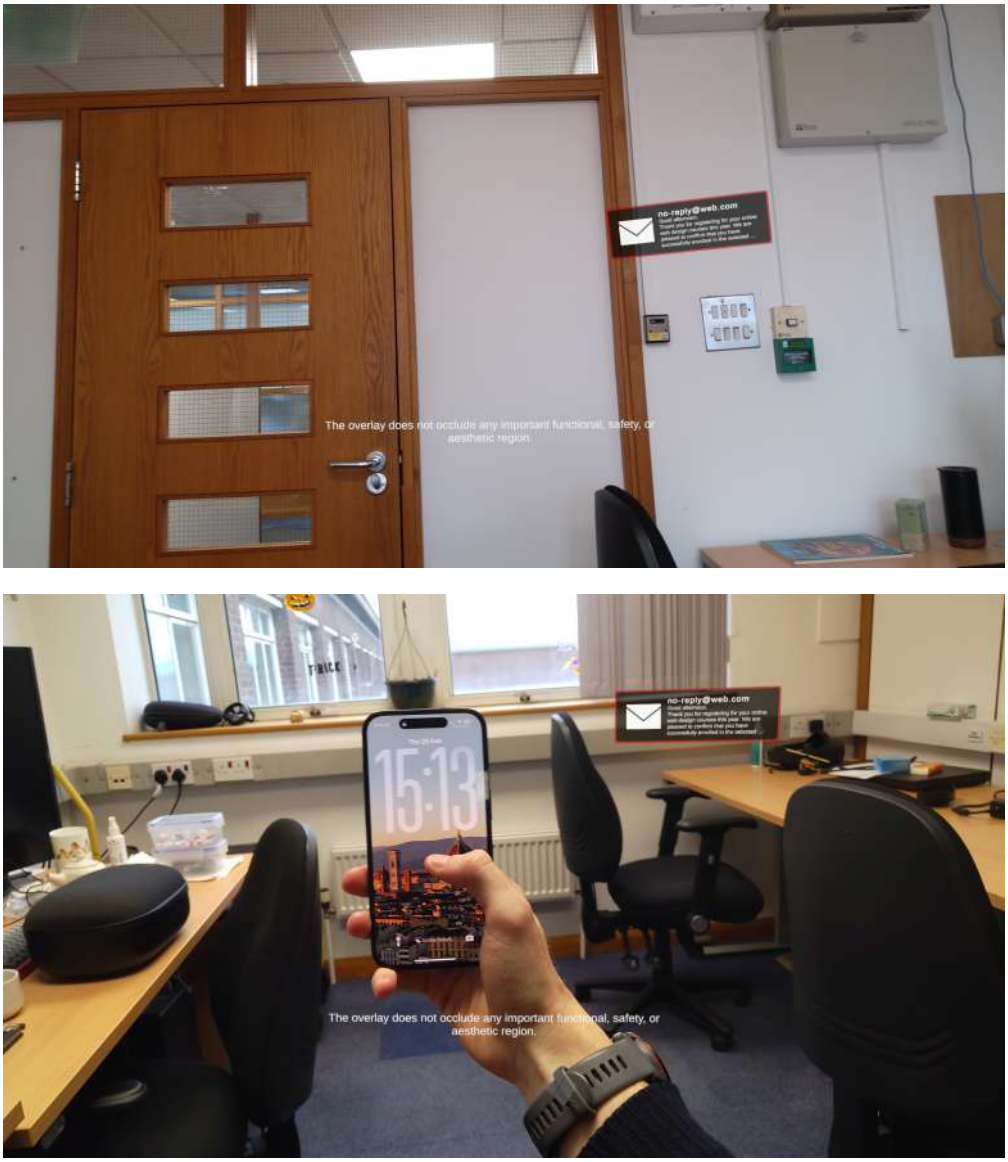


Figura 10.1: Esempi di *good placement*. Il modello valuta positivamente la posizione dell'interfaccia e fornisce spiegazioni coerenti con il contesto della scena.

Nel complesso, le osservazioni qualitative indicano che il modello è in grado di fornire spiegazioni plausibili e coerenti con il contesto visivo, contribuendo a rendere il comportamento del sistema più interpretabile e trasparente.

