UNIVERSITÀ DI PISA

DIPARTIMENTO DI INFORMATICA

Corso di Laurea in Informatica

REALTÀ AUMENTATA
Analisi delle criticità nel settore degli smartphone

Tesi di laurea

Relatore:
Prof. ANTONIO CISTERNINO

Candidato:
MICHELE MORISCO

ANNO ACCADEMICO 2018-2019

INDICE

SIGLE E ABBREVIAZIONI

AR	Augmented reality, realtà aumentata.
VR	Virtual Reality, realtà virtuale.
AV	Augmented virtuality, virtualità aumentata.
MR	Mixed Reality, realtà mista.
SLAM	Simultaneous localization and mapping.
VIO	Visual inertial odometry.
VO	Visual odometry.
IMU	Inertial measurement unit, unità di misura inerziale.
HMD	Head-mounted display, schermo montato sulla testa.
SDK	Software development kit.
CV	Computer Vision.
COM	Concurrent Odometry and Mapping.
TOF	Time-of-Flight, tempo di volo.
6DOF	Six degrees of freedom, sei gradi di libertà.
SDF	Signed distance function, funzione di distanza segnata.
TSDF	Truncated signed distance function, funzione di distanza segnata e troncata.
ICP	Algoritmo Iterative Closest Point.
CCD	Charge-Coupled Device, dispositivo ad accoppiamento di carica.

INTRODUZIONE

In questa tesi viene analizzato lo stato attuale della ricerca nella realtà aumentata sugli smartphone. In particolare, vengono esaminate le criticità in ambito mobile e le differenze tra la realtà aumentata e la realtà mista.

Le motivazioni che mi hanno spinto ad approfondire questo argomento sono l'interesse che ho nutrito in questi anni per la realtà aumentata e i suoi impieghi; inoltre, negli ultimi tempi ha avuto un gran successo evolvendosi e diventando sempre più fondamentale nel mercato odierno. Questa tecnologia ha avuto una significante crescita, in particolar modo nell'ambito mobile a causa dell'ascesa dell'intelligenza artificiale, di cui ha permesso la ricerca di nuovi approcci in Computer Vision.

Lo scopo di questo studio è quello di fornire una disamina della realtà aumentata applicata agli smartphone mettendo in evidenza i suoi limiti e verificando alcune possibili soluzioni a tali problemi.
Per raggiungere questo scopo, si è preso in esame i dispositivi provvisti di una fotocamera RGB e si è analizzato le differenze tra due diversi tipi di realtà aumentata:
Google ARCore, un SDK per lo sviluppo della realtà aumentata nei dispositivi portatili con sistema operativo Android; e Microsoft HoloLens, un dispositivo per la realtà aumentata che permette l'utente di interagire con gli oggetti virtuali in real-time.
Inoltre, si è sviluppata una applicazione mobile per testare alcuni suoi limiti riscontrati durante la ricerca.

La tesi comprende tre capitoli: nel primo capitolo, verranno date nozioni e conoscenze generiche per tutto quel che riguarda la realtà aumentata soffermandosi nel divario tra essa e la realtà mista, e inoltre, spiegando i principali algoritmi che utilizza. Nel secondo capitolo, verrà discussa la differenza tra ARCore e HoloLens in modo da definire le loro diverse metodologie per lo sviluppo della realtà aumentata; inoltre, verranno analizzati i principali algoritmi su cui si basano i due casi.

Infine, nel terzo e ultimo capitolo, si mostreranno i principali limiti e problemi che si possono riscontrare durante l'utilizzo della realtà aumentata negli smartphone; l'applicazione mobile creata con Google ARCore, per questo studio, dimostrerà alcune di queste criticità.

Con questa ricerca si è potuto ottenere un quadro della situazione odierna della realtà aumentata in ambito mobile; nelle conclusioni finali di questa tesi verranno esposti i risultati ottenuti riguardo all'intero studio.

Capitolo 1
REALTÀ AUMENTATA

Prima di affrontare il problema cruciale di questa tesi, in questo capitolo vengono date nozioni e conoscenze generiche riguardo alla realtà aumentata e le metodologie per applicarla.

1.1 **La realtà aumentata: definizione e tipi di AR**

La realtà aumentata (AR) negli ultimi anni ha avuto un gran successo, evolvendosi e diventando sempre più fondamentale nel mercato odierno: il mercato AR è stato valutato a 10,7 miliardi di dollari nel 2019 e si prevede che raggiungerà i 72,7 miliardi di dollari entro il 2024.

Questa tecnologia ha avuto, in particolar modo, un aumento della domanda nel commercio al dettaglio e nel commercio elettronico grazie alle sue caratteristiche, poiché aiuterà i rivenditori a ridurre il divario di acquisto tra clienti e prodotti online; soprattutto, grazie ai maggior investimenti fatti in questo mercato dalle più importanti compagnie come Facebook, Apple, Qualcomm, Inc. e Google. [1]

Per esempio, IKEA ha creato un'applicazione IOS che permette all'utente di piazzare mobili virtuali nella propria stanza; in questo modo, il cliente potrà verificare in anteprima l'oggetto da acquistare. Questa app si chiama IKEA Place. [2]

Inoltre, questa tecnologia ha avuto molti investimenti nel settore dell'intrattenimento; un esempio è Niantic, Inc. che nel 2016 ha sviluppato Pokémon Go, uno dei giochi mobile più redditizi in ambito AR.

Ma cos'è l'AR?

L'AR si tratta di un'esperienza interattiva del mondo reale in cui oggetti che risiedono in tale mondo sono arricchiti da informazioni percettive generate dal computer attraverso diverse modalità sensoriali.

Tale tecnologia si può definire come un sistema che soddisfa tre importanti caratteristiche, ovvero l'unione tra il mondo reale e quello virtuale, l'accurata registrazione 3D di oggetti reali e virtuali e, infine, l'interazione in tempo reale. [3][4]
Ci sono diversi tipi di AR sviluppati nel corso degli anni: quella marker-based che utilizza marcatori e immagini per poter calcolare la posizione e il tracking di un modello 3D.
E quella markerless che utilizza algoritmi di Computer Vision (CV) per definire la posizione di un target.
Dopodiché, l'AR si suddivide anche dal tipo di dispositivo utilizzato come gli Head-Mounted-Device (HDM), ovvero Microsoft HoloLens, e i dispositivi portatili come smartphone e tablet.

1.2 **Continuum realtà-virtualità: definizione e differenze tra AR, AV e MR**

Il continuum della virtualità è una scala continua che varia tra la realtà completamente virtuale ad una realtà completamente reale. Come illustrato nella Figura 1, il continuum realtà-virtualità comprende quindi tutte le possibili variazioni e composizioni di oggetti reali e virtuali.
Il motivo per cui un'ambiente completamente virtuale e reale siano situati alle estremità opposte è molto semplice; in un mondo virtuale i limiti delle leggi fisiche possono essere superati.
Invece, in un mondo reale, chiaramente, deve essere limitato dalle leggi della fisica.
Il caso a sinistra della Figura 1 definisce qualsiasi ambiente costituito solamente da oggetti reali e comprende tutto ciò che potrebbe essere osservato durante la visione di una scena del mondo reale o tramite uno schermo.
Il caso a destra, invece, definisce ambienti costituiti esclusivamente da elementi virtuali che comprendono simulazioni grafiche sia su monitor che immersive.
L'area tra i due estremi, in cui si mescolano sia il reale che il virtuale, è chiamata realtà mista (MR), ove essa è costituita dalla realtà aumentata e dalla virtualità aumentata. [5]

Figura 1. Reality–virtuality continuum (Wikipedia)

Dato che solitamente viene utilizzato AR e MR come sinonimi, questa configurazione ci permette di esprimere nel dettaglio le seguenti nomenclature.

In AR, all'utente vengono fornite ulteriori informazioni generate dal computer che migliorano la percezione della realtà.

La virtualità aumentata (AV) si riferisce, invece, a spazi prevalentemente virtuali in cui gli elementi fisici, come persone od oggetti, sono integrati dinamicamente e possono interagire con il mondo virtuale in tempo reale.

Infine, la MR, come è stato già detto precedentemente, comprende entrambe le due tecnologie sopracitate, non solo sovrapponendo gli oggetti virtuali agli oggetti del mondo reale, bensì consentendo all'utente di interagire direttamente con tali oggetti combinati. [6]

Del resto, la classificazione che fa riferimento la Figura 1 non è sufficiente nella configurazione della tassonomia, poiché definisce una visione vaga dell'ambiente generale dei diversi sistemi AR/AV; visto che definisce esplicitamente il concetto di display AR e li distingue dalla classe generale di display AV. [7]

Nonostante ciò, per questo studio, ci si accontenta della definizione mostrata dalla Figura 1.

In questa tesi, si farà riferimento per semplicità ad AR come la realtà aumentata nei dispositivi mobile quali smartphone e tablet. Invece, con MR verrà considerata la realtà aumentata nei dispositivi HMD come Microsoft HoloLens 2.

1.3 **I principali algoritmi in AR**

Nell'AR, il dispositivo deve calcolare la sua posizione nell'ambiente circostante. Quindi, esso calcola attraverso la relazione spaziale tra sé stesso e i features point.

Questo processo si chiama Simultaneous Localization and Mapping (SLAM).

Il seguente processo è utilizzato principalmente in robotica per permettere ad un robot, in un ambiente sconosciuto, di localizzarsi all'interno di una mappa che ha precedentemente costruito.

SLAM si suddivide in diversi algoritmi, per esempio, ORB-SLAM o Visual-SLAM.

Inoltre, ci sono altri modi per determinare la posizione e l'orientamento di un dispositivo, come la Visual Odometry (VO).

Questo processo è molto utilizzato da alcuni kit di AR come ARCore e ARKit, soprattutto, la variante Visual Inertial Odometry (VIO).

VIO utilizza l'unità di misura inerziale (IMU) all'interno del sistema VO, ovvero un sistema elettronico basato su sensori inerziali come accelerometri e giroscopi che permettono un monitoraggio della dinamica di un mezzo in movimento. [8]

Infatti, in quasi ogni smartphone sono presenti questi sensori che possono calcolare l'orientamento del dispositivo.

VO, come SLAM, tracciano i feature point[1] su un determinato numero di fotogrammi della camera e triangolano i features tracks in punti 3D usando la prima e ultima osservazione su ogni traccia. Dopodiché calcola la posa della telecamera rispetto ai punti 3D noti. [9]

Figura 2. Fotogramma di input (a sinistra) e feature tracks in output (a destra). I cerchi rappresentano le posizioni attuali dei feature point e le curve sono le feature track attraverso l'immagine.

[1] I feature points sono dei punti chiave che vengono calcolati attraverso ciò che vede la telecamera e servono per rappresentare una parte di un'immagine in modo da consentire la realizzazione del tracking dell'ambiente. Spesso i feature point rappresentano angoli, macchie, curve, punti o addirittura, strutture complesse come oggetti. [10]

La principale differenza tra VO e SLAM è che il VO si concentra maggiormente sulla coerenza locale e mira a stimare in modo incrementale il percorso della telecamera, posa dopo posa, e possibilmente eseguendo l'ottimizzazione locale.
Invece, SLAM mira ad ottenere una stima coerente a livello globale della traiettoria della camera e della mappa dell'ambiente. La coerenza globale si ottiene rendendosi conto che un'area precedentemente mappata è stata nuovamente visitata (loop closure, chiusura ad anello) e queste informazioni vengono utilizzate per ridurre l'accumulo di errori nelle stime. La Figura 3 mostra un panorama dei sistemi di VO e SLAM.

