

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA
CORSO DI LAUREA IN INGEGNERIA ELETTRONICA,
INFORMATICA E TELECOMUNICAZIONI

PROGETTAZIONE E SVILUPPO DI
APPLICAZIONI DI AUGMENTED REALITY
INTERATTIVE

Elaborata nel corso di: Sistemi Operativi

Tesi di Laurea di:
FABIO GAUDENZI

Relatore:
Prof. ALESSANDRO RICCI

ANNO ACCADEMICO 2012–2013
SESSIONE III

PAROLE CHIAVE

Augmented Reality

Mixed Reality

Sviluppo Software

SDK

A Federica e a tutti quelli che mi hanno aiutato in
questa impresa

Indice

Introduzione	ix
1 Introduzione alla Realtà Aumentata	1
1.1 Breve Storia della Realtà Aumentata	2
1.2 Wearable Device	4
1.3 Campi di Applicazione	5
2 Tracking, Registration, Visualization	9
2.1 Tracking e Registration	10
2.1.1 Tecniche di Tracking Sensor-Based	10
2.1.2 Tecniche di Tracking Vision-Based	11
2.1.3 Hybrid Tracking	13
2.2 Visualization	13
3 Interazione	17
3.1 Tecniche di interazione e UI	18
3.1.1 User Interfaces	18
3.1.2 Collaborative AR: applicazioni multi utente	21
3.2 Scenari Futuri, Blended Reality Space	23
4 Architettura di un'Applicazione AR e Pattern per la Progettazione	25
4.1 Architettura di un'Applicazione AR	25
4.1.1 Sottosistemi dell'Architettura	25
4.2 Pattern per la Progettazione dei Sottosistemi	27
4.2.1 Application	27
4.2.2 User Input	28
4.2.3 Tracking	29

4.2.4	World Model	30
4.2.5	Context	31
4.2.6	Output e Rendering	32
5	Strumenti Per Lo Sviluppo e per la Pubblicazione di Appli- cazioni AR	35
5.1	ARToolkit	35
5.2	Tipica struttura di un Framework AR	37
5.3	Architettura del sistema offerto dai principali provider	39
5.4	Layar	41
5.4.1	SDK API	41
5.4.2	Creator, Content-based API e AR Browser	41
5.5	Metaio	42
5.5.1	SDK API	43
5.5.2	Creator, Content-based API e AR Browser	45
5.5.3	CloudPlugin e AREL Interpreter	45
5.6	Vuforia e Wikitude	45
5.6.1	Vuforia	45
5.6.2	Wikitude	46
5.7	Confronto	46
	Conclusioni	49

Introduzione

Nel 2011, circa 5 milioni di persone usavano regolarmente applicativi mobili basati su qualche forma di realtà aumentata. Secondo alcune previsioni [1] entro il 2020 servizi che sfruttano questa tecnologia avranno una base di utenti che rasenta un settimo della popolazione mondiale, rendendo di fatto la realtà aumentata l'ottavo mezzo di comunicazione di massa dopo la stampa, l'audio registrato, il cinema, la radio, la televisione, internet ed infine i servizi multimediali mobile. La tesi si propone di illustrare le caratteristiche di questa tecnologia distinguendo varie tecniche e strumenti a disposizione degli sviluppatori per costruire l'esperienza di *augmented reality*. Entrando nel merito dello sviluppo software si analizza una architettura di riferimento proposta per sistemi di realtà aumentata e pattern per lo sviluppo dei diversi componenti. Infine vengono confrontati gli SDK oggi più diffusi cercando di evidenziarne caratteristiche comuni e peculiarità.

Nota Introduttiva Siccome al momento non è ancora presente una terminologia standard per descrivere le varie parti dell'esperienza di realtà aumentata, in questo documento si fa riferimento a quella definita dalla *community ARStandards* [7], impegnata in collaborazione con rappresentanti delle aziende del settore a proporre un set di termini, formati e soluzioni comuni alle applicazioni di *augmented reality*.

Capitolo 1

Introduzione alla Realtà Aumentata

Il termine *augmented reality* fu coniato dal ricercatore Tom Caudell [8] all'inizio degli anni '90 per descrivere i display usati dai tecnici dell'aviazione civile i quali sovrapponevano informazioni digitali alla visualizzazione del mondo reale. La definizione formale in materia di *computer science* estende il concetto pur mantenendo l'idea di base: la realtà aumentata consiste nel sovrapporre a dati provenienti da un ambiente reale informazioni digitali sotto forma di grafica sovrainposta, suoni o altre forme di "potenziamento" sensoriale, il tutto eseguito in tempo reale.

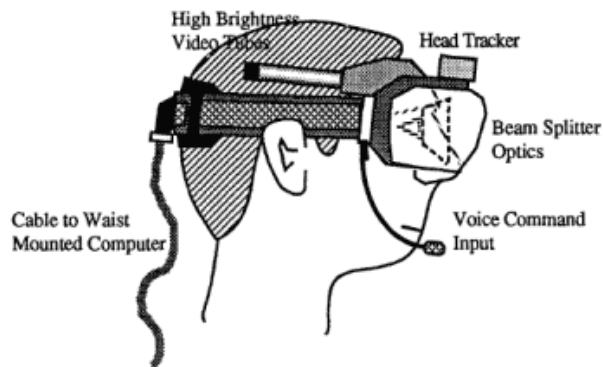


Figura 1.1: Uno schema concettuale di un HMD per la realtà aumentata

1.1 Breve Storia della Realtà Aumentata

Al di là della terminologia, la sovrainpressione di immagini al mondo reale è un concetto non nuovo al mondo della ricerca, nel 1968 infatti, Ivan Sutherland creò il primo sistema a realtà aumentata [33], costituito da un HMD (*Head Mounted Display*) la cui posizione era tracciata da un sensore meccanico o basato su ultrasuoni: questa macchina permetteva di sovrapporre all'ambiente semplici contorni digitali generando un effetto tridimensionale basato sui movimenti dell'utente.

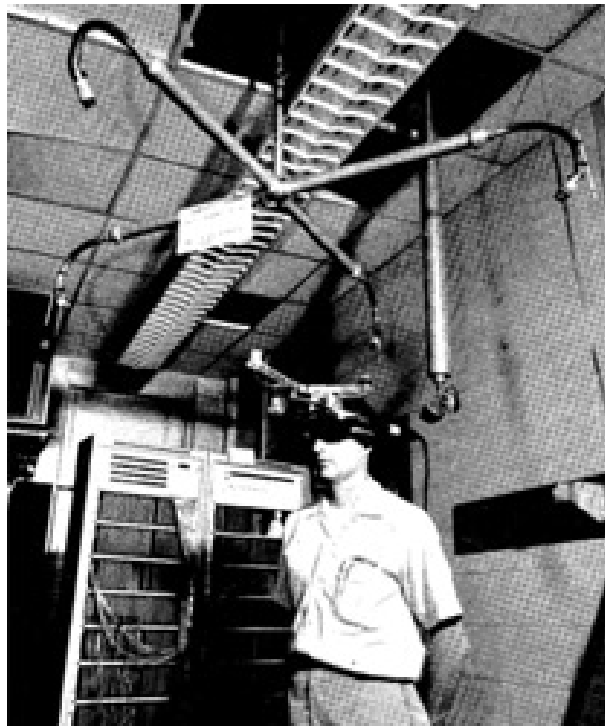


Figura 1.2: Il primo prototipo di HMD

Nel 1994, Paul Milgram e Fumio Kishino [24] introducono il concetto di *mixed reality* come “[...] qualunque cosa tra gli estremi del *Virtuality Continuum*”, dove il *Virtuality Continuum* si estende tra il mondo reale e quello virtuale, passando per la realtà aumentata e la *augmented virtuality*,

rispettivamente elementi virtuali sovrapposti alla realtà ed elementi reali sovrapposti ad un ambiente virtuale.