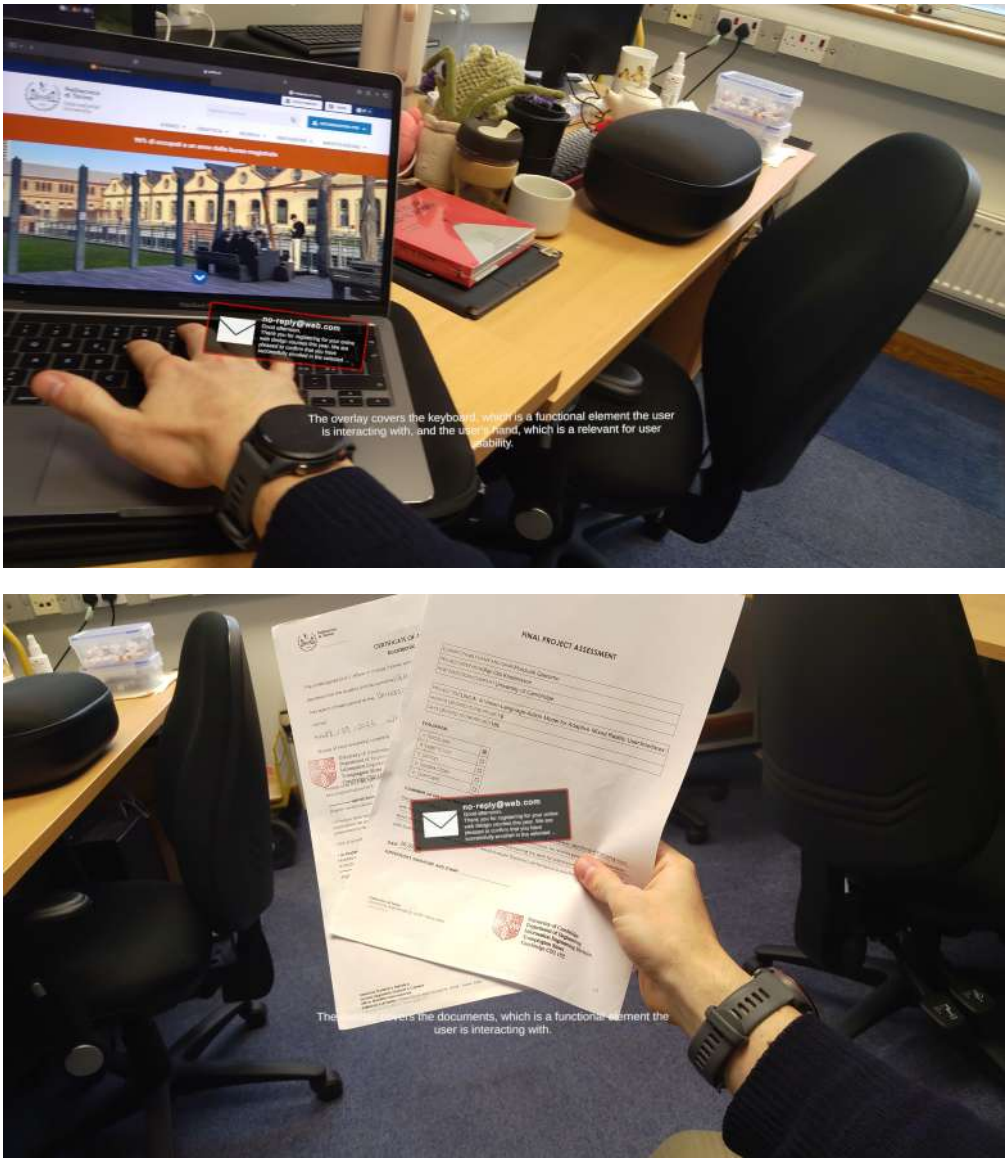


Figura 10.2: Esempi di *bad placement*. Il modello valuta negativamete la posizione dell'interfaccia e fornisce spiegazioni coerenti con il contesto della scena.

10.2.2 Limitazioni della valutazione qualitativa del task di placement

A differenza del task di *visibility*, la valutazione qualitativa del task di *placement* presenta alcune difficoltà metodologiche legate alla natura stessa del problema. Nel task di *visibility*, infatti, è possibile analizzare il comportamento del modello attraverso singoli frame della scena, verificando se la decisione binaria e la spiegazione

testuale risultino coerenti con il contesto visivo osservato.