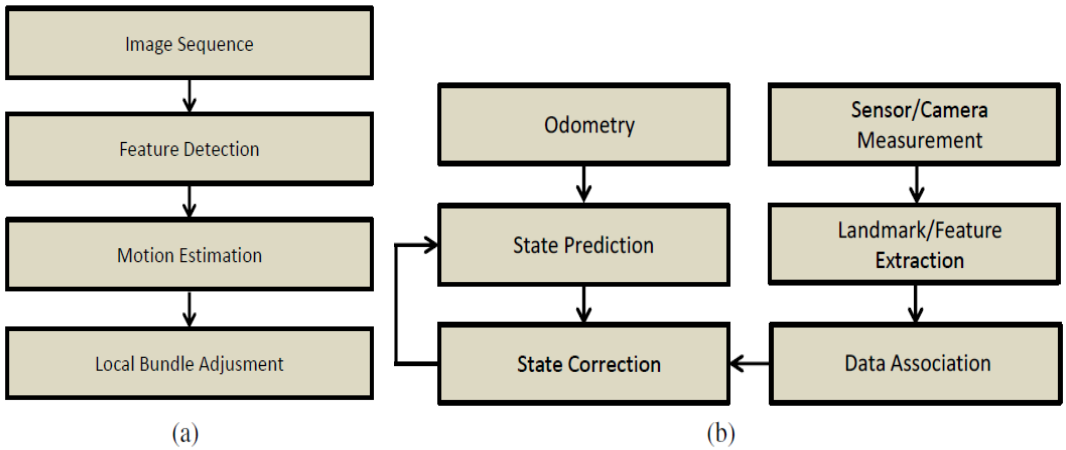

Figura 3. Diagramma che mostra le maggiori componenti di un: (a) VO (b) Filtro basato sul sistema SLAM [10]

VO è definita come il processo per determinare la posizione e l'orientamento di un dispositivo, analizzando una sequenza di immagini dell'ambiente circostante acquisite dalla telecamera. Tuttavia, la stima delle pose della telecamera in VO deve essere condotta in tempo reale.
Negli ultimi anni sono stati proposti molti metodi VO che possono essere suddivisi in metodi di camera monoculare e stereoscopica[2]. Inoltre, questi metodi vengono ulteriormente suddivisi in features matching, feature tracking e tecnologia del flusso ottico.
La metodologia SLAM, invece, permette di localizzare un dispositivo in un ambiente sconosciuto, mentre costruisce in modo incrementale una mappa nei suoi dintorni.

[2] La camera stereoscopica è una speciale tipo di fotocamera dotata di due obiettivi paralleli, posta alla stessa distanza degli occhi umani. Permette di simulare la visione binoculare e quindi creare immagini 3D. (12)

Negli ultimi anni si è nutrito un profondo interesse su questa metodologia, anche conosciuta come Visual-SLAM, grazie alle ricche informazioni disponibili dai sensori video passivi a basso costo rispetto alle telecamere IR. Tuttavia, il compromesso è un costo computazionale più elevato, con il requisito di algoritmi più sofisticati per l'elaborazione dell'immagine e per l'estrazione delle informazioni necessarie. Infatti, ci sono molte varianti di questo algoritmo. [11]

ARCore, per esempio, utilizza una sua versione di SLAM e VIO per determinare la posizione di un modello 3D collocato nell'ambiente, usando la stima del massimo delle probabilità a posteriori.

Capitolo 2
ARCORE E HOLOLENS

Questo capitolo descrive due diverse applicazioni dell'AR che si è preso in esame. Il primo si tratta di un SDK che permette un utente di sviluppare app in AR su dispositivi mobile; invece, il secondo è un dispositivo di MR, o meglio un HDM, che permette ad un utente di interagire con oggetti virtuali. Il capitolo seguente descriverà i principali algoritmi e approcci utilizzati da questi strumenti e le loro principali differenze.

2.1 **Google ARCore: Concurrent Odometry and Mapping**

Prima di analizzare il funzionamento di ARCore bisogna illustrare alcuni concetti fondamentali di questo SDK.

Il primo è il motion tracking ovvero quando uno smartphone si muove attraverso il mondo, ARCore utilizza il Concurrent Odometry and Mapping (COM) per capire dove si trova il telefono rispetto all'ambiente circostante. Esso rileva distinte caratteristiche o particolari nell'immagine catturata dalla fotocamera, chiamati feature points o descriptors, e usa questi punti per calcolare il cambiamento della sua posizione. Queste informazioni sono combinate con l'IMU per stimare la posizione e l'orientamento della telecamera, detta posa, rispetto all'ambiente nel tempo.

Adattando la posa della camera virtuale, che esegue il rendering dei contenuti 3D, con la posa della camera del dispositivo, uno sviluppatore potrà eseguire il rendering del contenuto virtuale con la prospettiva corretta. Il modello 3D renderizzato verrà sovrapposto all'immagine ottenuta dalla videocamera, facendolo apparire come un contenuto del mondo reale.

Il secondo concetto è il miglioramento costante nella sua comprensione dell'ambiente del mondo reale tramite i feature point e le superfici piane.

Google ARCore utilizza i feature point per rilevare le superfici come tavoli e muri. Queste informazioni permettono di posizionare oggetti virtuali su superfici piane.

Un'altra peculiarità è quella dei punti orientati che permettono di posizionare un oggetto su superfici angolate.

Quando si va posizionare un oggetto in un angolo, ARCore guarda i feature point vicini e li usa per provare a calcolare l'angolo della superficie dato dal feature point.

L'altra particolarità è migliorare la comprensione della propria posizione e del proprio ambiente utilizzando gli anchors (ancoraggi) per assicurare che ARCore segua la posizione dell'oggetto virtuale nel tempo.

Le superfici e i punti vengono chiamati trackable, ovvero un tipo speciale di oggetto che permette di ancorare un modello 3D ad esso.

Infine, Google ARCore fornisce una funzione che permette di creare una app AR il quale puntando la telecamera su una specifica immagine, essa fa emergere un oggetto virtuale. Questa funzione si basa molto sull'AR marker-based, vale a dire quella tecnologia che si concentra sul riconoscimento di un oggetto di forma nota per riuscire a identificare la posizione e l'orientamento nell'ambiente.

ARCore permette, quindi, di creare un database di immagini offline in modo da estendere i marcatori da riconoscere, l'algoritmo usato per il motion tracking è lo stesso della versione markerless; invece, per il riconoscimento delle immagini vengono usati algoritmi di Machine Learning di Google Vision. [13]

Entrando più nel dettaglio per quanto riguarda il motion tracking; ARCore, come è stato già detto in precedenza, utilizza un processo chiamato COM.

Questo processo è la base del SDK perché racchiude le principali caratteristiche dell'AR che si è già menzionato nel paragrafo 1.3, vale a dire gli algoritmi per poterla implementare.

COM si suddivide in diversi moduli: motion tracking, mapping e localization come viene mostrato dalla Figura 4.

Il modulo front-end del motion tracking ha il compito di ricevere i dati dai sensori ottici, inerziali e di profondità e traccia il movimento del dispositivo.

Esso calcola le diverse pose nel tempo in base ai feature descriptors, o detti anche feature points, che corrispondono all'impatto visivo [3] delle caratteristiche degli oggetti nell'ambiente e stimano le posizioni tridimensionali di queste caratteristiche; inoltre,

[3] L'impatto visivo degli oggetti si intende il modo in cui riflettono e trasmettono la luce. Il colore degli oggetti è determinato dalle parti dello spettro della luce che vengono riflesse o trasmesse senza essere assorbite. [14]

il processo di motion tracking calcola la posa del dispositivo da inviare al modulo back-end di mapping.

Questo modulo è configurato nel memorizzare una varietà di mappe basate su feature descriptors salvati e che periodicamente vengono aggiunti altri descriptors.

Il modulo costruisce una rappresentazione tridimensionale dell'ambiente basata sull'insieme di mappe memorizzate e sui features descriptors ricevuti dal modulo di motion tracking.

Questa rappresentazione tridimensionale dell'ambiente, ovvero una mappa, viene inviata al modulo di localizzazione.

Tale modulo esegue una chiusura ad anello (loop closure) minimizzando le discrepanze tra i feature descriptors corrispondenti per calcolare una posa localizzata; questa posa corregge l'orientamento nell'esposizione stimata dal modulo di motion tracking e viene inviata periodicamente a tale componente per creare l'output da mandare al modulo dell'API.

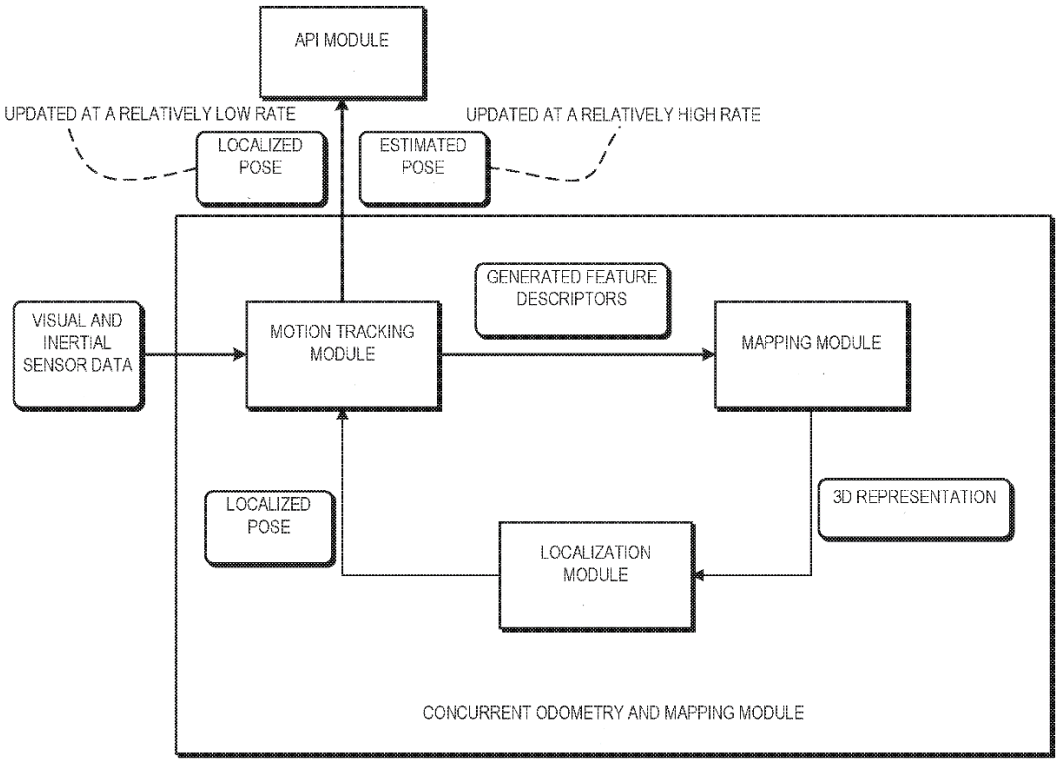

Figura 4. Diagramma che illustra il modulo COM

Tracciando il movimento basato su dati di sensori sia ottici che inerziali e costruendo la mappa tridimensionale basata sull'insieme di mappe salvate, nonché di feature descriptor periodicamente aggiornati, correggendo l'orientamento del movimento monitorato eseguendo una chiusura ad anello, il dispositivo è in grado di eseguire

motion tracking e costruire mappe di ambienti altamente precisi anche con risorse limitate.

Come viene mostrato dalla Figura 5, il modulo del motion tracking include due componenti: il modulo di identificazione delle feature e il mappatore di ambiente.