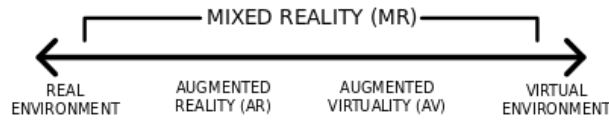


Figura 1.3: Il Reality-Virtuality Continuum

Nella loro pubblicazione, Milgram e Kishino analizzano la relazione tra *augmented* e *virtual reality*: per realtà virtuale si intende una situazione in cui l'utente è totalmente immerso in un mondo completamente sintetico che può o meno avere caratteristiche comuni con il mondo reale (ad esempio può non sottostare alle leggi della fisica) mentre nel caso della realtà aumentata gli elementi che appartengono al livello virtuale sono in qualche modo legati ad una o più proprietà del mondo reale, ad esempio un oggetto visualizzato su una superficie può essere vincolato ad essa. Due anni dopo, Jun Rekimoto [29] introduce il concetto di marker, ovvero un pattern stampato identificabile da una telecamera che permetteva entro certi limiti di tracciarne i movimenti con sei gradi di libertà (6DoF)¹: questo è alla base della cosiddetta metodologia di tracciamento *marker-based* approfondita più avanti in questa tesi. La prima indagine sullo stato della ricerca interamente dedicata alla realtà aumentata risale al 1997 quando Ronald Azuma presenta "A survey of augmented reality"[4] dove propone una definizione formale di realtà aumentata specificata in tre caratteristiche essenziali:

- Combina reale e virtuale;
- È interattiva in tempo reale;
- *Registered in 3D*

L'ultimo punto si riferisce al tipico problema dell'allineamento degli elementi sovrapposti con l'ambiente reale in base al punto di vista dell'osservatore, spesso detto *the Registration problem*. La definizione di Azuma da

¹Si riferisce al movimento nello spazio tridimensionale di un corpo rigido, ovvero la traslazione e rotazione rispetto agli assi perpendicolari di un sistema di riferimento (traslazione nelle tre direzioni, imbardata, beccheggio e rollio).

una parte permette di considerare realtà aumentata le tecnologie che non prevedono l'uso di un HMD o di particolari strumentazioni, dall'altra permette di escludere esperimenti simili che però non ricadono completamente nei vincoli posti dal ricercatore: l'esempio fatto riguarda i film con immagini in computer grafica (2-D o 3-D) sovrainposte a quelle reali, i quali non sono né interattivi né in tempo reale. Durante i primi anni duemila, osserviamo la diffusione delle tecnologie e degli strumenti per sviluppatori oggi alla base di quasi tutti gli applicativi mobile AR a partire dal primo telefono cellulare con fotocamera passando per il protocollo Wi-Fi fino a sensori di vario genere a basso consumo energetico (ad esempio GPS e giroscopi).

1.2 Wearable Device

*There's difference between technology you can wear and technology DESIGNED to be worn*¹

La tecnologia AR necessita una piattaforma hardware che fornisca un'ampia collezione di sensori e funzionalità: per questo nel corso degli anni il principale ostacolo alla diffusione di applicazioni mobile di questo genere è stata, oltre all'elevata capacità computazionale richiesta dal software, la scarsa portabilità dei dispositivi capaci di garantire una buona qualità dell'esperienza. Di recente abbiamo osservato la commercializzazione di componenti ad alta precisione ma di dimensioni ridotte: l'integrazione di prodotti come videocamere 3D con sensori di profondità e microproiettori ad alta definizione ha portato alla nascita di numerosi dispositivi con grandi potenzialità in campo di *augmented reality* classificabili nella categoria dei *wearable device*. Con il termine *wearable technology* si definiscono computer ed altri dispositivi elettronici integrati in vestiti o accessori indossabili. I *wearable device* sono strettamente legati al campo dell' *ubiquitous computing* che mira all'integrare elementi dotati di capacità computazionale in oggetti di uso quotidiano. Recentemente sono stati presentati diversi esempi di dispositivi *wearable* progettati per supportare una qualche forma di *mixed reality*: i primi *smartglass* sono un chiaro esempio della direzione in cui lo sviluppo di questa tecnologia è orientato. Gli *smartglass* sono una tipologia di *wearable*

¹C'è una differenza tra tecnologia che puoi indossare e tecnologia disegnata per essere indossata

computer che consiste in un OHMD (*Optical Head Mounted Display*) o in un paio di occhiali con, a seconda dello scopo, un HUD (*Heads-up Display*) o un visore trasparente per la realtà aumentata. Epson ha recentemente reso disponibile al pubblico un modello di *smartglass* progettati specificamente per fornire esperienze di *augmented reality* (*Moverio BT-200*) dotati di un display semitrasparente che copre l'intero campo visivo dell'utente ed una serie di sensori integrati. La nascita di hardware dedicato e la già presente diffusione di dispositivi mobili in grado di gestire applicazioni e presentare contenuto AR apre ad un grande numero di applicazioni per questa tecnologia.



Figura 1.4: Epson Moverio BT-200 - Fonte: Epson

1.3 Campi di Applicazione

I principali campi di applicazione della realtà aumentata sono oggi la pubblicità, l'intrattenimento, la navigazione assistita di città e luoghi di interesse e l'ingegneria industriale: è possibile trovare inoltre proposte di applicazione in campo medico e nell'educazione.

Militare La tecnologia AR è da tempo alla base di strumentazioni militari legate soprattutto a visori tattici che permettono una visualizzazione efficace di obiettivi ed informazioni: applicazioni del genere sono spesso utilizzate nelle simulazioni per l'addestramento ed in alcuni casi direttamente sul campo [21].

Pubblicità La maggior parte delle soluzioni commercialmente più diffuse sfruttano il concetto di realtà aumentata per aggiungere contenuto virtuale, a partire da modelli 3D fino a video ed applicazioni interattive, a materiale stampato come riviste e manifesti pubblicitari fornendo alle aziende interessate una serie di strumenti per la creazione di contenuto che non richiede esperienza nello sviluppo software.