Nel caso del task di *placement*, invece, il modello produce una decisione che si traduce direttamente in un'azione nello spazio tridimensionale, ossia nel posizionamento dell'interfaccia all'interno dell'ambiente reale. La qualità di tale decisione dipende non solo dal contenuto del singolo frame utilizzato come input, ma anche dalla relazione spaziale tra l'interfaccia, gli oggetti presenti nella scena e la prospettiva dell'utente.

Per questo motivo, una valutazione basata esclusivamente su immagini statiche risulta limitata. Un singolo frame non consente infatti di cogliere completamente l'effetto del posizionamento dell'interfaccia durante l'interazione con l'ambiente, né di valutare come tale posizione venga percepita dall'utente durante il movimento o il cambiamento di prospettiva.

Di conseguenza, la valutazione qualitativa del task di *placement* risulta più appropriata attraverso l'osservazione del comportamento del sistema nel tempo. A tal fine, è stata effettuata una registrazione video direttamente dalla camera di HoloLens 2 utilizzando il tool *Windows Device Portal*, che consente di acquisire il flusso video del visore durante l'esecuzione dell'applicazione. La registrazione, della durata di circa un minuto, mostra il funzionamento del sistema in un contesto reale e permette di osservare il comportamento della pipeline UiVLA durante il posizionamento dinamico dell'interfaccia nell'ambiente.

Il video è disponibile su YouTube al seguente link: `https://youtu.be/DU4-zvyQoYs`

Capitolo 11

Conclusioni

Questa tesi ha investigato la fattibilità del paradigma *vision–language–action* per l'adattamento delle interfacce utente in Mixed Reality, proponendo *UiVLA* come pipeline end-to-end in grado di trasformare una singola osservazione visiva della scena in decisioni operative sull'interfaccia. L'obiettivo non era ottimizzare una funzione di costo definita a priori, né costruire un sistema basato su regole ed euristiche progettate manualmente, ma verificare se un modello multimodale potesse apprendere direttamente una mappatura *osservazione* $\rightarrow$ *azione* attraverso fine-tuning supervisionato su task specifici.

Nel lavoro, l'adattamento dell'interfaccia è stato formalizzato attraverso due compiti complementari: il task di *placement*, che seleziona una posizione candidata per l'elemento UI, e il task di *visibility*, che valuta la qualità percettiva del posizionamento corrente e determina se mantenere o correggere lo stato dell'interfaccia. Questa formulazione rende esplicito il passaggio da Vision–Language Model (VLM) a Vision–Language–Action (VLA): l'output del modello non è un testo descrittivo fine a sé stesso, ma un'istruzione simbolica che modifica lo stato del sistema.

Dal punto di vista metodologico, la necessità di specializzare un backbone Vision–Language generico ha motivato la costruzione di un dataset dedicato e la definizione di un processo di addestramento progressivo. In particolare, prima di esplorare l'addestramento congiunto, è stata verificata la specializzazione su ciascun task in regime di *single-task learning*, riducendo il rischio di investire risorse computazionali in un regime multi-task senza una validazione preliminare della fattibilità. Successivamente, gli esperimenti di *multi-task learning* hanno mostrato che le due competenze possono essere integrate in un unico backbone condiviso, coerentemente con la natura iterativa della pipeline UiVLA.

Un contributo centrale di questa tesi riguarda l'integrazione del modello all'interno di un sistema reale basato su *HoloLens 2*. L'architettura implementata adotta una pipeline distribuita client–server, nella quale il visore gestisce l'acquisizione del contesto e l'aggiornamento dell'interfaccia, mentre l'inferenza del modello viene

delegata a un nodo remoto dotato di GPU. I risultati osservati in esecuzione reale sono risultati coerenti con le prestazioni emerse in fase di fine-tuning: la latenza end-to-end della pipeline si colloca tipicamente tra circa $380\,ms$ e $500\,ms$, confermando la fattibilità di un ciclo decisionale near real-time in un contesto interattivo. Tale risultato è particolarmente rilevante poiché la pipeline è *condizionale* (attivata da trigger contestuali) e non basata su inferenza continua su ogni frame, riducendo inferenze ridondanti in condizioni di scena stabile.