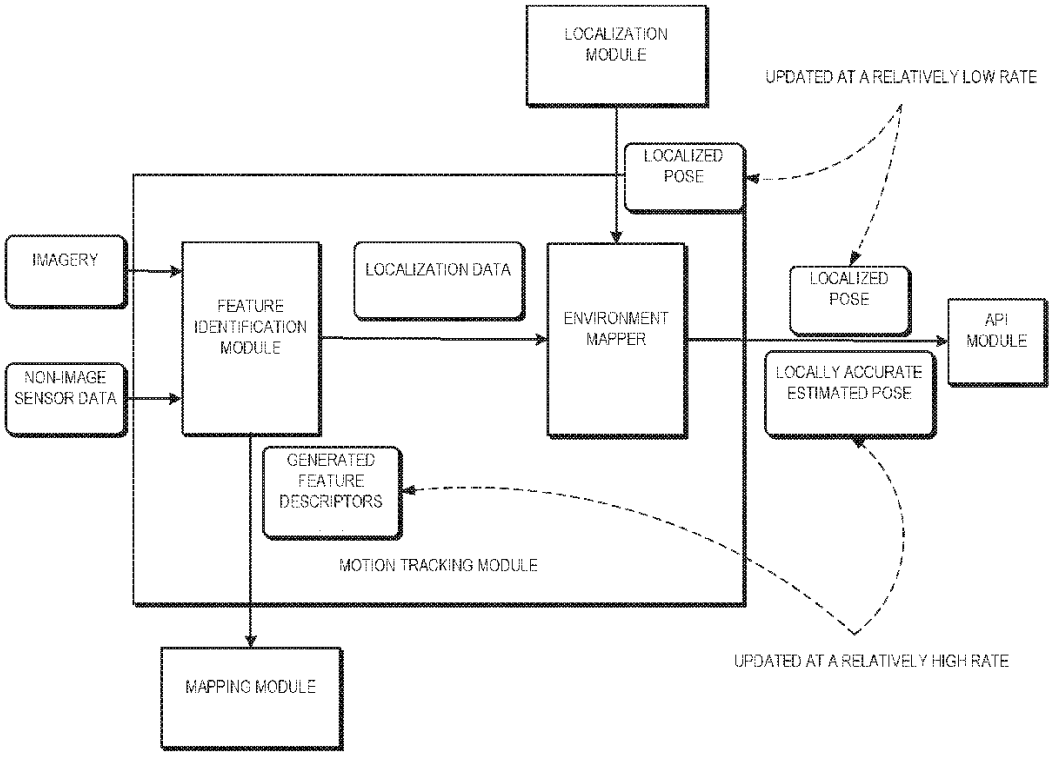

Figura 5. Diagramma che illustra il modulo del motion tracking

Il primo riceve dati ottici dalle immagini catturate dalla telecamera e da dati che concernono l'IMU; alla base di queste informazioni tale componente identifica le caratteristiche di un'immagine generando i features descriptors e li confronta con altri descriptors noti dalla cronologia del movimento tracciato precedentemente.

Inoltre, fornisce i features descriptors generati al modulo di mapping.

Il mappatore di ambiente è configurato per generare o modificare una posa locale calcolata in modo accurato in base ai dati di localizzazione. In particolare, la componente analizza i features descriptors nei dati di localizzazione per identificare la posizione delle caratteristiche di un fotogramma di riferimento del dispositivo; dopodiché stima l'evoluzione della posa del dispositivo nel tempo, nonché le posizioni dei punti 3D nell'ambiente.

Per trovare i valori di corrispondenza in base ai dati reperiti dai sensori, il mappatore d'ambiente risolve un problema di ottimizzazione non lineare.

Questa componente può riconciliare le posizioni relative dei differenti feature point per identificare la posizione di ciascun punto in un fotogramma di riferimento e memorizzarli in una posa locale calcolata in modo accurato. Successivamente, il modulo del motion tracking fornisce e aggiorna la collocazione del dispositivo stimata relativamente ad alto grado consegnandola al modulo API.

Il mappatore d'ambiente è anche configurato per interrogare periodicamente il modulo di localizzazione per una posa localizzata e aggiornata.

Il modulo della mappatura, come si nota dalla Figura 6, include una componente di memoria e una di matching per i feature descriptors.

La prima citata è configurata a memorizzare un insieme di mappe dell'ambiente osservate dal dispositivo.

Questo insieme di mappe possono includere feature descriptors noti di caratteristiche di oggetti nell'ambiente, che possono essere utilizzati collettivamente per generare una rappresentazione tridimensionale del mondo.

Il modulo di matching per i feature descriptors è configurato per ricevere i punti generati dal modulo del motion tracking e li confronta con quelli noti.

Inoltre, esso costruisce una rappresentazione tridimensionale dell'ambiente locale basato sui feature point noti presenti nell'insieme di mappe memorizzate e sui punti ricevuti periodicamente dal modulo di motion tracking.

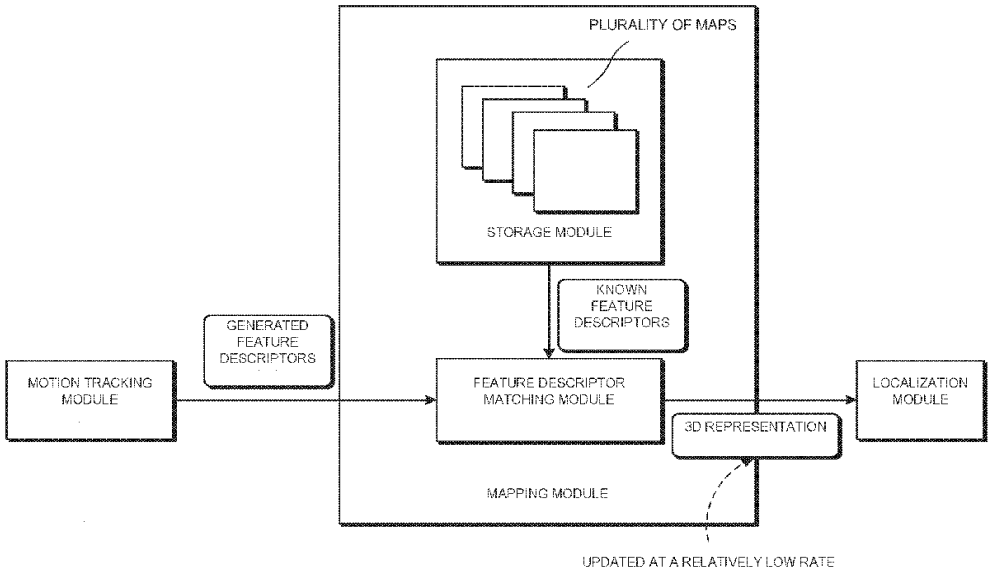

Figura 6. Diagramma che mostra il modulo di mappatura di COM

L’ultimo modulo del COM da vedere è quello di localizzazione che contiene al suo interno un rivelatore di discrepanze tra feature descriptors e una componente di loop closure (chiusura ad anello) come si può notare dalla Figura 7.

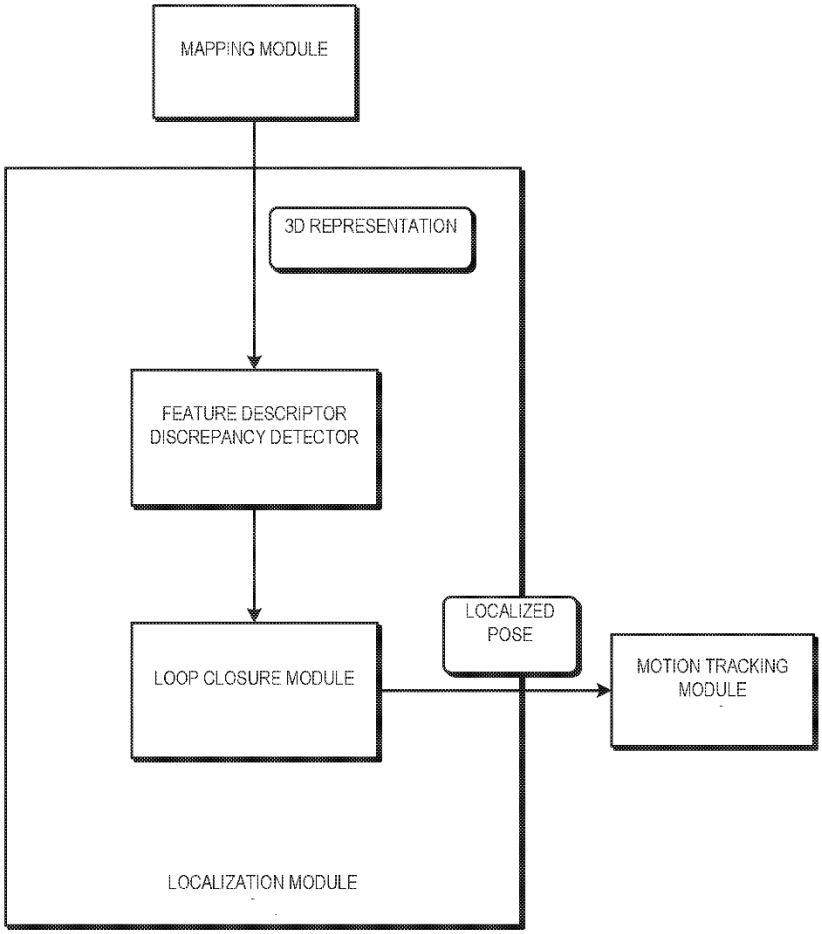

Figura 7. Diagramma che mostra il modulo di localizzazione

Il rivelatore di discrepanze è configurato a ricevere una rappresentazione tridimensionale dell’ambiente dal modulo di mappatura e trasforma i dati geometrici associati ai features descriptors, generati dalla posa calcolata, con i punti corrispondenti, da adeguare con i dati geometrici, associati ad una mappa memorizzata avente un punto corrispondente simile.

Quando il modulo di localizzazione trova un numero sufficiente di feature descriptors da quelli generati e una mappa memorizzata, per confermare che tali descriptors generati contengono descrizioni comuni tra i punti di riferimento, allora il modulo di localizzazione calcola una trasformazione tra feature point generati e quelli noti, disponendo i dati geometrici dei feature descriptors analoghi.

Il modulo della chiusura ad anello è configurato a trovare una posa corrispondente del dispositivo, dati i punti 3D nell’ambiente, e risolvere, con un algoritmo di co-

ottimizzazione, il perfezionamento della disposizione dei feature descriptors corrispondenti.

Il problema della co-ottimizzazione può essere risolto da un algoritmo di Gauss-Newton o da un altro algoritmo noto per l'ottimizzazione delle trasformazioni per generare una posa localizzata del dispositivo.

Il modulo della chiusura ad anello genera quindi una posa localizzata che corregge l'orientamento nella disposizione calcolata e la invia al modulo di motion tracking.

La posa localizzata può essere fornita ad un'applicazione in esecuzione sul dispositivo per consentire la realtà aumentata di riconoscere in modo più efficiente e accurato l'ambiente locale che il dispositivo ha attraversato in precedenza. [15]

2.2 **Microsoft HoloLens: KinectFusion**

Gli HoloLens sono un paio di smart glasses per la realtà mista sviluppati e prodotti da Microsoft.

Questo dispositivo utilizza quattro telecamere per il tracciamento della testa e due camere IR per il tracciamento degli occhi.

Per calcolare la profondità utilizza una telecamera a tempo di volo (ToF)[4] da 1 Megapixel e un IMU comprendente di giroscopio, accelerometro e magnetometro.

Quindi, la postura della testa viene determinata dalla posizione e orientamento degli HoloLens, stimato mediante l'IMU e l'algoritmo Iterative Closest Point (ICP).[5]

Questo dispositivo ricostruisce l'ambiente circostante come modello tridimensionale mediante la camera di profondità, le camere di comprensione ambientale e l'algoritmo KinectFusion. [16]

Quest'ultimo è stato originariamente sviluppato per la ricostruzione 3D utilizzando la camera di profondità del Kinect, ovvero un dispositivo con rivelamento del movimento.