Figura 1.5: Un'applicazione AR permette di visualizzare informazioni e oggetti 3D direttamente sulla pagina di una rivista. Fonte: Metaio

Progettazione In campo industriale, applicazioni già commercializzate sfruttano visualizzazioni AR di modelli e prototipi per facilitare il processo creativo e ridurre i tempi di produzione[23]; con la stessa tecnica è possibile ad esempio osservare il modello di un edificio in costruzione direttamente sovrapposto al cantiere[23]. Le potenzialità della augmented reality tuttavia superano la semplice, seppur innovativa, capacità di visualizzazione di contenuto: ad esempio nel processo di assemblaggio e manutenzione una applicazione AR può permettere la sovrapposizione di istruzioni direttamente sull'oggetto in questione ed allo stesso tempo fornire un efficace mezzo di comunicazione con un esperto il quale può assistere in tempo reale all'operazione [25]. Applicazioni di *augmented reality* garantiscono quindi vantaggi prima impensabili sfruttando tecnologie precedenti [20]: prima di tutto la visualizzazione aumentata delle informazioni permette all'utente di non isolarsi dalla realtà e di spostare la propria attenzione su attività di natura

diversa (ad esempio seguire un manuale di istruzioni virtuale) senza particolari sforzi o passaggi; inoltre, riprendendo la definizione di Azuma, l'aspetto dell'interazione in tempo reale con elementi virtuali è di vitale importanza in applicazioni di questa tecnologia: la possibilità di manipolare il contenuto aumentato con gesti ed oggetti fisici è molto più intuitiva e naturale della tradizionale coppia mouse e tastiera.

Navigazione e Turismo L'uso della realtà aumentata per sovrapporre indicazioni stradali o evidenziare edifici e luoghi di interesse è una funzionalità già diffusa in applicazioni mobile sul mercato: inoltre grandi aziende automobilistiche come KIA e Mercedes stanno investendo per integrare la tecnologia di visualizzazione AR nei sistemi di navigazione integrati sulle automobili [19].

Medicina In campo medico esistono sistemi sperimentali che sfruttano una forma di realtà aumentata interattiva nella terapia di esposizione per il trattamento di disturbi d'ansia e fobie [13]: grazie ad un proiettore ed a sensori di posizione e rilevamento dei movimenti del paziente, i medici sono in grado di creare in sicurezza un'esperienza di AR in cui il paziente viene immerso visualizzando in maniera controllata l'oggetto del disturbo.

Educazione e Cultura La realtà aumentata ha svariate possibili applicazioni nel campo dell'intrattenimento educativo: Lai e Wang [20] propongono l'utilizzo della tecnologia per creare esperienze interattive per l'insegnamento, eliminando il tradizionale vincolo dello schermo di un PC. Da tempo sistemi basati su realtà aumentata e *mixed reality* trovano applicazioni concrete nei musei e nei siti di interesse culturale: il progetto ARTSense [11] mira a realizzare guide per i visitatori che reagiscano dinamicamente alla posizione, ai gesti ed allo sguardo dell'utente inserendo elementi visivi o tracce audio per migliorare l'esperienza della visita.

Capitolo 2

Tracking, Registration, Visualization

L'esperienza di *augmented reality* è realizzabile partendo da alcuni requisiti: prima di tutto occorre una combinazione di hardware e software in grado di generare il contenuto virtuale da sovrapporre, ad esempio un oggetto in 3-D od un particolare suono; è fondamentale inoltre la scelta delle cosiddette tecniche di *tracking*, il modo in cui viene rilevata e processata la posizione dell'osservatore, e di *registration*, ovvero la tecnica e gli strumenti per assicurarsi un buon livello di allineamento con i dati provenienti dall'ambiente reale. Ugualmente importante è l'aspetto della visualizzazione delle informazioni e dei dispositivi oggi usati nelle applicazioni AR, approfondito più avanti in questo capitolo. Possiamo dividere il processo comune a tutte le applicazioni AR in tre principali passi: tracking, registration e visualization. Nella fase di tracking viene analizzato il flusso dati in ingresso al sistema, individuando eventuali punti di riferimento: a seconda della tecnica usata, questo coincide con calcolare la posizione di elementi visivi rispetto all'osservatore o tracciare i movimenti dell'utente grazie a sensori di varia natura. Una volta ricavate queste informazioni, nella fase di registration si modificano i parametri degli oggetti virtuali in modo da allinearli con gli elementi ricavati nella prima fase; nel caso più generale di mixed reality, questa seconda fase consiste nello stabilire in che modo le informazioni e gli elementi aumentati verranno inseriti nel contesto reale (ad esempio riprodurre un determinato suono quando l'utente si trova in una certa posizione). Durante la fase di visualizzazione, gli elementi grafici virtuali vengono visualizzati

dal dispositivo di output grazie a strumenti software come librerie grafiche e motori di rendering.

2.1 Tracking e Registration

La maggior parte delle tecnologie basate su *augmented reality* necessita la conoscenza in tempo reale della posizione degli oggetti virtuali da visualizzare, la posizione del display o, alternativamente, la posizione degli oggetti relativamente al display. La parte di sistema incaricata di ottenere le suddette informazioni è detta *tracking (sub)system*: esistono diverse tecniche per tracciare la posizione ed i movimenti dei vari elementi in gioco, procediamo a classificarle in base al tipo di hardware su cui si appoggiano.

2.1.1 Tecniche di Tracking Sensor-Based

Le tecniche di tracking dette *sensor-based* sfruttano sensori magnetici, acustici, inerziali, ottici e/o meccanici. Come vedremo, tutti questi hanno rispettivi vantaggi e svantaggi compensabili dalla combinazione di più sensori e di algoritmi in grado di filtrare i dati ottenuti per garantire una maggiore accuratezza.

GPS Tracking L'esempio più comune di *sensor-based tracking system* è il cosiddetto *Global Positioning System* (GPS): un ricevitore GPS sfrutta i segnali radio trasmessi da un determinato numero di satelliti in orbita attorno alla Terra per calcolare la sua posizione [14]. Ogni satellite trasmette in continuazione la propria posizione e la posizione degli altri satelliti nel sistema: i ricevitori sfruttano le variazioni del *time of arrival* (ToA) delle informazioni ricevute da almeno quattro satelliti per ricavare le proprie coordinate. La tecnologia, usata inizialmente dall'esercito americano, oggi viene sfruttata dai navigatori satellitari per mezzi di trasporto ed in campo mobile, spesso in combinazione con un sensore magnetico che provveda a rilevare l'orientamento del dispositivo. Il segnale GPS può risultare compromesso quando il ricevitore non ha una visuale chiara del numero minimo di satelliti necessari: spesso si hanno queste situazioni all'interno o nelle vicinanze di edifici i quali possono schermare le onde radio dei satelliti. [10]

Acoustical Tracking Nel 2004 Newman et al. [26] dimostrano come sensori ad ultrasuoni possano essere usati per rilevare la configurazione del sistema in un ambiente chiuso: gli *acoustical tracking system* ottengono la posizione di oggetti ai quali è collegato un trasmettitore di ultrasuoni calcolando l'intervallo di tempo tra la partenza dell'onda e l'arrivo (*time-of-flight*, TOF) oppure lo sfasamento tra l'onda proveniente dall'oggetto e quella di una sorgente con posizione nota (*phase-shift trackers*). Uno dei principali limiti di questa categoria di sensori è la frequenza di aggiornamento dipendente dalla propagazione del suono e dalle condizioni atmosferiche: questo problema può essere compensato con apparecchiature più complesse e rilevamenti più rapidi.