Va tuttavia evidenziato che il raggiungimento di tali tempi di risposta dipende in modo critico dall'infrastruttura hardware del nodo remoto: per mantenere latenze nell'ordine di centinaia di millisecondi è necessario eseguire l'inferenza su GPU che supportino in modo efficiente formati a precisione ridotta (ad esempio `bfloat16`) per il modello Qwen2.5-VL. In assenza di tale supporto, i tempi di inferenza possono degradare drasticamente fino a valori incompatibili con l'interazione, arrivando all'ordine di decine di secondi.

Accanto ai risultati positivi, l'analisi del sistema ha evidenziato alcune limitazioni. In primo luogo, le spiegazioni testuali (*reason*) generate nel task di *visibility* risultano talvolta poco specifiche o non perfettamente allineate ai dettagli della scena. Questo comportamento è plausibilmente legato alla qualità delle annotazioni presenti nel dataset supervisionato utilizzato per il fine-tuning. Poiché il modello apprende la struttura delle spiegazioni direttamente dalle reason fornite nel dataset, eventuali imprecisioni o semplificazioni nelle annotazioni vengono inevitabilmente riflesse nelle spiegazioni generate dal modello.

In secondo luogo, la qualità delle decisioni dipende anche dalla disponibilità di segnali semantici prodotti da moduli di percezione esterni. Ad esempio, l'uso di detector open-vocabulary come YOLO-World introduce vincoli legati al tipo di classi riconoscibili: in scenari con persone, il sistema tende a rilevare la classe *person* senza distinguere parti del corpo, e non è in grado di discriminare se mani o arti appartengano all'utente che indossa il visore o ad altri soggetti presenti nella scena.

Queste limitazioni riducono la granularità della rappresentazione semantica dell'ambiente e possono influenzare la qualità delle motivazioni generate dal modello o delle decisioni di adattamento in scenari complessi.

Nonostante tali limitazioni, i risultati ottenuti indicano che il paradigma Vision–Language–Action rappresenta una direzione promettente per la Mixed Reality. Rispetto agli approcci basati su ottimizzazione esplicita, UiVLA sposta l'attenzione da funzioni obiettivo rigide e vincoli progettati manualmente a una mappatura appresa dai dati, potenzialmente più flessibile rispetto alla variabilità del contesto reale. Rispetto a pipeline che combinano VLM e moduli rigidi non addestrabili congiuntamente, UiVLA mostra come sia possibile centralizzare la decisione in un backbone multimodale specializzato sui task, mantenendo una formulazione unificata e coerente con un ciclo percezione–decisione–azione.

A partire dal lavoro presentato, diversi sviluppi futuri risultano naturali. Un primo passo consiste nel miglioramento del dataset, introducendo controlli di qualità più stringenti sulle annotazioni e aumentando la copertura di scenari complessi e out-of-distribution, al fine di ottenere spiegazioni più affidabili e decisioni più robuste. Un secondo sviluppo riguarda l'integrazione di moduli di percezione più informativi, in grado di fornire segnali più granulari sull'utente e sull'ambiente (ad esempio segmentazione di parti del corpo o riconoscimento di mani egocentriche). Infine, un'estensione rilevante consisterebbe in valutazioni empiriche con utenti per misurare l'impatto dell'adattamento automatico su aspetti quali carico cognitivo, usabilità e accettabilità, completando la validazione ingegneristica con una validazione HCI.

Nel complesso, questa tesi fornisce una dimostrazione concreta della fattibilità di una pipeline end-to-end per l'adattamento della UI in Mixed Reality guidata da un modello multimodale fine-tuned, mostrando coerenza tra risultati sperimentali e comportamento del sistema in scenari reali, e delineando una base solida per sviluppi futuri verso sistemi più robusti, spiegabili e generalizzabili.

Appendice A

Prompt utilizzati negli esperimenti

In questa appendice sono riportati i prompt testuali utilizzati per il fine-tuning e l'inferenza del modello nei task di *visibility* e *placement*. I prompt sono presentati nella loro forma completa al fine di garantire la riproducibilità degli esperimenti.

A.1 Prompt per il task di visibility

A.1.1 Fine-tuning con supervisione sul solo token binario

Visibility Prompt

Task: You see **two images** from a head-mounted camera:

- **Image 1**: the original camera view.
- **Image 2**: the same view with a UI overlay.