[4] La telecamera a tempo di volo è uno strumento che permette di stimare in tempo reale la distanza tra la telecamera e gli oggetti o la scena inquadrati, misurando il tempo che occorre ad un impulso luminoso per percorrere il tragitto telecamera-oggetto-telecamera. [17]

[5] L'algoritmo ICP viene impiegato per minimizzare la differenza tra due nuvole di punti ed è spesso utilizzato per ricostruire superfici 2D o 3D da diverse scansioni, per localizzare i dispositivi e ottenere una pianificazione ottimale del percorso. Nell'algoritmo una nuvola di punti, detta riferimento o destinazione, viene mantenuta fissa mentre l'altra, l'origine, viene trasformata per adattarsi al meglio al riferimento. [18]

Mentre gli algoritmi per stimare la posa della telecamera e per estrarre la geometria dalle immagini, citati nel paragrafo 1.3, si sono evoluti nel tempo, anche le stesse tecnologie della fotocamera lo hanno fatto.

Le nuove telecamere di profondità basate sul ToF o quelle di rilevamento della luce strutturata offrono misurazioni precise della profondità in un dispositivo integrato. Con l'arrivo di Microsoft Kinect, tale rilevamento ha ricevuto un'ampia accessibilità a livello di consumatore.

Poiché gli utenti possono prendere un dispositivo Kinect e spostarlo per generare una ricostruzione 3D uniforme e costantemente aggiornata. Utilizzando solo i dati di profondità, il sistema traccia continuamente la posa a 6 gradi di libertà (6DOF) del sensore utilizzando tutti i dati in tempo reale disponibili dal sensore Kinect, anziché un sottoinsieme di funzioni astratte, e integra le misurazioni di profondità in un denso modello volumetrico.

Utilizzando i dati di profondità, il sistema può funzionare in completa oscurità, mitigando qualsiasi problema di scarsa illuminazione. [19]

L'algoritmo KinectFusion si suddivide in quattro componenti come viene mostrato dalla Figura 8:

Misurazione della superficie: una fase di preelaborazione, in cui una mappa densa di vertici e una mappa normalizzata vengono generate dai dati grezzi sulla profondità ottenute dagli HoloLens.

Ricostruzione della superficie: il processo globale di fusione della scena, in cui data la posa determinata dal tracciamento dei dati sulla profondità, la misurazione della superficie è integrata nel modello di scena mantenuto con una rappresentazione volumetrica della funzione di distanza troncata.

Previsione della superficie: Questo evento avviene eseguendo il ray casting della funzione di distanza nel fotogramma considerato per fornire una previsione di superficie densa rispetto alla quale è disposta la mappa di profondità in tempo reale.

Stima della posa del sensore: il rilevamento del sensore in tempo reale viene ottenuto applicando l'algoritmo ICP su più scale della superficie prevista e la misurazione dal sensore corrente.

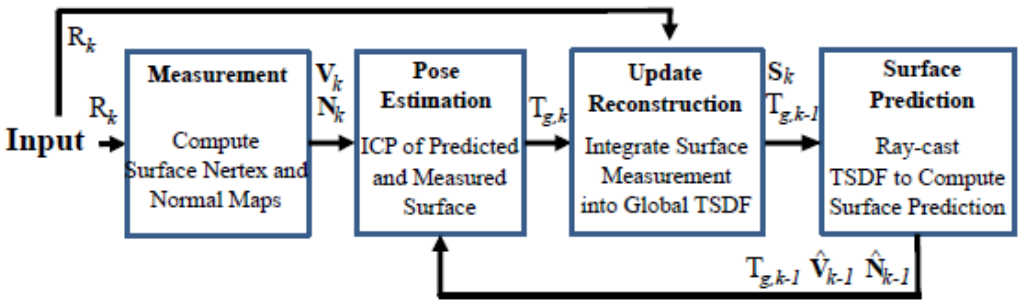

Figura 8. Diagramma del sistema del KinectFusion

In KinectFusion, viene usata la matrice di calibrazione costante della telecamera K per trasformare dal piano del sensore al pixel dell'immagine. Trasforma un punto dal fotogramma locale k al fotogramma globale usando la matrice di trasformazione SE_3[6] 6DOF.

La funzione $q = \pi(p)$ include proiezioni prospettiche dopo l'omogeneizzazione.

Durante la fase di preelaborazione, cioè nello stato di misurazione della superficie, si ha al tempo k una mappa di profondità grezza, denominata $R_k(u)$ dove ogni pixel di immagine $u = (u, v)^T$.

Ottiene una misura del punto metrico nel fotogramma k del sensore $p_k = R_k(u)K^{-1}$ e viene applicato un filtro bilaterale che riduce il rumore delle immagini preservando i contorni; in questo modo, si ottiene la mappa della profondità preservata da discontinuità con riduzione del rumore $D_k(u)$. [20]

Filtra i valori di profondità del sensore in riferimento al fotogramma k che produce la mappa dei vertici V_k dove $V_k(u) = D_k(u)K^{-1}$.

Dopodiché trova la normale della mappa V_k usando il prodotto incrociato con la mappa dei vertici vicini; infine, data la camera per la trasformazione globale del fotogramma del vertice usando la trasformazione $T_{g,k}$ completamente omogenea, associato al calcolo della superficie, il vertice del fotogramma globale è $V_k^g(u) = T_{g,k}\,\dot{V}_k(u)$ e la mappatura equivalente normalizzata nel fotogramma globale è $N_k^g(u) = R_{g,k}N_k(u)$.

L'output del primo stato, come mostrato nella Figura 8, è proprio V_k e N_k, rispettivamente la mappa dei vertici e la mappa normalizzata.

[6] L'insieme euclideo $SE_3 := \{R, t | R \epsilon SO_3, t \epsilon R^3\}$ dove SO_3 costituisce il gruppo di trasformazioni ortogonali appropriate, o meglio, l'insieme speciale ortogonale. L'insieme di matrici appartenenti a SO_3 rappresentano solamente l'operazione di rotazioni pure.

Nello stato di aggiornamento di ricostruzione della superficie viene calcolata la media tra le funzioni di distanza segnata (SDF) di più nuvole di punti allineate al fotogramma globale che restituisce la fusione globale della superficie.

La funzione SDF troncata (TSDF) permette di impedire alle superfici rilevate di interferire tra loro, in caso di occlusioni.

Vengono memorizzati due valori ad ogni posizione di TSDF, il segnale troncato corrente denominato $F_k(p)$ e il vettore di peso $W_k(p)$.

Troncare l'incertezza del valore di profondità in modo tale che si trovi in un intervallo $[-\mu, +\mu]$; i punti non visibili al di fuori di quei valori verranno ignorati, SDF quindi descrive la regione di incertezza solo dove esiste la misurazione.

La fusione consiste nel calcolare la media ponderata semplice di tutti i TSDF calcolati per ciascuna mappa di profondità, in modo da eliminare il rumore presente globalmente da ogni misurazione TSDF con rumore.

Utilizzando la ricostruzione più aggiornata disponibile arriva la possibilità di calcolare una previsione di superficie, trasformando la superficie codificata nel livello zero, impostato $F_k = 0$, in una camera virtuale con la stima attuale $T_{g,k}$. Quindi, la previsione della superficie viene memorizzata come mappa di vertici $\hat{V}_k$ e come mappa normalizzata $\hat{N}_k$ nel fotogramma di riferimento k e viene utilizzata nella fase successiva di stima della posa della telecamera.

Nella fase di stima della posa, anziché utilizzare la selezione delle feature, viene adottato un approccio di associazione dei dati di proiezione rapida, poiché viene applicato molto meno movimento tra un fotogramma all'altro.

KinectFusion traccia il frame del sensore corrente adattando una misurazione della superficie attiva (V_k, N_k) rispetto alla previsione del fotogramma precedente (V_{k-1}, N_{k-1}) e la rilocalizzazione è necessaria se il tracciamento è stato perso.

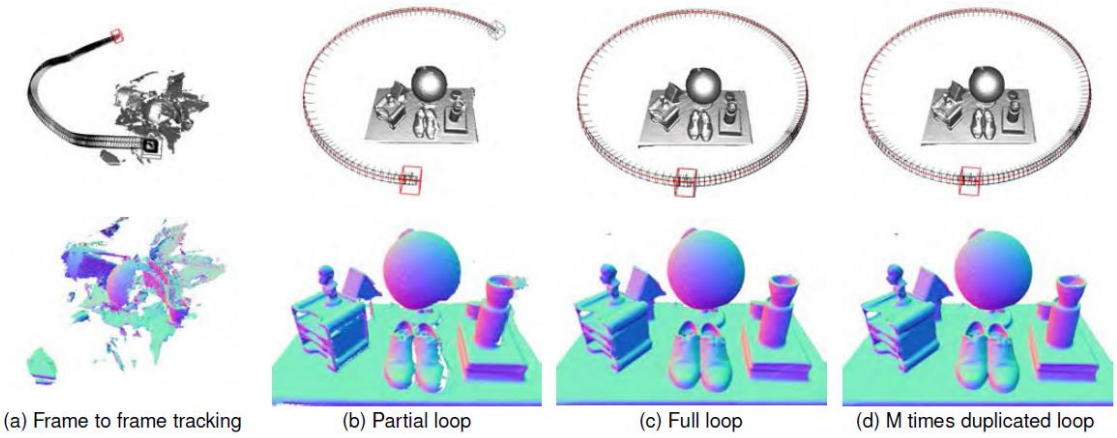

Figura 9. Osservazione del movimento circolare per evidenziare le caratteristiche di KinectFusion

Come viene mostrato dalla Figura 9, per ogni colonna l'immagine superiore mostra la traiettoria stimata del sensore e nella parte inferiore mostra la qualità di ricostruzione con una mappatura normalizzata.

Il tracciamento da fotogramma a fotogramma (a), in cui la posa di ciascun nuovo fotogramma è calcolato dalla registrazione rispetto all'ultimo frame; dunque, il rapido accumulo di errori viene provocato da una traiettoria non-circolare e si ha, quindi, una scarsa ricostruzione.

Per quanto riguarda all'approccio (b), l'elaborazione viene interrotta creando una traiettoria circolare completa per due terzi.

Nell'immagine (c) viene mostrata la chiusura ad anello dove l'ultimo fotogramma elaborato è una duplicazione del primo; invece, in (d) si tiene conto degli stessi dati calcolati in (c) fornendoli all'algoritmo per quattro volte.

Questo migliora l'ordinamento tra i fotogrammi in loop closure e gli artefatti da ricostruire sono ridotti.

L'algoritmo fonde N fotogrammi TSDF per stimare la posa del sensore da frame a frame usando ICP, i frame 1…L, con $L < N$, sono passati attraverso la mappatura e la pipeline di tracciamento per la chiusura incompleta del loop, invece, i fotogrammi 1…N hanno passato il tracciamento e la mappatura per la chiusura completa del loop.[21]

Capitolo 3
I LIMITI DELLA REALTÀ AUMENTATA

Questo capitolo mostrerà i maggiori limiti e problemi della realtà aumentata, dove si è utilizzato Google ARCore come esempio, e inoltre si utilizzeranno altri SDK per verificare alcune possibili soluzioni a tali difficoltà.
Per questa sezione, si è sviluppata una piccola applicazione mobile che testerà i principali aspetti di Google ARCore verificando i limiti di questa tecnologia. Per la realizzazione dell'app si è utilizzato Unity3D, un motore grafico multipiattaforma.