Magnetic Tracking I sensori magnetici sfruttano un campo magnetico (quello terrestre oppure uno generato artificialmente) per calcolare il proprio orientamento e, nei casi più complessi, la propria posizione [10]. La maggior parte dei sensori magnetici è in grado quindi di rilevare la posizione solo in un grado di libertà, rendendo necessaria quindi la presenza di altri componenti che compensino nell'operazione; inoltre la presenza di variazioni del campo magnetico, causate dalla presenza di apparecchiature elettroniche o conduttori metallici, può causare imprecisioni nel rilevamento dei dati.

Sensori di movimento inerziali (IMU) A differenza dei casi precedenti, i sensori inerziali non ottengono informazioni da fonti esterne, ma misurano caratteristiche relative del movimento come l'accelerazione: i più comuni sensori di movimento sono gli accelerometri ed i giroscopi.

2.1.2 Tecniche di Tracking Vision-Based

Una tipologia molto rilevante di tecniche di *tracking* sfrutta algoritmi di elaborazione delle immagini, ottenute da un flusso video in ingresso, per ricavare la posizione dell'osservatore. In materia di *computer vision*, possiamo distinguere queste tecniche in due categorie: *feature-based* e *model-based* [34].

Le cosiddette *feature-based technique* consistono nel trovare corrispondenze tra le immagini analizzate ed altre già memorizzate cercando di allinearle con una serie di trasformazioni 3-D: queste immagini campione possono

rappresentare caratteristiche facilmente riconoscibili dell'oggetto da rilevare, tipicamente spigoli o texture particolarmente definite. Una problematica comune a queste tecniche è l'elevato numero di falsi positivi: una possibile soluzione consiste nell'utilizzare simboli grafici appositamente realizzati in modo da risultare rilevabili ad un algoritmo di elaborazione delle immagini.

Marker-based Tracking Il concetto di *visual* o *fiducial marker* venne presentato da Jun Reikimoto [29] nel 1996: solitamente si tratta di un simbolo in due dimensioni che presenta caratteristiche ricorrenti, ad esempio un bordo quadrato contenente una texture bianca e nera il tutto asimmetrico dal punto di vista rotazionale. I marker possono essere riconosciuti usando una qualunque videocamera con risoluzione sufficiente, la quale è spesso integrata con il display in modo da permettere un buon allineamento con il punto di vista dell'osservatore anche senza utilizzare dispositivi *see-through* o strumentazioni ad-hoc. Solitamente l'elemento digitale viene sovrapposto al marker, nascondendo quest'ultimo alla vista dell'utente. Per funzionare in maniera corretta, il marker deve essere visualizzato entro certi limiti di inclinazione e luminosità altrimenti si possono verificare problematiche di riconoscimento che portano spesso a falsi positivi o al mancato rilevamento: per questo sono state proposte diverse soluzioni tra cui l'introduzione di marker tridimensionali in forma cubica [20] in grado di fornire almeno una superficie convenientemente orientata verso l'osservatore. Se da un lato le tecniche di *marker-based tracking* richiedono una qualche forma di preparazione, esse hanno il vantaggio di essere a basso costo ed utilizzabili sia al chiuso che all'aperto.

Natural Feature Tracking In alternativa all'uso dei *fiducial marker*, la posizione dell'osservatore può essere calcolata affidandosi al riconoscimento di caratteristiche naturali ricorrenti negli oggetti come linee, punti, spigoli o texture da cui il nome di *Natural Feature Tracking* (NFT). I sistemi AR che sfruttano queste tecniche richiedono una fase di calibrazione nella quale gli oggetti da tracciare vengono analizzati dal sistema da più angolazioni per memorizzarne le caratteristiche rilevanti al riconoscimento. La natura degli algoritmi NFT è sensibilmente più complessa rispetto a quelli *marker-based* esponendo il sistema a problemi di gestione di risorse computazionali ed al maggior rischio di falsi positivi o latenze.

Model-based Tracking Una categoria più recente di tecniche *vision-based* consiste nell'utilizzo di modelli CAD tridimensionali o di pattern 2-D per il processo di tracking. Il punto di forza di questa metodologia, detta *model-based tracking* si basa sul fatto che la conoscenza a priori della forma degli oggetti permette di prevedere in maniera accurata i movimenti nascosti all'osservatore. Gli algoritmi di rilevamento bordi giocano un ruolo importante in tutti gli approcci *vision-based*, in particolare per l'ultima categoria mostrata, dove i bordi e gli spigoli (edge) degli oggetti ne permettono l'identificazione.

Alcuni SDK descritti più avanti in questa tesi, offrono una particolare tecnica di *tracking vision-based* chiamata *Simultaneous Localization and Mapping* (SLAM): questa consiste nel rilevare in tempo reale caratteristiche particolari dell'ambiente e di memorizzarle in una mappa di punti tridimensionale costantemente aggiornata.

2.1.3 Hybrid Tracking

Per alcune applicazioni di AR o MR in generale, le tecniche di tracking derivate dalla *computer-vision* non riescono spesso ad assicurare da sole una buona qualità di tracciamento: per questo fin dai primi anni duemila sono state introdotte tecniche dette di *hybrid tracking* [5] che combinano *sensor-based* e *vision-based* per compensare alle mancanze di entrambe le tipologie. Un esempio di *hybrid tracking* considera un sistema che sfrutta il GPS correggendone le imprecisioni con un algoritmo di *computer-vision* [5]; un altro mostra come sia possibile compensare movimenti bruschi della videocamera o degli oggetti con sensori di movimento e sensori magnetici [30][31].

2.2 Visualization

Un altro aspetto rilevante a proposito di *augmented reality* riguarda il modo in cui vengono mostrate le informazioni del livello aumentato e le tipologie di display sfruttati. È possibile distinguere le seguenti categorie di strumenti di visualizzazione [10]:

- Head-Mounted Display (HMD)

- Hand-held Display
- Ambient Projectors
- Handheld Projectors

Head-Mounted Display La prima categoria di display, tra le più usate nei primi sistemi AR, è quella dei cosiddetti *Head-Mounted display* (HMD): si tratta di dispositivi composti da uno o due schermi combinati con sistemi ottici in grado di fornire un effetto di sovrapposizione corretto dal punto di vista prospettico nonostante la vicinanza agli occhi dell'osservatore. Le due principali tipologie di HMD sono distinte dal metodo di visualizzazione dell'ambiente reale: se il sistema riproduce le immagini ricevute da una videocamera connessa con il display opportunamente elaborate si parla di *video-see-through display* mentre se le immagini digitali vengono sovrainposte usando display semitrasparenti od effetti ottici parliamo di *optical-see-through display*. Questa tecnologia, inizialmente considerata non *consumer-ready* a causa dell'ingombranza e del costo dei dispositivi, è stata recentemente riproposta come piattaforma per applicazioni AR grazie ai progressi visti in materia di hardware dedicato.