Decide whether the UI overlay in **Image 2** covers any safety-relevant or visually important content.

Definition of "covers" (use only this):

- The overlay **covers** something if any part of the object or region **behind the overlay is occluded** (not visible).
- Ignore everything printed **on the overlay itself**, such as icons, logos, text, or borders.
- Decide **only based on what is behind the overlay**.

Do not treat camera or privacy effects as coverage:

- fish-eye distortion, motion blur, depth-of-field blur, defocus
- noise or compression artifacts
- privacy blur, pixelation, or mosaic applied to faces or bodies

These effects **alone** do not count as covered. Coverage requires occlusion **only by the overlay**.

Conditions for answer 0: Answer **0** if the overlay covers, even partially, **any** of the following:

- hands, phone, tools, screens, buttons, or labels to read (used or about to be used)
- floor, doors, windows, or signs
- any part of a person (face or body)
- a visually salient region (high edge contrast or vivid color)

Outdoor notes: Still answer **0** if the overlay covers:

- crosswalks, curbs, or lane markings
- traffic lights or traffic signs
- approaching vehicles or bikes

Border cases:

- If an object is near or adjacent to the overlay, it is **not covered**; it must be **behind** the overlay and occluded.
- A transparent or semi-transparent overlay **still counts as coverage** if it hides edges or textures behind it.
- If only uniform or empty background is under the overlay, answer **1**.
- When uncertain about overlap for safety-relevant content, prefer answer **0**.

OUTPUT FORMAT:
Reply with **exactly one character**: `0` or `1`.
No words, no explanation, no punctuation, no spaces, no newline.

A.1.2 Fine-tuning con formato standard di output

Visibility Prompt

Task: You see **two images** from a head-mounted camera:

- **Image 1**: the original camera view.
- **Image 2**: the same view with a UI overlay.

Goal: Decide whether the UI overlay in **Image 2** covers any **safety-relevant or visually important content**.

Definition of "covers" (use only this):

- The overlay **covers** something if any part of the object or region **behind the overlay is occluded** (not visible).
- Ignore everything printed **on the overlay itself** (icons, logos, text, borders).
- Decide **only based on what is behind the overlay**.

Do not treat camera or privacy effects as coverage:

- fish-eye distortion, motion blur, depth-of-field blur, defocus
- noise or compression artifacts
- privacy blur, pixelation, or mosaic applied to faces or bodies

These effects **alone** do not count as coverage. Coverage requires occlusion **only by the overlay**.

Answer 0 if the overlay covers (even partly) any of the following:

- hands, phone, tools, screens, buttons, or readable labels (used or about to be used)
- floor, doors, windows, or signs
- any part of a person (face or body)
- a visually salient region (high edge contrast or vivid color)

Outdoor notes (still answer 0 if covered):

- crosswalks, curbs, or lane markings
- traffic lights or traffic signs

- approaching vehicles or bikes

Border cases:

- If an object is **near or adjacent** to the overlay, it is **not covered**; it must be **behind** the overlay and occluded.
- A transparent or semi-transparent overlay **still counts as coverage** if it hides edges or textures behind it.
- If only uniform or empty background is under the overlay, answer **1**.
- When uncertain about overlap for safety-relevant content, prefer answer **0**.

Explanation Rules:

- Begin the explanation with:
  - `The overlay is not shown because` (for label 0)
  - `The overlay is shown because` (for label 1)
- Mention the specific object or region behind the overlay.

OUTPUT FORMAT (MANDATORY):

- First character must be exactly: `0` or `1`
- Then one space
- Then one explanation sentence
- No prefixes such as "Answer:" and no extra spaces or newlines

Examples:

```
0 The overlay is not shown because it covers the
user's hands on the keyboard.
1 The overlay is shown because it covers only a
uniform empty wall.
0 The overlay is not shown because it occludes
part of a person's arm.
1 The overlay is shown because it covers only
non-salient background.
```

A.1.3 Fine-tuning con label e reason

Visibility Prompt

Task: You see **one image** from a head-mounted camera with a UI overlay applied. Decide whether the UI overlay should be **shown** or **not shown**.