3.1 **L'efficienza dell'hardware sui sistemi AR**

Gli smartphone sono migliorati sotto ogni aspetto rilevante per l'AR negli ultimi anni, ovvero ci sono processori più veloci, più memoria, interfacce di input migliorate, display più grandi e superiori. L'ecosistema degli smartphone fornisce tutto quel che serve per distribuire app in AR come una soluzione esclusivamente software per un pubblico di massa. Tuttavia, il fatto che ci siano stati molti miglioramenti tecnici e logistici, ci sono ancora importanti ostacoli per una distribuzione in larga scala. [22] Di seguito, verranno mostrati alcuni ostacoli legati all'hardware.

3.1.1 *Qualità e gestione della fotocamera*

La qualità di riproduzione delle immagini dei sensori della fotocamera, installate negli smartphone, è carente in cattive condizioni di illuminazione. Le immagini possono essere sfocate o i colori potrebbero soffrire di irregolarità. Inoltre, l'accesso al livello hardware del sensore della fotocamera è di norma vietato. Le API forniscono un accesso solo di alto livello alla fotocamera, impedendo il controllo dell'esposizione, dell'apertura o della lunghezza focale. I sensori CCD[7] di piccole dimensioni sono la

[7] I sensori CCD (Charge-Coupled Device) consistono in un circuito integrato formato da una riga, o da una griglia, di elementi semiconduttori in grado di accumulare una carica elettrica proporzionale all'intensità della radiazione elettromagnetica che li colpisce. Questi elementi sono accoppiati in modo che ognuno

causa delle maggiori quantità di rumore nella registrazione video o immagini della fotocamera, rovinando le prestazioni degli algoritmi di CV.

Inoltre, la qualità persa durante l'acquisizione dell'immagine può non essere compensata facilmente da ulteriori fasi di elaborazione.

Nel caso degli HoloLens, utilizzando un sensore di profondità ToF e telecamere IR, gestisce più adeguatamente i problemi sopracitati. Infatti, il sensore di profondità può funzionare anche con una scarsa illuminazione visto che l'algoritmo KinectFusion, menzionato nel paragrafo 2.2, permette la mappatura tridimensionale dell'ambiente circostante. In questo modo, con le dovute precauzioni, HoloLens può utilizzare al meglio gli algoritmi di CV per poter creare applicazioni in AR.

Negli ultimi mesi, Apple sta realizzando, assieme a Sony, un sensore 3D per i loro smartphone e tablet che utilizzano il sensore ToF, questo migliorerà ulteriormente lo sviluppo di app in AR sul mercato mobile.

3.1.2 *Consumo energetico*

I sensori della fotocamera richiedono molta energia quando funzionano costantemente con frame rate elevati, e inoltre, gli algoritmi di CV sono esigenti dal punto di vista computazionale e, dunque, tendono a scaricare rapidamente la batteria degli smartphone.

Allo stesso modo, i sensori e le interfacce di rete, come Wi-Fi e connessioni dati, utilizzano abbondantemente l'energia.

Perciò, le applicazioni AR eseguite negli smartphone e tablet causano il consumo rapido della batteria; di conseguenza, devono essere progettate in modo che vengano utilizzate solo per brevi periodi, piuttosto, che come funzionalità sempre attiva.

In questo caso, Google ARCore, in particolare nell'app realizzata per questa tesi, si è progettato un modo per prolungare la batteria permettendo allo schermo di andare in stand-by quando non si sta facendo il tracking dell'oggetto.

di essi, sollecitato da un impulso elettrico, possa trasferire la propria carica ad un altro elemento adiacente. Inviando al dispositivo una sequenza temporizzata d'impulsi, si ottiene in uscita un segnale elettrico grazie al quale è possibile ricostruire la matrice dei pixel che compongono l'immagine. [23]

3.1.3 *Dipendenza dalla rete*

L'accesso a grandi quantità di dati in remoto soffre di diversi problemi. Prima di tutto, la latenza della rete può arrecare danno al comportamento istantaneo della AR, causando lag[8].

In secondo luogo, se si volesse utilizzare la tecnologia AR in modo costante, l'accesso ai dati remoti sarebbe possibile solo con tariffari costosi e, che in questo momento, possono non essere ancora disponibili.

Infine, la copertura della rete potrebbe essere insufficiente in alcune aree. Ciò lascia le applicazioni AR completamente autonome come una delle opzioni praticabili, il che implica un uso intensivo delle capacità di archiviazione del dispositivo.

Ma, un'altra opzione, è quella dell'impiego del 5G, ovvero la tecnologia di rete mobile che supererà il 4G.

Questa tecnologia avrà una velocità dati di decine di megabit al secondo per decine di migliaia di utenti con parecchie centinaia di migliaia di connessioni simultanee. Inoltre, con la copertura migliorata e l'efficienza dei segnali potenziata si avrà una latenza significativamente ridotta in confronto con il 4G. [25]

Nei prossimi anni, con l'investimento sul 5G, l'AR potrà offrire migliori funzionalità andando ad aumentare l'utilizzo di questa tecnologia per ogni singolo utente.

[8] Il lag, o ritardo, è un fenomeno fastidioso nel caso in cui il tempo di comunicazione fra due computer sia estremamente alto, dovuto alla latenza delle informazioni. [24]

3.2 **Limitazioni nella Computer Vision**

Uno dei vantaggi degli smartphone è che la localizzazione del dispositivo non è relegata soltanto al sensore della fotocamera, bensì è possibile utilizzare altri sensori disponibili come l'accelerometro, il giroscopio e il GPS. Si è già menzionato nel paragrafo 2.1 che Google ARCore utilizza una sua versione di VIO. Questo permette di sviluppare una localizzazione del dispositivo più robusta e rapida, molto adatta per gli smartphone.

Tuttavia, anche con l'aiuto dei sensori, la localizzazione basata sugli algoritmi di CV rimane un compito oneroso.

3.2.1 *Ambienti inconsistenti: AR marker-based*

Un problema della CV è la difficoltà nel tracciamento degli ambienti interni che presentano texture sterili o addirittura prive di particolarità; ovvero, quegli ambienti dove non sono presenti qualità della superficie, ad esempio un muro monocromatico, su cui verrà inserito l'oggetto virtuale.

Come è già stato accennato nel paragrafo 2.1, Google ARCore utilizza i feature point per rilevare la presenza di superfici e per poter calcolare il tracking degli oggetti virtuali posizionati nella scena; se la scena non presenterà alcune caratteristiche visive considerevoli, verranno applicati molti meno feature point.

Questo complicherà al dispositivo di localizzarsi e di conseguenza, anche la correttezza nell'applicazione dell'AR sarà difficoltosa.

L'AR marker-based utilizza immagini che vengono rilevate dalla fotocamera e vengono utilizzate per calcolare la posizione degli oggetti virtuali collocati nella scena; la maggior parte sono in bianco e nero, ma si possono usare pure i colori purché il contrasto tra loro sia riconosciuto correttamente dalla fotocamera.

I marcatori possono essere di due tipi: quelli semplici, costituiti da una o più forme base formate da quadrati neri su sfondo bianco; oppure quelli elaborati che utilizzano immagini semplici che vengono riconosciute e lette dalla fotocamera.[26]

Google ARCore ha la capacità di creare le Augmented Image, ovvero immagini che vengono riconosciute dall'applicazione e tale app suggerisce la posizione fisica di queste immagini. L'API tiene traccia sia delle immagini statiche che quelle in movimento, per esempio un'immagine posta su un piano che viene mossa dall'utente. ARCore dopo aver iniziato a tracciare il marcatore, fornisce stime per

posizione, orientamento e dimensioni fisiche di esso. Queste stime vengono continuamente perfezionate man mano che il dispositivo raccoglie più dati.
Una volta che l'immagine è stata completamente rilevata, ARCore è in grado di continuare a tracciare la posizione e l'orientamento dell'immagine, anche se l'immagine si è temporaneamente spostata fuori dal campo visivo della fotocamera.
I marker devono soddisfare alcuni requisiti, di cui devono essere ricchi di dettagli, avere un buon contrasto tra i colori e non utilizzare pattern ripetitivi. Inoltre, l'immagine non deve avere una compressione pesante e la sua risoluzione deve essere di almeno 300x300 pixel. Se i marcatori si attengono a questi requisiti, ARCore funzionerà con prestazioni quasi ottimali durante la fase di tracking. [27]
La Figura 10 mostra la parte di codice che aggiunge un oggetto virtuale all'immagine inquadrata dalla fotocamera. Viene associato un ancoraggio all'immagine in modo da permettere il tracking corretto del modello 3D.

```
/* Crea un ancoraggio per le augmented images che si stanno tracciando
   e che non si sono tracciate precedentemente. Rimuove il prefab se non c'è più tracking.*/
foreach (var image in m_TempAugmentedImages)
{
    ARImageVisualizer visualizer = null;
    //Ottiene il valore dal dizionario corrispondente all'immagine inquadrata.
    m_Visualizers.TryGetValue(image.DatabaseIndex, out visualizer);

    //Controlla se l'immagine sia in fase di tracking e che visualizer sia vuoto.
    if (image.TrackingState == TrackingState.Tracking && visualizer == null)
    {
        // Crea un ancoraggio per assicurare che ARCore mantenga il tracking sull'immagine e lo aggiunge nella lista.
        Anchor anchor = image.CreateAnchor(image.CenterPose);
        visualizer = (ARImageVisualizer)Instantiate(
          AugmentedImageVisualizerPrefab, anchor.transform);
        visualizer.Image = image;
        m_Visualizers.Add(image.DatabaseIndex, visualizer);
    }
    //Altrimenti, se la fase di tracking si è fermata viene rimossa dalla lista e l'oggetto viene distrutto.
    else if (image.TrackingState == TrackingState.Stopped && visualizer != null)
    {
        m_Visualizers.Remove(image.DatabaseIndex);
        GameObject.Destroy(visualizer.gameObject);
    }
}
```

Figura 10. Frammento di codice che illustra la collocazione dell'oggetto quando si inquadra l'immagine.

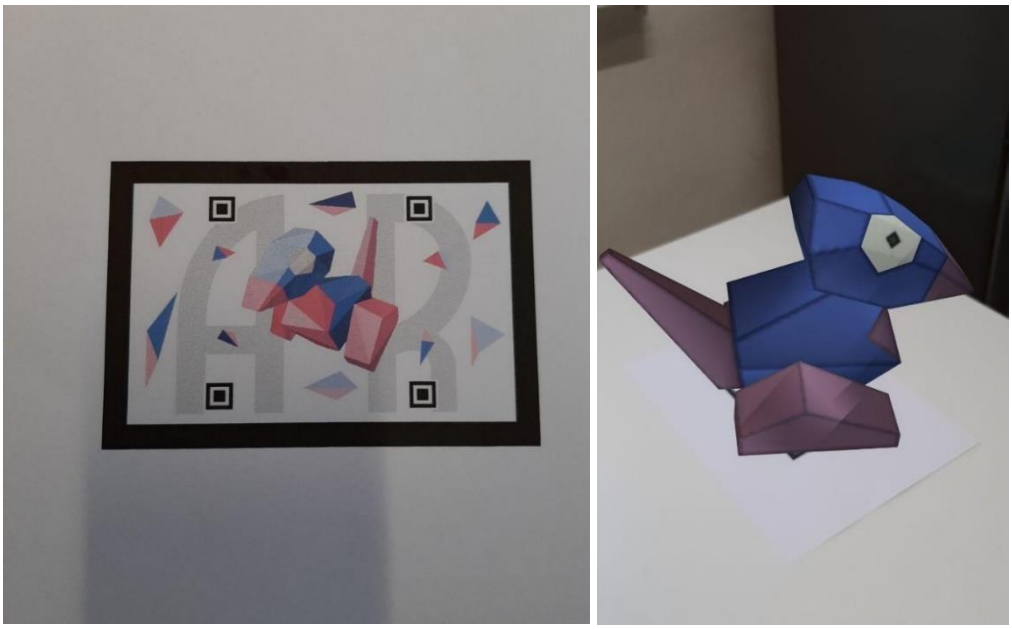

Figura 11. A sinistra il marker da visualizzare e a destra il tracking con l'oggetto virtuale.