Hand-Held Display I display dei dispositivi mobili (*hand-held display*) sono oggi tra i più usati dagli applicativi AR in quanto possiedono ormai capacità computazionali e sensori in grado di garantire una buona esperienza AR. La loro diffusione commerciale fornisce una ampia base di potenziali utenti: i comuni tablet, telefoni cellulari, e smartphone sono infatti considerate ottime piattaforme per semplici applicazioni AR. Gli *hand-held display* rientrano nella categoria *video-see-through* ovvero sfruttano le immagini provenienti dalla videocamera del dispositivo per generare il risultato finale mostrato sullo schermo in tempo reale. Questa categoria di display, in un certo senso, può essere definita attraverso la metafora della *magic lens*: il termine indica generalmente una interfaccia *see-through* che permette all'utente avere una visione diversa della realtà mentre la osserva attraverso la lente. L'idea fu inizialmente concepita per applicazioni desktop e di virtual reality e solo in seguito applicato ad esempi AR, “*In a sense, all Augmented Reality (AR) displays are magic lenses, in that the virtual content can only be viewed through the display.*”[6]. Esistono applicazioni di

augmented reality spesso legate all'intrattenimento ed alla pubblicità che sfruttano come dispositivo di output un normale schermo, portatile o meno, per fornire un'esperienza AR che rispecchia il cosiddetto paradigma del *magic mirror*: un sistema composto da videocamera e display si comporta come uno specchio dove l'utente può vedersi riflesso assieme ad oggetti virtuali. Queste semplici configurazioni non richiedono hardware aggiuntivo, per questo sono attualmente le più usate in applicazioni AR.

Projection-Based AR I sistemi che visualizzano le informazioni del livello AR proiettandole direttamente sulle superfici e sugli oggetti reali si dicono *projection-based*: un tradizionale proiettore, solitamente fisso ed in un ambiente chiuso, permette l'utilizzo del sistema in maniera meno intrusiva rispetto ai display ed inoltre risulta naturalmente predisposto ad applicazioni multi utente. Questi strumenti (*ambient projectors*) tuttavia non garantiscono la portabilità tipica dei dispositivi mobili e necessitano particolari condizioni di luminosità per funzionare in maniera corretta.

Hand-held Projectors L'introduzione di proiettori portatili (*hand-held projectors*) ha portato a nuove sperimentazioni [18] introducendo sistemi in grado di riempire alcune lacune dei classici sistemi portatili, tra cui la limitata risoluzione dell' *hand-held display* ed il ristretto campo visivo di un HMD, mantenendo, entro certi limiti, lo stesso grado di portabilità. Un esempio di questa categoria è SixthSense [12]: il progetto consiste in un proiettore ed una telecamera indossabili in grado di riconoscere gesti grazie a marker colorati sulle dita dell'utente e visualizzare un'interfaccia su qualunque superficie. Un modello di *smartglass* recentemente presentato sfrutta una configurazione simile integrata in un paio di occhiali: il progetto CastAR integra una coppia di microproiettori nella montatura i quali proiettano le immagini su di uno speciale pannello trasportabile. Il pannello è realizzato con materiali riflettenti in grado di visualizzare immagini in alta definizione ed allo stesso tempo, grazie a piccoli circuiti integrati, tracciare eventuali marker sulla propria superficie.

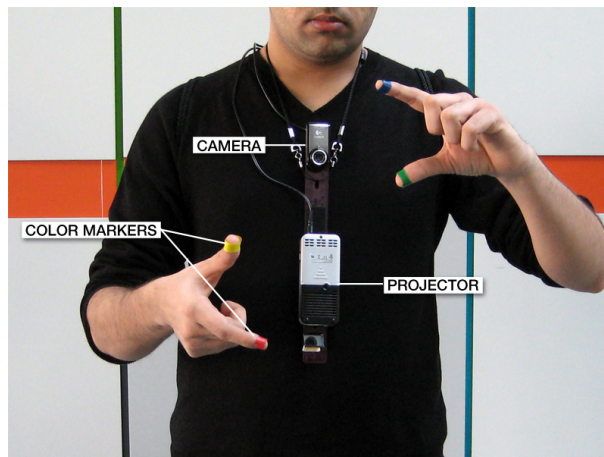


Figura 2.1: Il sistema SixthSense



Figura 2.2: Il prototipo CastAR - TechnicalIllusions

Capitolo 3

Interazione

*AR is slowly moving from being a simple visualization tool to an immersive, interactive MEDIUM.[9]*¹

*The computer interface is, above all, a facilitator of interaction*²

Oltre a sovrapporre informazioni alla percezione del mondo reale, un sistema AR deve fornire una interfaccia di interazione che può essere composta da elementi reali e virtuali. Per sfruttare appieno le potenzialità della tecnologia, questo sistema dovrebbe permettere all'utente di accedere e modificare gli oggetti virtuali allo stesso modo in cui interagisce con quelli reali, ovvero con movimenti tridimensionali con sei gradi di libertà: tuttavia nella maggior parte delle applicazioni mobile le interazioni sono spesso limitate a click e movimenti in due dimensioni sulla superficie del display. Il cosiddetto paradigma di interazione *Windows, Icons, Menus and Pointing* (WIMP) non risulta particolarmente efficace se applicato a sistemi AR. L'interfaccia, sebbene in maniera diversa rispetto ad una comune interfaccia desktop, dovrebbe garantire la selezione e possibilmente la manipolazione diretta degli oggetti virtuali. Questo paradigma di interazione tuttavia, risulta ancora una sfida aperta per la ricerca. In questo capitolo vengono elencate le principali categorie di tecniche di interazione per sistemi basati su *augmented reality*.

¹La realtà aumentata si sta lentamente trasformando da un semplice strumento di visualizzazione ad un medium interattivo ed immersivo

²L'interfaccia di un computer è, prima di tutto, un mezzo per facilitare l'interazione

3.1 Tecniche di interazione e UI

La realtà aumentata fa da ponte tra il mondo reale e quello virtuale rendendo possibile usare oggetti reali, anche di uso comune, come interfaccia di input verso il sistema. Una applicazione AR quindi non isola completamente gli utenti dal mondo reale, ma, al contrario, fornisce un paradigma di interazione più intuitivo e naturale rispetto alla tradizionale coppia mouse-tastiera. Una conseguenza di questi vantaggi è il fatto che l'attenzione dell'utente possa passare senza particolari sforzi da elementi reali a quelli virtuali e viceversa, rendendo sempre più trasparente l'integrazione di quest'ultimi. Fino a qualche anno fa, le applicazioni commerciali basate sulla tecnologia AR si concentravano sull'aspetto di visualizzazione limitando l'interazione con l'utente ad operazioni come la rotazione di modelli tridimensionali per poterli osservare da un diverso punto di vista [10] : l'uso della realtà aumentata viene così ridotto a quello un controller interattivo.

3.1.1 User Interfaces

Tangible UI Una *tangible user interface* (TUI) è una tipologia di interfaccia grafica che permette l'interazione con un sistema software attraverso oggetti fisici: si tratta di una categoria molto ampia in cui rientrano i mouse e gli schermi sensibili al tocco. I *touch-screen* sono oggi le interfacce di input più utilizzate in applicazioni basate sulla realtà aumentata mobile. Fin dai primi anni duemila una configurazione di questo tipo è conosciuta come Tangible Augmented Reality [34] [18]. Oltre alle ormai diffuse interfacce touch screen, un interessante esempio di questo tipo di tecniche è quello presentato da Kato [18]: una persona usa una paletta per selezionare ed organizzare una serie di mobili in una applicazione AR per arredare una stanza, i movimenti dell'oggetto reale vengono mappati sotto forma di comandi usando svariati sensori trasformandolo in un puntatore 3D. Il principale vantaggio di questa applicazione consiste nella possibilità di interagire con il sistema senza particolare esperienza e, nel caso servano istruzioni specifiche, quest'ultimo è in grado di mostrare sfruttando le potenzialità di visualizzazione AR quali movimenti corrispondono ai comandi di input.