Decision Rule:

- Output **0** if the overlay covers any important or safety-relevant content.
- Output **1** if the overlay does not occlude any relevant region.

Definition of "covers":

- The overlay **covers** something if any part of the object or region **behind the overlay is occluded**.
- Ignore icons, text, or graphics printed **on the overlay itself**.
- Decide **only based on what is behind the overlay**.

Do **not** treat the following as coverage:

- fish-eye distortion, motion blur, defocus, depth-of-field blur
- image noise or compression artifacts
- privacy blur, pixelation or mosaic applied to faces or bodies

Conditions for answer 0 (overlay should NOT be shown):
A) Functionality

- The overlay covers objects the user is interacting with or about to use.
- Examples: hands, phone, tools, screens, buttons, bottles, readable labels.
- Functionality always counts, regardless of distance.

B) Safety and Social Acceptability (depth-aware)

- Consider these labels only if they are **close to the user**.
- Labels: floor, road, window, door, stairs, rug, traffic light, car, bus, person.

C) Aesthetics (depth-aware)

- The overlay covers a visually salient region (strong edge contrast or vivid color).

Border Cases

- Objects near the overlay but not behind it are **not covered**.
- Transparent overlays still count as coverage if edges or textures are hidden.
- If only empty or uniform background is under the overlay, answer **1**.
- When uncertain about safety-relevant coverage, prefer **0**.

Explanation Rules

- Provide **exactly one short sentence**.
- Start the sentence with:
  - `The overlay covers` (for label 0)
  - `The overlay does not occlude` (for label 1)
- Mention the object or region behind the overlay.
- Do not mention uncertainty or the model itself.

OUTPUT FORMAT:
`<0 or 1> <one sentence explanation>`

A.2 Prompt per il task di placement

Placement Prompt

Task: You are given an image captured from a head-mounted camera. The image contains several **labeled candidate regions**, identified by **letters** shown on the image. Each letter marks the **center of a fixed-size UI overlay**.
If a letter is selected, the UI overlay is placed **centered on that letter**. If the overlay would extend outside the image boundaries, it is **shifted only as much as needed** to fully fit inside the frame.

Goal: Choose the **best letter** where placing the UI overlay would cause the **lowest interference with the scene**.

Decision Rules:

- Avoid covering objects that are **close to the user**.
- It is acceptable to cover **distant background surfaces**.
- Do **not** cover the object the user is currently interacting with or attending to, even if it is close.
- Prefer **visually neutral and non-intrusive areas** such as walls, floors, or empty regions.
- Avoid covering **people, hands, or functionally important objects**.

Objective: Minimize **visual, functional, and perceptual interference** with the scene.

OUTPUT FORMAT (MANDATORY):

- Valid answers are **only the letters shown in the image**.
- Reply with **only one single letter**.
- Do **not** include explanations, words, symbols, punctuation, or formatting.

Bibliografia

[1] Zhipeng Li, Christoph Gebhardt, Yves Inglin, Nicolas Steck, Paul Streli e Christian Holz. «SituationAdapt: Contextual UI Optimization in Mixed Reality with Situation Awareness via LLM Reasoning». In: *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. UIST '24. Pittsburgh, PA, USA: Association for Computing Machinery, 2024. ISBN: 9798400706288. DOI: 10.1145/3654777.3676470. URL: https://doi.org/10.1145/3654777.3676470 (cit. alle pp. 2, 11, 12, 26).

[2] David Lindlbauer, Anna Maria Feit e Otmar Hilliges. «Context-Aware Online Adaptation of Mixed Reality Interfaces». In: *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. UIST '19. New Orleans, LA, USA: Association for Computing Machinery, 2019, pp. 147–160. ISBN: 9781450368162. DOI: 10.1145/3332165.3347945. URL: https://doi.org/10.1145/3332165.3347945 (cit. a p. 2).

[3] Yifan Zhong et al. *A Survey on Vision-Language-Action Models: An Action Tokenization Perspective*. 2025. arXiv: 2507.01925 [cs.RO]. URL: https://arxiv.org/abs/2507.01925 (cit. alle pp. 2, 21).

[4] Florian Bordes et al. *An Introduction to Vision-Language Modeling*. 2024. arXiv: 2405.17247 [cs.LG]. URL: https://arxiv.org/abs/2405.17247 (cit. alle pp. 2, 7).