3.2.2 *L'illuminazione nella AR*

Un'altra questione nella CV è l'illuminazione, si scopre che negli ambienti esterni un numero di feature presenti nell'immagini naturali non si riferisce a caratteristiche naturali. Le ombre proiettate dagli oggetti nella scena fanno sì che si formino angoli e linee che si muovono dinamicamente al variare delle condizioni di illuminazione o meteorologiche. Di conseguenza, il sovrannumero di valori anomali e discrepanze influisce sulla qualità della localizzazione, indipendentemente dalla scelta dell'algoritmo.

Anche se negli ultimi anni ci sono stati progressi in questo settore, non si è ancora raggiunto un risultato soddisfacente, soprattutto, per gli ambienti esterni.

Google ARCore per stimare l'illuminazione utilizza un API che analizza una determinata immagine per scovare riferimenti visivi e fornire informazioni dettagliate sull'illuminazione della scena. Quindi, vengono utilizzate queste informazioni durante il rendering degli oggetti virtuali per illuminarli nelle stesse condizioni della scena in cui sono posizionati.

La stima dell'illuminazione svolge la maggior parte del lavoro fornendo dati dettagliati che consentono di imitare i vari tratti dell'illuminazione durante il rendering di oggetti virtuali. Questi tratti sono le ombre, la luce ambientale, l'ombreggiatura, i riflessi e la specular highlight, o detta anche lumeggiatura. [28]

```
//Controlla se la stima della luce è nella modalità che genera l'intensità ambientale e la correzione del colore.
if (estimate.Mode == LightEstimationMode.AmbientIntensity)
{
    // Normalizza l'intensità dei pixel per il grigio medio nello spazio gamma.
    const float middleGray = 0.466f;
    float normalizedIntensity = estimate.PixelIntensity / middleGray;

    // Applica la correzione del colore lungo la normalizzazione dell'intensità dei pixel nello spazio gamma.
    Shader.SetGlobalColor(
        "_GlobalColorCorrection",
        estimate.ColorCorrection * normalizedIntensity);

    // Setto _GlobalLightEstimation per la retro-compatibilità.
    Shader.SetGlobalFloat("_GlobalLightEstimation", normalizedIntensity);
}
```

Figura 12. Frammento di codice che mostra la gestione della luce ambientale.

```
//Altrimenti se non genera luce ambientale, controlla che il frame abbia un timestamp diverso.
else if (m_LightEstimateTimestamp != estimate.Timestamp)
{
    m_LightEstimateTimestamp = estimate.Timestamp;

    //Controlla che la luce direzionale sia presente.
    if (DirectionalLight != null)
    {
        //La luce direzionale viene attivata/abilitata se disattivata.
        if (!DirectionalLight.gameObject.activeSelf || !DirectionalLight.enabled)
        {
            DirectionalLight.gameObject.SetActive(true);
            DirectionalLight.enabled = true;
        }

        /*aggiorna la rotazione della luce direzionale con la rotazione della luce direzionale reale stimata da ARCore.
          Anche il colore della luce viene aggiornato.*/
        DirectionalLight.transform.rotation = estimate.DirectionalLightRotation;
        DirectionalLight.color = estimate.DirectionalLightColor;
    }

    RenderSettings.ambientMode = AmbientMode.Skybox;
    RenderSettings.ambientProbe = estimate.AmbientProbe;

    /*Controlla se la sessione corrente sia in modalità EnviromentalHDRWithReflections, in modo da aggiornare la cubemap
      del riflessi.*/
    if (estimate.Mode == LightEstimationMode.EnvironmentalHDRWithReflections)
    {
        RenderSettings.defaultReflectionMode = DefaultReflectionMode.Custom;
        RenderSettings.customReflection = estimate.ReflectionProbe;
    }
}
```

Figura 13. Frammento di codice che mostra la gestione della luce direzionale.

Nell'app realizzata per testare le funzionalità di ARCore si esamina questo task: nella Figura 12 si può notare un frammento del codice che gestisce la luce ambientale applicandola all'oggetto virtuale, prima di applicare la correzione del colore viene normalizzata l'intensità dei pixel per il corrente frame, ottenuto dalla fotocamera. L'intensità dei pixel è rappresentata tramite un intervallo chiuso da 0 a 1, dove lo 0 rappresenta il nero e l'1 il bianco.

La Figura 13 mostra il caso in cui bisogna applicare la luce direzionale nella scena considerando la fonte di luce presente nella realtà; ARCore aggiorna la rotazione e

il colore della luce direzionale con quella presente nella scena reale usando i quaternioni[9] per ruotare la luce.

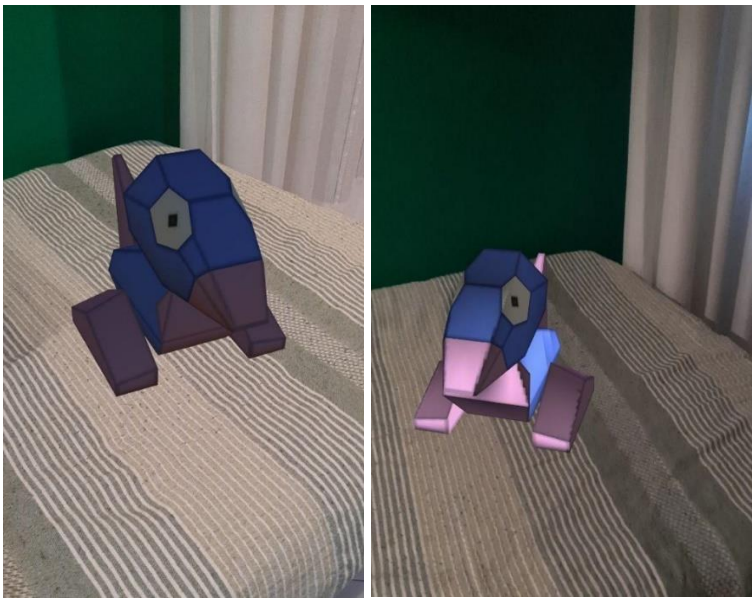

Figura 14. A sinistra l'oggetto virtuale gli viene applicata la luce ambientale, a destra è stata applicata l'HDR[10] ambientale senza riflesso.

3.2.3 *L'occlusione*

Una importante sfida dell'AR il quale ci sono ricerche che si stanno mobilitando a trovare una soluzione è proprio l'applicazione del realismo dell'AR su scenari dinamici, in particolare l'occlusione.

L'occlusione si verifica quando un oggetto in uno spazio 3D sta bloccando un altro oggetto dalla vista. In AR, per mantenere alti i livelli di realismo, gli oggetti virtuali e la scena reale devono essere perfettamente allineati e consentire agli oggetti di comportarsi come farebbero in condizioni normali. Uno dei problemi principali nell'applicare l'occlusione è la mancanza di informazioni utilizzando i metodi tradizionali per la ricostruzione 3D. Il secondo problema è il movimento della fotocamera tra due fotogrammi; di questo, le app AR non sono consapevoli di tutte le differenze e non possono compensare adeguatamente questo movimento, provocando il jitter di oggetti virtuali, ovvero una variazione di una o più caratteristiche dell'oggetto tracciato. [31]

---

[9] Il quaternione è un oggetto formale di tipo $a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}$ dove a, b, c, d sono numeri reali e $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ sono dei simboli che si comportano in modo simile all'unità immaginaria dei numeri complessi. Essi trovano un'applicazione nella modellizzazione delle rotazioni nello spazio. [29]

[10] L'High Dynamic Range (HDR) consente di visualizzare una gamma più ricca di colori rispetto allo Standard Dynamic Range (SDR). [30]

Recentemente, per risolvere questo problema Google ha sviluppato Depth API. Questo kit consente di utilizzare gli algoritmi di profondità del movimento per creare una mappa di profondità utilizzando una singola fotocamera RGB. La mappa viene creata prendendo più immagini da diverse angolazioni e confrontandole mentre si sposta il telefono per stimare la distanza di ogni pixel.

In questo modo, si può applicare l'occlusione correttamente negli oggetti virtuali piazzati nella scena. [32]

Chiaramente questo kit è ancora in piena fase di sviluppo e non è possibile testare le sue funzionalità, quindi per adesso bisogna accontentarsi delle demo o delle dimostrazioni video mostrate da Google.

Però, Apple in questo campo ha già sviluppato People Occlusion, ovvero una funzionalità di ARKit che permette di applicare l'occlusione degli oggetti virtuali solamente con le persone. Questa funzione identifica le aree nel feed della telecamera in cui risiedono le persone e impediscono all'oggetto virtuale di disegnare i suoi pixel nella regione in cui risiede l'individuo, come viene mostrato nella Figura 16. [33]

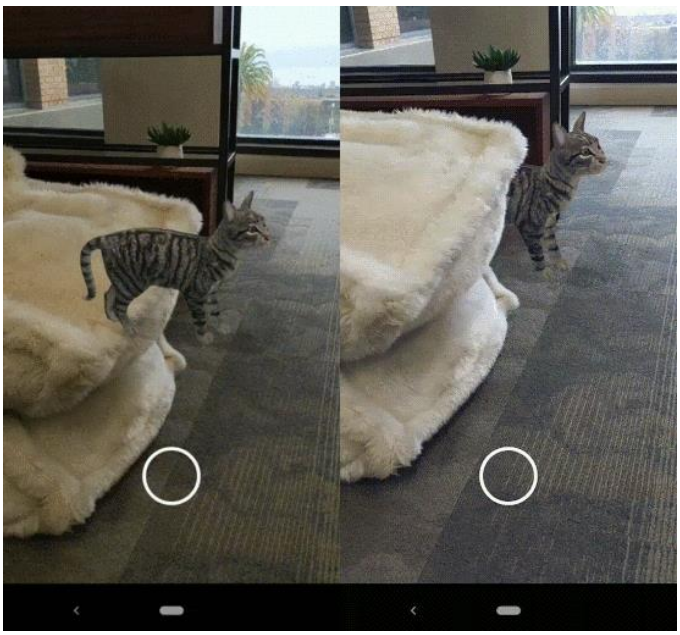

Figura 15. Un gatto virtuale senza occlusione a sinistra e con occlusione a destra con Depth API.

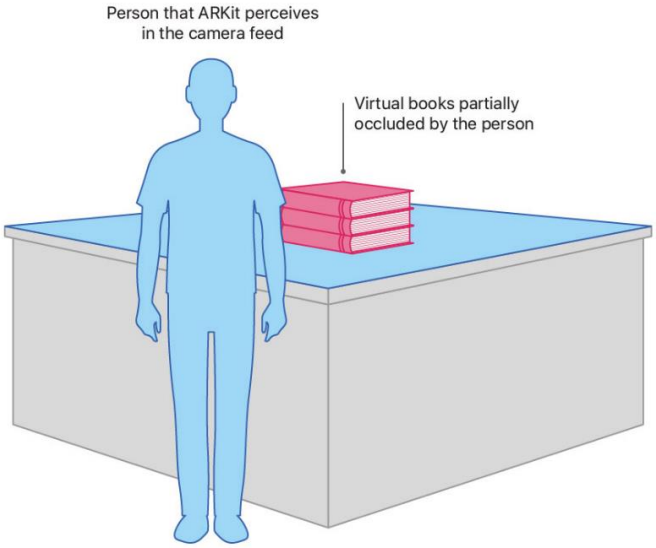

Figura 16. Infografica sul funzionamento di People Occlusion su ARKit.