Haptic UI Una *haptic AR experience* è basata su di un sistema in grado di comunicare attraverso vibrazioni, variazioni di temperatura o in generale qualunque segnale percepibile dal senso del tatto. Questi feedback vengono proposti all'utente quando viene rilevato un particolare elemento nelle vicinanze: questo elemento può essere visuale, uditivo, basato sulle coordinate del sistema o su una qualsiasi caratteristica rilevante dell'ambiente circostante.

Kinetic UI Grazie all'introduzione di dispositivi in grado di tracciare fedelmente i movimenti dell'utente, è stato possibile introdurre interfacce basate sul riconoscimento dei gesti. Utilizzando sistemi di questo tipo, vengono liberate le mani dell'utente permettendo un'interazione più naturale con gli elementi dell'interfaccia e con gli oggetti virtuali. Recentemente sono stati proposti modelli di *smartglass* e SDK dedicati che incorporano interfacce di questo genere [28] [22] [27].

Tracking dello sguardo (*Gaze Tracking*) Sempre con lo scopo di liberare le mani dell'utente, una tecnica di interazione alternativa è quella di rilevare i movimenti oculari interpretandoli come input al sistema. Vista la ormai prossima commercializzazione di dispositivi indossabili come ad esempio *smartglasse* e simili, questa proposta potrebbe affermarsi come standard di interazione in applicazioni AR *hands free*. Un esempio di interfaccia grafica che sfrutta questa tecnica è mostrata nei *concept design* proposti da APX-Labs [3] per la loro piattaforma di sviluppo applicazioni per *smartglasse*.

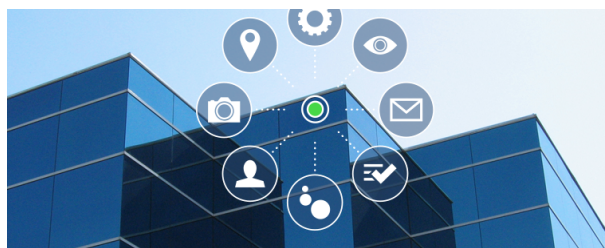


Figura 3.1: Il menu fornito dall'interfaccia di Skylight

Interfaccia Acustica e Riconoscimento vocale L'idea di usare contenuto audio per fornire esperienze AR è già alla base di diverse applicazioni oggi in commercio, inoltre microfoni ed auricolari di alta qualità possono già trovarsi in numerosi dispositivi mobili in quanto facilmente integrabili ed economici. Per questi motivi l'uso di algoritmi di riconoscimento vocale e feedback auditivi risulta essere, sotto alcune condizioni, un'affidabile tecnica di interazione per sistemi AR; tuttavia ambienti con un forte inquinamento acustico possono seriamente impedire un corretto funzionamento di questo tipo di interfacce.

Immissione del testo Il problema di immissione del testo riguarda gran parte delle tecniche di interazione per sistemi AR: se da un lato questo tipo di sistemi offre metodi naturali di selezione, modifica ed accesso al contenuto virtuale, dall'altro non fornisce un efficace sostituto della tastiera per l'immissione del testo. I recenti progressi degli algoritmi di riconoscimento vocale e l'utilizzo di strumenti come penne digitali tuttavia dimostrano l'esistenza di soluzioni utili in un ampio campo di applicazione. Una recente proposta attribuita a Samsung [2] riguarda il possibile uso di una tastiera sovrapposta alla mano dell'utente come metodo di input, un'idea simile è stata proposta da Desale e Ahire per il progetto SixthSense [12]: sfruttando un proiettore portatile è possibile visualizzare una tastiera numerica sulla mano dell'utente.

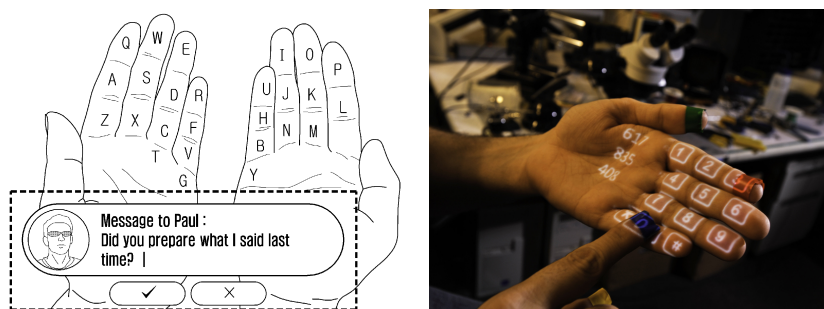


Figura 3.2: A sinistra, lo schema concettuale, a destra la tastiera virtuale del progetto SixthSense

Hybrid UI Viste le potenzialità delle varie tecniche di interazione, è plausibile che applicazioni commerciali AR sfruttino interfacce miste per migliorare l'esperienza e compensare alle lacune dei singoli casi.

3.1.2 Collaborative AR: applicazioni multi utente

Un altro aspetto rilevante è quello della cosiddetta *collaborative AR*: permettere cioè l'interazione tra più utenti sia nello stesso luogo che a distanza. A questo scopo, un sistema AR o MR può essere sfruttato per 'aumentare' uno spazio di lavoro condiviso introducendo fattori che un normale sistema di comunicazione non permette di considerare. Il primo esempio di collaborative AR che sfrutta una tecnologia mobile è AR Tennis [15] in questa applicazione due giocatori osservano attraverso i loro telefoni cellulari una serie di marker per ARToolkit ed il sistema sovrappone il modello virtuale di un campo da tennis. Durante il gioco, gli utenti possono colpire una pallina virtuale muovendo i dispositivi, con conseguente suono e vibrazione. Buona parte delle applicazioni CAR riguardano l'uso di un sistema di videoconferenza AR ampliando le possibilità di comunicazione tra gli utenti, fino ad adesso limitate dalla tecnologia usata. Stafford et al. [32] propongono una nuova metafora di interazione multi-utente, applicata al caso di utenti guidati a distanza dalle indicazioni di un supervisore esperto ('godlike interaction') le quali vengono mostrate sovrapposte alla visuale del mondo fisico: nel caso in questione, un utente all'aperto vede la mano del supervisore che lo guida nel compito da svolgere.

Hybrid Interaction L'esempio descritto sopra presuppone diversi metodi di interazione con il sistema e gli altri utenti: per realizzare una applicazione AR multi utente che garantisca sufficiente flessibilità occorre quindi sviluppare un'infrastruttura in grado di adattarsi a diversi dispositivi di input e soprattutto identificare quali sono le tecniche di interazione ottimali che fanno uso di questi ultimi, in quanto in campo AR non esiste ancora uno standard paragonabile ai classici paradigmi di interazione desktop e mobile.