[5] Zongxia Li, Xiyang Wu, Hongyang Du, Fuxiao Liu, Huy Nghiem e Guangyao Shi. *A Survey of State of the Art Large Vision Language Models: Alignment, Benchmark, Evaluations and Challenges*. 2025. arXiv: 2501.02189 [cs.CV]. URL: https://arxiv.org/abs/2501.02189 (cit. alle pp. 2, 7).

[6] Brianna Zitkovich et al. «RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control». In: *Proceedings of The 7th Conference on Robot Learning*. A cura di Jie Tan, Marc Toussaint e Kourosh Darvish. Vol. 229. Proceedings of Machine Learning Research. PMLR, giu. 2023, pp. 2165–2183. URL: https://proceedings.mlr.press/v229/zitkovich23a.html (cit. a p. 2).

[7] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929 (cit. alle pp. 8, 13).

[8] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL]. URL: https://arxiv.org/abs/2302.13971 (cit. a p. 8).

[9] Aakanksha Chowdhery et al. *PaLM: Scaling Language Modeling with Pathways*. 2022. arXiv: 2204.02311 [cs.CL]. URL: https://arxiv.org/abs/2204.02311 (cit. a p. 8).

[10] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou e Jingren Zhou. *Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond*. 2023. arXiv: 2308.12966 [cs.CV]. URL: https://arxiv.org/abs/2308.12966 (cit. alle pp. 8, 10).

[11] Gokul Yenduri et al. *Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions*. 2023. arXiv: 2305.10435 [cs.CL]. URL: https://arxiv.org/abs/2305.10435 (cit. a p. 8).

[12] Junnan Li, Dongxu Li, Silvio Savarese e Steven Hoi. *BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models*. 2023. arXiv: 2301.12597 [cs.CV]. URL: https://arxiv.org/abs/2301.12597 (cit. a p. 9).

[13] Haotian Liu, Chunyuan Li, Qingyang Wu e Yong Jae Lee. *Visual Instruction Tuning*. 2023. arXiv: 2304.08485 [cs.CV]. URL: https://arxiv.org/abs/2304.08485 (cit. a p. 9).

[14] Jean-Baptiste Alayrac et al. *Flamingo: a Visual Language Model for Few-Shot Learning*. 2022. arXiv: 2204.14198 [cs.CV]. URL: https://arxiv.org/abs/2204.14198 (cit. a p. 9).

[15] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma e Furu Wei. *Kosmos-2: Grounding Multimodal Large Language Models to the World*. 2023. arXiv: 2306.14824 [cs.CL]. URL: https://arxiv.org/abs/2306.14824 (cit. a p. 10).

[16] Aisha Alansari e Hamzah Luqman. *Large Language Models Hallucination: A Comprehensive Survey*. 2025. arXiv: 2510.06265 [cs.CL]. URL: https://arxiv.org/abs/2510.06265 (cit. a p. 10).

[17] Charles Lovering, Michael Krumdick, Viet Dac Lai, Seth Ebner, Nilesh Kumar, Varshini Reddy, Rik Koncel-Kedziorski e Chris Tanner. *Language Model Probabilities are Not Calibrated in Numeric Contexts*. 2025. arXiv: 2410.16007 [cs.AI]. URL: https://arxiv.org/abs/2410.16007 (cit. a p. 10).

[18] John J. Dudley, Jason T. Jacques e Per Ola Kristensson. «Crowdsourcing Interface Feature Design with Bayesian Optimization». In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–12. ISBN: 9781450359702. DOI: 10.1145/3290605.3300482. URL: https://doi.org/10.1145/3290605.3300482 (cit. a p. 11).

[19] John J. Dudley, Jason T. Jacques e Per Ola Kristensson. «Crowdsourcing Design Guidance for Contextual Adaptation of Text Content in Augmented Reality». In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: 10.1145/3411764.3445493. URL: https://doi.org/10.1145/3411764.3445493 (cit. alle pp. 11, 36, 55).

[20] Gang Li e Yang Li. «Spotlight: Mobile UI Understanding using Vision-Language Models with a Focus». In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=9yE2xEj0BH7 (cit. alle pp. 11, 12).