3.3 **L'interazione nell'AR**

Nella piattaforma mobile, l'interazione tra l'utente e l'oggetto virtuale avviene indirettamente attraverso lo schermo del proprio dispositivo. Questo ci garantisce che un utente possa interagire molto limitatamente con gli oggetti virtuali presenti nella scena; considerando che gli HoloLens permettono una interazione real-time con l'intera scena virtuale.

3.3.1 *L'interazione con gli HoloLens*

HoloLens introduce le interazioni istintive, ovvero una filosofia coniata da Microsoft per la piattaforma di MR; questo traguardo è stato raggiunto dall'impiego di sensori come il tracciamento di mani e occhi, presenti nel dispositivo, e alcune tecnologie di input come il linguaggio naturale, per combinarli in modelli multimodali senza soluzione di continuità. Microsoft per realizzare questa funzionalità ha impiegato un singolo modello di interazione end-to-end per garantire la semplicità nell'uso del dispositivo.

Il primo approccio prevede la manipolazione diretta, quindi, il contatto diretto con i modelli virtuali usando le mani. L'idea alla base di questo approccio è che gli oggetti si comportano esattamente come farebbero nel mondo reale; per esempio, i pulsanti vengono attivati premendoli, gli oggetti possono essere raccolti afferrandoli e il contenuto 2D si comporta come un touchscreen virtuale.

Sugli HoloLens, le mani vengono riconosciute e interpretate come modelli scheletrici sinistro o destro; per implementare l'idea di toccare gli ologrammi direttamente con le mani, idealmente, vengono usati cinque collider attaccati alle punte delle cinque falangi di ciascun modello scheletrico della mano. Però, non si consiglia questo modello dato che c'è la probabilità di causare collisioni imprevedibili con gli oggetti virtuali; di conseguenza, viene utilizzato di norma un solo collider associato all'indice.

Il dispositivo consente manipolare gli oggetti olografici 3D applicando un rettangolo di selezione a ciascun oggetto 3D. Il rettangolo fornisce una migliore percezione della profondità attraverso il suo shader di prossimità; la manipolazione prevede: lo spostamento, la rotazione e la possibilità di scalare l'oggetto con dei semplici gesti.

Con gli HoloLens è possibile afferrare gli oggetti e spostarli; se sono oggetti piccoli, l'utente dovrà afferrarlo usando solo due dita, invece, se l'oggetto è piuttosto grande dovrà utilizzare l'intero palmo della mano ed eseguire la presa con le cinque dita.

Un problema nell'interazione è la possibilità di innescare involontariamente un ologramma; per cui, HoloLens utilizza il tracciamento oculare che aiuta a identificare meglio qual è l'intento dell'utente.

Questa funzione serve per ridurre l'attivazione involontaria di una risposta di manipolazione, ma anche per migliorare l'identificazione degli ologrammi da afferrare o colpire poiché il punto d'intersezione potrebbe non essere chiaro dalla prospettiva dell'utente. Il tracciamento degli occhi, quindi, migliora l'accuratezza del gesto. [31]

Infine, come funzionalità saliente degli HoloLens c'è l'input vocale che permette di comandare direttamente un ologramma senza dover usare i gesti delle mani; quando si utilizzano i comandi vocali, lo sguardo viene generalmente utilizzato come meccanismo di puntamento, sia con un cursore sia per indirizzare implicitamente il comando su un'applicazione che si sta guardando. [32]

3.3.2 *L'interazione sugli smartphone*

L'unica interazione possibile negli smartphone, come già stato accennato, è attraverso lo schermo, indirettamente.

Google ARCore fornisce, in questo caso, delle funzioni che riconoscono le varie gesture come il Drag&Drop o la rotazione di oggetti virtuali; infatti come viene mostrato nella Figura 17 si può notare che questa funzione gestisce il Drop dell'oggetto sulla superficie, dove l'utente ha scelto la nuova posizione dell'ancoraggio. In questo modo, l'utente può decidere di traslare l'oggetto nell'ambiente circostante.

```
GameObject oldAnchor = transform.parent.gameObject;

Pose desiredPose = new Pose(m_DesiredAnchorPosition, m_LastHit.Pose.rotation);

Vector3 desiredLocalPosition =
    transform.parent.InverseTransformPoint(desiredPose.position);

if (desiredLocalPosition.magnitude > MaxTranslationDistance)
{
    desiredLocalPosition = desiredLocalPosition.normalized * MaxTranslationDistance;
}

desiredPose.position = transform.parent.TransformPoint(desiredLocalPosition);

Anchor newAnchor = m_LastHit.Trackable.CreateAnchor(desiredPose);
transform.parent = newAnchor.transform;

Destroy(oldAnchor);

m_DesiredLocalPosition = Vector3.zero;

// Ruota se la direzione del piano è cambiata.
if (((desiredPose.rotation * Vector3.up) - transform.up).magnitude > k_DiffThreshold)
{
    m_DesiredRotation = desiredPose.rotation;
}
else
{
    m_DesiredRotation = transform.rotation;
}

// Assicura che la posizione sia stata aggiornata l'ultima volta.
m_IsActive = true;
```

Figura 17. Esempio di codice che gestisce lo stato di Drop dell'oggetto virtuale con Google ARCore.

Ci sono aziende che stanno cercando modi per interagire con gli oggetti virtuali come accade in HoloLens.

Manomotion, un'azienda specializzata in software di CV, ha sviluppato un SDK che permette l'interazione in tempo reale con tali oggetti tramite l'uso delle mani. Per risolvere ciò, hanno implementato una soluzione per riconoscere i vari gesti della mano; infatti essa ha 27 gradi di libertà (DoF)[11]: quattro per dito di cui tre per l'estensione e la flessione e uno per l'abduzione e l'adduzione; il pollice ha cinque

[11] I gradi di libertà di un punto materiale sono variabili indipendenti necessarie per determinare univocamente la sua posizione nello spazio. [33]

DoF e i rimanenti sei DoF per la rotazione e la traslazione del polso. Questo rende la registrazione video del movimento delle mani e dita estremamente impegnativa a causa dell'elevato numero di DoF della cinematica delle mani; soprattutto, è ancor più complicato negli smartphone a causa della potenza limitata e per i calcoli costosi.

Nella Figura 18 viene mostrato una soluzione schematizzata e adottata da Manomotion, la sequenza di immagini della query catturata dai sensori verrà analizzata per segmentare la mano e le dita dell'utente. Gli algoritmi di analisi delle immagini come la rimozione dello sfondo, la classificazione e il rilevamento dei features point vengono utilizzati per rilevare le mani/dita.

Gli algoritmi esistenti per il tracciamento delle mani e riconoscimento dei gesti possono essere raggruppati in due categorie: approcci basati sull'aspetto e approcci basati sul modello della mano 3D. I primi si basano su un confronto diretto dei gesti con le caratteristiche delle immagini 2D.

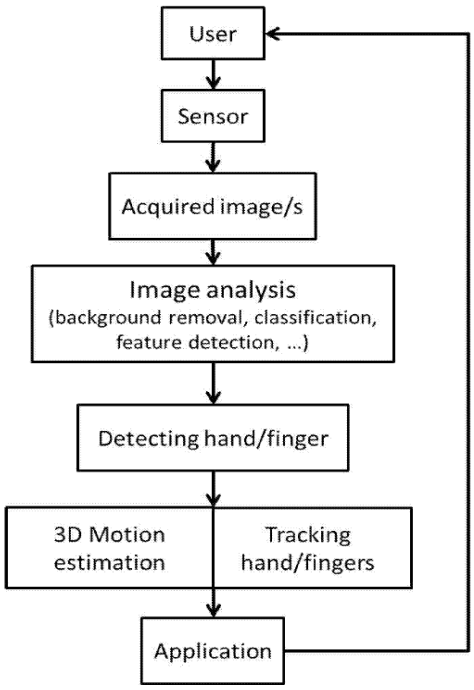

Figura 18. Diagramma di flusso che illustra schematicamente un metodo per tracciare e riconoscere i gesti.

Le principali caratteristiche delle immagini per riconoscere le mani includono i suoi colori e le loro forme. Lo svantaggio degli approcci basati sulle caratteristiche è che generalmente viene richiesta una segmentazione pulita delle immagini per estrarre le funzioni manuali. Spesso è difficile trovare le particolarità della mano locale a causa dell'occlusione e sono necessari alcuni tipi di euristica per gestire la grande varietà di gesti manuali. Invece di utilizzare le caratteristiche nelle immagini 2D

per rilevare la mano, gli approcci basati sul modello 3D utilizzano un modello tridimensionale per definire le sue varie pose. Il problema principale è che la mano 3D è un modello complesso deformabile a 27 DoF. Per definire tutte le sue pose sotto diverse viste, è quindi necessario un database molto grande; la corrispondenza delle immagini delle query dall'ingresso video con tutte le immagini manuali nel database è dispendiosa in termini di tempo e computazionalmente onerosa.

Manomotion per riconoscere un gesto 3D viene eseguito in un dispositivo, come uno smartphone, attraverso un database di immagini gestuali; quindi il dispositivo comunica con un sensore adattato per catturare un'immagine della mano 3D.

Il database dei gesti comprende caratteristiche indicizzabili di immagini di gesti normalizzate; infatti, in questo database vengono indicizzate queste qualità per reperire facilmente l'immagine corrispondente.

Tali particolarità comprendono una posizione e un orientamento per ciascun pixel dell'immagine normalizzata. Il metodo prevede l'acquisizione del gesto 3D tramite il sensore e la normalizzazione viene acquisita in conformità con le immagini gestuali normalizzate del database.

Quindi, la metodologia e il sistema proposti si basano sulla ricerca di una corrispondenza in un database gestuale estremamente ampio che include vari tipi di gesti delle mani con tutte le possibili variazioni nella rotazione e nel posizionamento: il database è composto da milioni di immagini di gesti dove sono annotati gli specifici parametri del movimento 3D, ovvero tre parametri di posizioni e tre di orientamento.

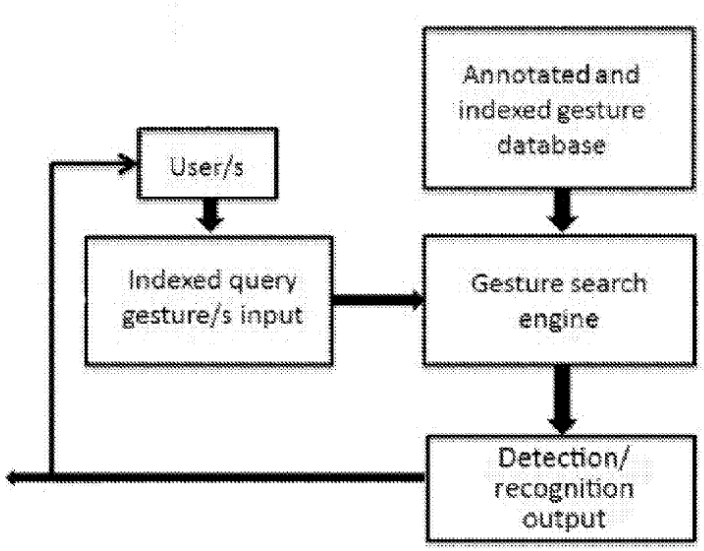

Figura 19. Diagramma che illustra il sistema dell'API.