Figura 3.3: ARTennis



Figura 3.4: Supervisor Collaborative AR

3.2 Scenari Futuri, Blended Reality Space

La tecnologia AR crea opportunità per esplorare nuovi paradigmi di interazione tra il mondo reale e quello virtuale: la classica coppia mouse-tastiera associata alla metafora della scrivania, seppur flessibile come interfaccia input-output bidimensionale, è infatti limitata quando applicata a casi come spazi di lavoro condivisi o ambienti distribuiti [16]. Recentemente, nella letteratura, i concetti di Mixed Reality, TUI e Ubiquitous-Pervasive Computing vengono sempre più spesso affiancati nell'esplorazione di possibili nuove tecniche di interazione uomo-macchina: il mondo fisico viene così 'trasformato' in un'interfaccia di input, rendendo il design delle tecniche di interazione un punto focale nel processo di sviluppo del software. Hoshi et al [16] presentano la nozione di Blended Reality Space come un ambiente MR progettato appositamente per fornire un'esperienza immersiva ed interattiva basata sul rilevamento dei movimenti dell'utente, in grado di permettere l'accesso alle informazioni digitali in maniera trasparente. Nella loro pubblicazione, applicano il concetto appena presentato realizzando un sistema interattivo di comunicazione integrato per la casa e dedicato a fornire un innovativo metodo di accesso alle informazioni digitali a persone anziane (AGNES). Il sistema è composto da una serie di display integrati nell'ambiente, oggetti fisici come dispositivi di input e meccanismi di interazione tra cui il riconoscimento di gesti. Utilizzando un piccolo oggetto circolare realizzato con diversi materiali, gli utenti sono in grado di inviare messaggi e richiedere servizi semplicemente toccandolo o manipolandolo in maniera diversa: gesti come bussare sul legno per chiamare qualcuno fanno parte del modo naturale di comunicare e quindi risultano di facile comprensione anche ad utenti meno esperti. Le informazioni digitali ricevute, messaggi o notifiche, vengono mostrate utilizzando particolari suoni, luci o addirittura correnti d'aria: ad esempio un vento forte che muove le tende della stanza con una luce led rossa può indicare un messaggio urgente.

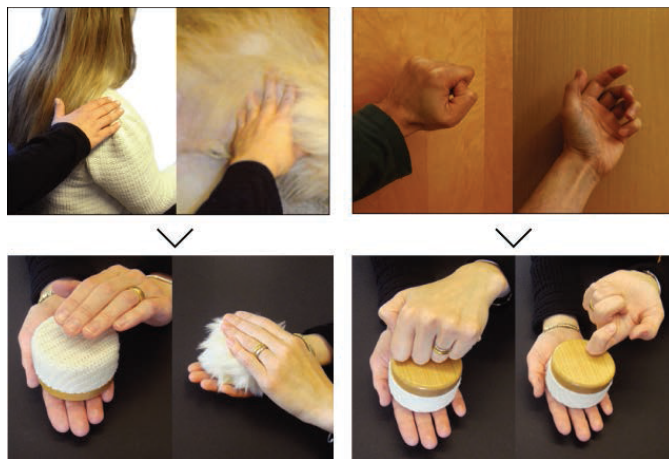


Figura 3.5: Esempi di input al sistema AGNES

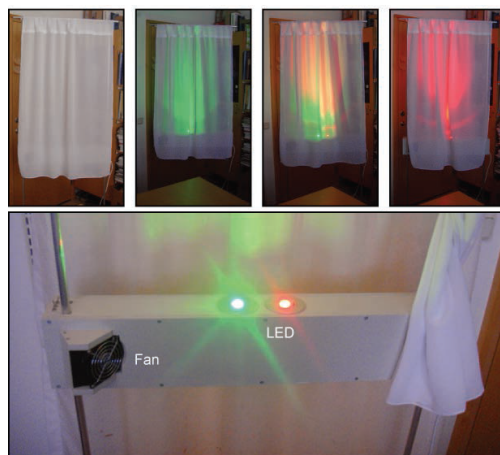


Figura 3.6: Dispositivi di output del sistema

Capitolo 4

Architettura di un'Applicazione AR e Pattern per la Progettazione

In questo capitolo viene presentata una architettura di riferimento generale di un'applicazione AR illustrandone i vari moduli ed alcune proposte di pattern per la progettazione di quest'ultimi.

4.1 Architettura di un'Applicazione AR

Nel 2004 Brügger et al. [7], in seguito allo studio di vari sistemi software di *augmented reality*, mostrano che sebbene risultino apparentemente diversi, essi condividono una struttura architeturale di base: per questo viene proposta una architettura di riferimento basata su un insieme di sottosistemi ognuno riservato a particolari operazioni.

4.1.1 Sottosistemi dell'Architettura

Vengono qui identificati i componenti fondamentali di un'applicazione di *augmented reality*:

Application Subsystem L'*application subsystem* è un elemento astratto che mira ad includere tutta la logica specifica dell'applicazione. Esso può

consistere nell'insieme di codice *application specific* e dati di configurazione e contenuto, è inoltre responsabile dell'inizializzazione del sistema.

Tracking Subsystem Il processo fondamentale di *tracking* deve essere gestito parallelamente alle altre operazioni dell'applicazione per garantire una corretta esperienza in tempo reale. Le varie implementazioni delle tecniche di *tracking* possono sfruttare il cosiddetto *world model* per ottenere informazioni sull'ambiente in cui operano. I risultati di questo processo vengono comunicati alla parte di sistema incaricata della visualizzazione ed al *Context Subsystem*.

User Input Subsystem Questo sottosistema rileva e gestisce tutte le azioni dell'utente. È importante effettuare la distinzione tra questo tipo di input e altri come il movimento ed il cambiamento di posizione: per questo vengono considerate *User Input* tutte le azioni consapevolmente fatte dall'utente per controllare il sistema. Le diverse tecniche di interazione supportate possono essere gestite da un componente apposito: i dati in ingresso dai dispositivi di rilevamento vengono prima processati in ricerca di un *user input* da questo sottosistema e poi smistati agli altri componenti.

User Output Subsystem Questa parte controlla quello che viene mostrato all'utente ed in generale l'output del sistema verso quest'ultimo. Oltre alla gestione di elementi grafici tridimensionali, esso comprende anche notifiche, testo e audio. I componenti che fanno parte di questo sistema sono responsabili di parte del processo di *registration* e dell'effettivo *rendering* degli oggetti del livello aumentato.

Context Subsystem Il *Context subsystem* gestisce diverse tipologie di dati e le rende disponibili agli altri sottosistemi. A seconda dell'applicazione, potrebbe essere necessario gestire informazioni di varia natura ad esempio preferenze dell'utente, dati provenienti dai sensori e dal sistema di *tracking*, informazioni su risorse accessibili al sistema.

World Model Subsystem Le informazioni ricavate dall'ambiente, ad esempio le posizioni degli elementi virtuali relative ad oggetti fisici, vengono memorizzate in un *world model*, una struttura che rappresenta come il

sistema AR percepisca il mondo fisico e le proprietà degli elementi virtuali integrati al suo interno. Questo componente può consistere in un insieme di rappresentazioni digitali di oggetti reali, una collezione di oggetti virtuali e parametri necessari agli algoritmi di *tracking*. Durante l'esecuzione, il sottosistema gestisce inoltre gli accessi al *world model*.

4.2 Pattern per la Progettazione dei Sottosistemi

Brügge et al. [7], inseriscono inoltre nella loro pubblicazione un interessante elenco di possibili approcci all'implementazione delle singole parti dell'architettura: ne viene riportato qui sotto l'elenco con una breve descrizione.