[21] Joseph Redmon e Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV]. URL: https://arxiv.org/abs/1804.02767 (cit. a p. 12).

[22] Bashar Alsadik e Samer Karam. «The Simultaneous Localization and Mapping (SLAM)-An Overview». In: *Journal of Applied Science and Technology Trends* 2 (nov. 2021), pp. 120–131. DOI: 10.38094/jastt204117 (cit. a p. 12).

[23] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV]. URL: https://arxiv.org/abs/1405.0312 (cit. a p. 12).

[24] Shuai Bai et al. *Qwen2.5-VL Technical Report*. 2025. arXiv: 2502.13923 [cs.CV]. URL: https://arxiv.org/abs/2502.13923 (cit. a p. 13).

[25] Thomas Langerak, Kashyap Todi, Ben Lafreniere, Ruta Desai e Tanya Jonker. «XAIUI: User Belief-Driven Explainable AI for Context-Aware Adaptive Interfaces». In: *ACM Trans. Interact. Intell. Syst.* (nov. 2025). Just Accepted. ISSN: 2160-6455. DOI: 10.1145/3774657. URL: https://doi.org/10.1145/3774657 (cit. alle pp. 25, 57, 129).

[26] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li e Jianfeng Gao. *Set-of-Mark Prompting Unleashes Extraordinary Visual Grounding in GPT-4V*. 2023. arXiv: 2310.11441 [cs.CV]. URL: https://arxiv.org/abs/2310.11441 (cit. alle pp. 29, 31, 64).

[27] G. H. Levkine. *Prewitt, Sobel and Scharr Gradient 5x5 Convolution Matrices*. Image Process. Articles, Second Draft. Online; accessed 2025-12-29. 2012. URL: https://www.hlevkin.com/hlevkin/47articles/SobelScharrGradients5x5.pdf (cit. a p. 36).

[28] Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang e Ying Shan. «YOLO-World: Real-Time Open-Vocabulary Object Detection». In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 16901–16911. DOI: 10.1109/CVPR52733.2024.01599 (cit. a p. 36).

[29] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov e Rohit Girdhar. *Masked-attention Mask Transformer for Universal Image Segmentation*. 2022. arXiv: 2112.01527 [cs.CV]. URL: https://arxiv.org/abs/2112.01527 (cit. alle pp. 38, 69).

[30] René Ranftl, Alexey Bochkovskiy e Vladlen Koltun. *Vision Transformers for Dense Prediction*. 2021. arXiv: 2103.13413 [cs.CV]. URL: https://arxiv.org/abs/2103.13413 (cit. a p. 39).

[31] Aleksei Bochkovskii, Amaël Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R. Richter e Vladlen Koltun. *Depth Pro: Sharp Monocular Metric Depth in Less Than a Second*. 2025. arXiv: 2410.02073 [cs.CV]. URL: https://arxiv.org/abs/2410.02073 (cit. alle pp. 40, 41).

[32] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua e Sabine Süsstrunk. «SLIC superpixels». In: *Technical report, EPFL* (giu. 2010) (cit. a p. 67).

[33] Alexander Kirillov et al. *Segment Anything*. 2023. arXiv: 2304.02643 [cs.CV]. URL: https://arxiv.org/abs/2304.02643 (cit. a p. 68).

[34] Roboflow. *How to Fine-Tune Qwen2.5-VL with a Custom Dataset*. 2024. URL: https://blog.roboflow.com/fine-tune-qwen-2-5/ (cit. a p. 85).

[35] Research Computing Services, University of Cambridge. *Cambridge Service for Data Driven Discovery (CSD3)*. https://www.hpc.cam.ac.uk/. Accessed: 2026-01. 2024 (cit. a p. 87).

[36] Debapriya Maji, Soyeb Nagori, Manu Mathew e Deepak Poddar. *YOLO-Pose: Enhancing YOLO for Multi Person Pose Estimation Using Object Keypoint Similarity Loss*. 2022. arXiv: 2204.06806 [cs.CV]. URL: https://arxiv.org/abs/2204.06806 (cit. a p. 97).

[37] Michael Crawshaw. *Multi-Task Learning with Deep Neural Networks: A Survey*. 2020. arXiv: 2009.09796 [cs.LG]. URL: https://arxiv.org/abs/2009.09796 (cit. a p. 117).