La Figura 19 rappresenta il sistema adottato dall'API sviluppato da Manomotion che comprende quattro componenti principali: il database indicizzato pre-elaborato, l'unità di elaborazione che riceve una query del gesto, un motore di ricerca di gesti in tempo reale che recupera automaticamente la migliore corrispondenza dal database e, infine, il livello di interfaccia che riceve il risultato del motore e lo applica all'applicazione in corso. [34]

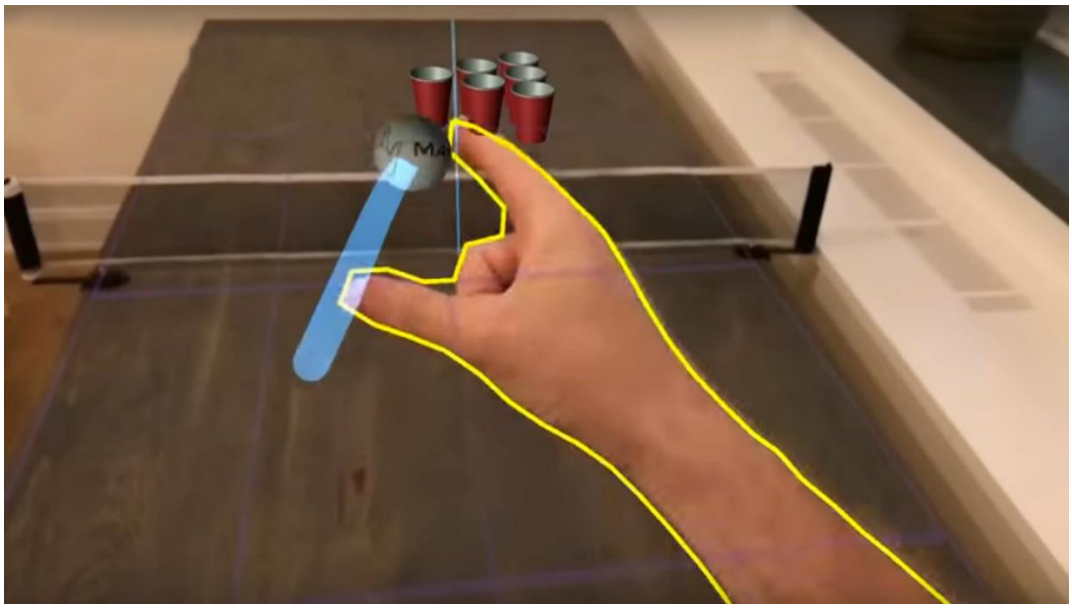

Figura 20. Demo di interazione con gli oggetti virtuali tramite l'SDK Manomotion. (dal sito di Manomotion)

CONCLUSIONI

Questa tesi ha potuto definire un quadro della situazione odierna della realtà aumentata in ambito mobile, che è sempre in continuo sviluppo. Si è potuto notare, inoltre, che Microsoft HoloLens, un dispositivo costruito per mantenere alta la qualità della realtà mista; pone, in questo modo, un approccio molto più accurato ed efficiente rispetto ad uno smartphone.

In questa ricerca si è potuto definire le principali sfide che affliggono il mondo mobile per un impiego intenso della realtà aumentata e come si può notare, non tutti i problemi citati nell'ultimo capitolo sono di natura tecnica.

La panoramica delle criticità della realtà aumentata su smartphone ha denotato il loro ostacolo per un maggior utilizzo, al di fuori nel campo dell'intrattenimento e quello del marketing.

Tuttavia, negli ultimi anni le tecnologie hanno avuto un rapido sviluppo migliorando sia il lato hardware che software degli smartphone; questa tendenza è in continua espansione, come è stato sottolineato nell'ultimo capitolo, soprattutto, con Depth API di Google o con l'SDK di Manomotion.

Su quest'ultimo, particolarmente, ci sono ancora molte complicazioni sul riconoscimento della mano tramite la camera RGB che hanno accentuato le enormi difficoltà sul riconoscimento dei gesti e provvisti di una grande latenza; però questo potrebbe essere facilmente risolto utilizzando una camera IR.

In ogni caso, questo ha dimostrato che gli smartphone possono avvicinarsi, ma non eguagliare le funzionalità di un dispositivo in realtà mista come HoloLens; anche se, un ulteriore sviluppo e ricerca su queste tecnologie potrebbe portare dei risultati inaspettati.

Si sostiene, quindi, che la diversità dei problemi dimostra la complessità di sviluppare la realtà aumentata sugli smartphone, rendendolo un degno campo per la futura ricerca.

Una raccomandazione per indagini future potrebbe essere quella di realizzare uno studio approfondito su un aspetto critico individuato in questa tesi come, per esempio, l'interazione e il riconoscimento dei gesti con gli oggetti virtuali.

BIBLIOGRAFIA E SITOGRAFIA

[1] "Augmented Reality Market by Offering (Hardware (Sensor, Displays & Projectors, Cameras), Software), Device Type (Head-mounted, Head-up), Application (Enterprise, Consumer, Commercial, Healthcare, Automotive), and Region - Global Forecast to 2024", Markets&Markets, 2020.

[2] https://www.ikea.com/au/en/apps/IKEAPlace.html

[3] https://it.wikipedia.org/wiki/Realt%C3%A0_aumentata

[4] https://en.wikipedia.org/wiki/Augmented_reality

[5] https://en.wikipedia.org/wiki/Reality%E2%80%93virtuality_continuum

[6] https://en.wikipedia.org/wiki/Mixed_reality

[7] MILGRAM Paul, TAKEMURA Haruo, AKIRA Utsumi e KISHINO Fumio, "Augmented Reality: A class of displays on the reality-virtuality continuum", ATR Communication Systems Research Laboratories, 1994

[8] https://it.wikipedia.org/wiki/Unit%C3%A0_di_misura_inerziale

[9] https://en.wikipedia.org/wiki/Visual_odometry#Visual_Inertial_Odometry

[10] https://en.wikipedia.org/wiki/Feature_(computer_vision)

[11] YOUSIF Khalid, BAB-HADIASHAR Alireza e HOSEINNEZHAD Reza, "An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics", RMIT University, 2015

[12] https://it.wikipedia.org/wiki/Fotocamera_stereoscopica

[13] https://developers.google.com/ar

[14] https://en.wikipedia.org/wiki/Visual_appearance

[15] NERURKAR Esha, LYNEN Simon e ZHAO Sheng, "System and method for concurrent odometry and mapping", Google Inc., 2017

[16] YANG Liu, HAIWEI Dong, LONGYU Zhang, ABDULMOTALEB El Saddik, "Technical Evaluation of HoloLens for Multimedia: A First Look", University of Ottawa, 2018

[17] https://it.wikipedia.org/wiki/Telecamera_a_tempo_di_volo

[18] https://en.wikipedia.org/wiki/Iterative_closest_point

[19] https://en.wikipedia.org/wiki/Kinect

[20] https://it.wikipedia.org/wiki/Filtro_bilaterale

[21] NEWCOMBE Richard A., IZADI Shahram, HILLIGES Otmar, MOLYNEAUX David, KIM David, DAVISON Andrew J., KOHLI Pushmeet, SHOTTON Jamie, HODGES Steve, FITZGIBBON Andrew, "KinectFusion: Real-Time Dense Surface Mapping and Tracking", Microsoft Research, 2011

[22] ARTH Clemens, SCHMALSTIEG Dieter, "Challenges of Large-Scale Augmented Reality on Smartphones", Graz University of Technology, 2011

[23] https://it.wikipedia.org/wiki/Dispositivo_ad_accoppiamento_di_carica

[24] https://it.wikipedia.org/wiki/Lag_(informatica)

[25] https://it.wikipedia.org/wiki/5G

[26] PATKAR Raviraj S., SINGH S. Pratap, BIRJE Swati V., "Marker Based Augmented Reality Using Android OS", Pune University e Mumbai University, 2013

[27] https://developers.google.com/ar/develop/c/augmented-images

[28] https://developers.google.com/ar/develop/unity/light-estimation

[29] https://it.wikipedia.org/wiki/Quaternione

[30] https://www.eizo.it/nozioni-pratiche/know-how-sul-tema-monitor/informazioni-sullhdr-di-che-cosa-si-tratta/

[31] https://xinreality.com/wiki/Occlusion

[32] https://developers.googleblog.com/2019/12/blending-realities-with-arcore-depth-api.html

[33] https://developer.apple.com/documentation/arkit/occluding_virtual_content_

with_people

[34] https://docs.microsoft.com/it-it/windows/mixed-reality/direct-manipulation

[35] https://docs.microsoft.com/it-it/windows/mixed-reality/voice-input

[36] https://it.wikipedia.org/wiki/Grado_di_libertà_(meccanica_classica)

[37] Shahrouz YOUSEFI, Haibo LI, Farid ABEDAN KONDORI, "Real-time 3D gesture recognition and tracking system for mobile devices", Manomotion, 2016

RINGRAZIAMENTI

Ed eccomi giunto alla fine di questa tesi e alla fine di un percorso di studi costellato di successi e altrettanti fallimenti.
Per raggiungere questo agognato traguardo sono caduto tante volte in questi anni universitari, ma stringendo forte i denti, mi sono sempre rialzato continuando la mia corsa verso la laurea, grazie soprattutto ai miei cari che mi hanno sempre appoggiato durante il mio percorso di studi. Infatti, vorrei dedicare queste ultime pagine a tutte quelle persone che mi hanno aiutato in un modo o nell'altro ad arrivare fin qui.
Prima di tutto, ringrazio il mio relatore Antonio Cisternino per la sua disponibilità e i suggerimenti durante il periodo della stesura della tesi.
Ringrazio poi la mia famiglia che mi ha dato la possibilità di continuare gli studi e di avere un futuro migliore, sostenendomi sia moralmente che economicamente, per questo ne sono molto grato.
Un ringraziamento va a mio padre Ruggero, che oltre al sostegno economico, mi ha dispensato consigli in questi anni; inoltre, mi ha fatto viaggiare tra le varie città europee facendomi scoprire il mondo e, soprattutto, dandomi la possibilità di staccare la spina dopo un'intensa sessione di esami.
Di conseguenza, non posso che ringraziare anche sua moglie, Sara. Anche se mi ha conosciuto quando ero in età adolescenziale mi ha supportato e mi fa piacere che faccia parte di questa famiglia, allargata!
Ringrazio il mio patrigno Andrea che mi ha trattato sempre come suo figlio fin da piccolo e ha offerto il suo sostegno economico assieme a mio padre per aiutarmi a raggiungere questo obiettivo.
Un grazie va a mio fratello Alessandro che mi ha fornito un sacco di spunti e per aver sopportato i miei sproloqui per le mie idee contorte.
Ma questa tesi la voglio dedicare a mia madre Raffaella che mi è sempre rimasta vicina e che mi ha sempre tirato su di morale quando mi vedeva triste o preoccupato durante ogni sessione di esame.

Una donna forte che mi ha cresciuto insegnandomi certi valori e guidandomi fino a questo momento.
Ora vorrei ringraziare i miei amici e colleghi di università che mi hanno alleggerito le giornate passate nel dipartimento e a mensa e ringraziarli per avermi strappato qualche risata durante il periodo delle lezioni. Dunque, ringrazio in ordine sparso Picci, Ciccio, Gersi, Corna, Marco, Filippo, Giovanni, Gabriele, Sergio, Lorenzo e Davide.
Vorrei, inoltre, ringraziare alcune persone che ho conosciuto su un forum dove ho potuto condividere alcuni miei interessi e creare un gruppo un po' folle ma divertente. Anche se non ci siamo ancora conosciuti di persona, questo gruppo mi ha colorato le mie grigie giornate da pendolare durante il periodo delle lezioni.
Ringrazio Lino, che mi ha dato consigli e dritte sulla grafica 3D.
Un grazie a Gianmarco per avermi dato alcuni suggerimenti per studiare fisica.
E l'ultimo ringraziamento, ma non meno importante, va a Kevin che, con la sua passione per la scrittura creativa, mi ha spronato a migliorare la mia abilità di scrittura.
Infine, ringrazio tutte quelle persone che non ho nominato esplicitamente ma che mi hanno aiutato a crescere e hanno lasciato un'impronta nella mia vita. Grazie!