4.2.1 Application

Main Executable. Realizzare l'applicazione usando un linguaggio di alto livello di astrazione per avere un maggiore controllo del flusso di esecuzione, la *augmented reality* è usata solamente come tecnica di visualizzazione. Lo svantaggio di questo pattern è lasciare allo sviluppatore la responsabilità di gestire il continuo processo di *registration* e *rendering* riducendo così la flessibilità e l'estensibilità del sistema.

Scripting. Lo scopo di questo pattern è quello di permettere il rapido sviluppo di nuove applicazioni sfruttando un linguaggio di scripting. I componenti che hanno vincoli di performance (solitamente le parti che devono operare in tempo reale) vanno implementati in un linguaggio tradizionale ed incapsulati in modo da offrire un'interfaccia nel linguaggio di scripting scelto. Questa configurazione permette la rapida prototipazione di applicazioni ma ha diversi limiti nel caso di applicazioni complesse. Un esempio concreto di linguaggio di scripting per la realtà aumentata è AREL (*Augmented Reality Experience Language*) di Metaio basato su JavaScript.

Node in a scene graph. Uno *scene graph* è una struttura dati organizzata sotto forma di un insieme di nodi usata spesso per rappresentare

spazialmente gli elementi di una scena grafica. L'obiettivo di questo pattern è quello di implementare la logica dell'applicazione nel *world model* integrando il codice nei nodi e di fatto in specifiche posizioni reali. Questo approccio offre la possibilità di implementare facilmente applicazioni condivise da più utenti vicini: un'interfaccia 3D può essere condivisa ma visualizzata da ogni utente da un diverso punto di vista.

Parte di un Event Loop. Come visto nella descrizione del processo di esecuzione di una applicazione ARToolkit, è possibile integrare la logica all'interno del *main loop*, lasciando il compito di gestire i processi di *tracking* e *rendering* ad un framework adatto allo scopo.

Servizio Web Questa astrazione gestisce un' esperienza AR come una tipologia di contenuto multimediale. Gran parte delle applicazioni AR diffuse commercialmente sono *content-based* ed un approccio *web-based* si rivela utile alla gestione di grandi quantità di dati multimediali. Sia il *world model* che le esperienze o scene AR possono essere viste come un documento. Il flusso di controllo è gestito da un server attraverso un servizio web accessibile al client parte dell'applicazione: se la risposta alla richieste di quest'ultimo contiene contenuto AR, questo verrà mostrato all'utente. Applicazioni che sfruttano questo pattern devono consentire e mantenere una connessione client server o implementare localmente istanze di entrambi.

Multimedia Flow Description È possibile appoggiarsi a linguaggi di alto livello per descrivere le esperienze AR, in questo modo si semplifica notevolmente la creazione di nuovo contenuto. I linguaggi più usati a questo scopo in SDK commerciali sono xml e HTML5. Il pattern si rivela utile quando è necessario, ad esempio, mostrare una sequenza di scene AR gestite da un componente in grado di interpretare i linguaggi di markup e controllare il flusso della presentazione.

4.2.2 User Input

Come già sottolineato precedentemente, la scelta delle tecniche di interazione con il sistema ed il modo in cui gli input dell'utente vengono interpretati è di notevole importanza se si vuole sfruttare tutte le potenzialità della

tecnologia AR: in questo paragrafo vengono riportati alcuni approcci allo sviluppo dei sottosistemi dedicati a questo aspetto.

Gestione interna all'applicazione. Il metodo più semplice per gestire gli *user input* è quello di includere la logica di interazione all'interno del codice applicativo: questo permette di creare tecniche ed interfacce ad hoc per ogni applicazione riducendo tuttavia l'estensibilità del sistema.

Usare le funzionalità di input di un browser AR. Quando viene utilizzata una applicazione esterna per il *rendering* del contenuto AR (ad esempio un *Browser AR*) è possibile sfruttare gli eventi generati dai click e gesti dell'utente all'interno dell'ambiente di esecuzione di questa come input alla propria applicazione. Questo approccio limita la portabilità del sistema, vista la grande varietà di dispositivi e metodi di input presenti in campo AR.

Reti di Input Device. Questo pattern riguarda l'uso di diversi dispositivi di input e tecniche di interazione: con questa configurazione, occorre introdurre un livello di astrazione che permetta l'interazione con un componente che operi da *controller*. *Unmiddleware* può rivelarsi utile nel caso si debba introdurre nuovi dispositivi dinamicamente.

4.2.3 Tracking

In questa sezione sono concentrati gli approcci allo sviluppo di componenti per la gestione di dati di tracking e non i particolari dispositivi o algoritmi.

Tracking Server. Una possibile soluzione all'elevata richiesta di risorse computazionali degli algoritmi di tracking è spostare il carico verso un server e restituire al client il solo risultato. Questo modello è sfruttabile nel caso si usi un sistema di tracking esterno.

Reti di Tracker. Un elevato numero di sensori può portare a problemi di gestione dei dati: questo pattern propone di incapsulare con interfacce software questi sensori e di permetterne la comunicazione su richiesta agli altri componenti del sistema attraverso un *middleware*. Una volta connessi,

il componente ed il sensore comunicano in maniera trasparente. Come per i dispositivi di input, l'uso di un *middleware* permette l'integrazione dinamica di sensori di vario genere.

Risorse del Sistema Operativo. Nel caso più semplice di un sistema di tracking non fisicamente distribuito, è possibile appoggiarsi ai sensori del dispositivo su cui viene eseguita l'applicazione: nel caso di un moderno *smartphone* videocamera, IMU e GPS sono accessibili attraverso i driver del sistema operativo. L'intero carico computazionale del processo di *tracking* tuttavia ricade sul dispositivo il quale deve essere in grado di gestire allo stesso tempo l'applicazione ed il sistema di tracking.

4.2.4 World Model

Codice OpenGL. Gli elementi virtuali possono essere direttamente descritti nel codice sorgente OpenGL, compilati e caricati durante l'esecuzione da una libreria. I parametri dei singoli modelli vengono modificati direttamente dal main loop ogniqualvolta risulti necessario modificare il punto di vista. Questo approccio si rivela utile in fase di *testing* ma risulta poco efficace in applicazioni più complesse in quanto OpenGL è una libreria di basso livello di astrazione per grafica 3D.

Scene Graph. Questa struttura dati è attualmente il metodo più usato per memorizzare scene grafiche, risulta quindi intuitivo sfruttarne il concetto nelle *AR Experience*.

Object Stream. Nel caso risulti necessario memorizzare oggetti in particolari formati o intere scene su disco, una possibile soluzione è la serializzazione dei dati ed il loro recupero quando necessari: questo introduce però tutti i problemi relativi agli *object stream* come ad esempio la necessaria conoscenza a priori delle diverse classi.

File di Configurazione. Il metodo più semplice per informare i componenti del *tracking subsystem* su quali caratteristiche o marker cercare è sfruttare un file di configurazione che possa essere caricato nella fase di inizializzazione. Nel caso di un elevato numero di elementi rilevabili ed

in generale informazioni più complesse di *marker* tuttavia può risultare problematico aggiornare in tempo reale il file.

Database. Se l'applicazione deve gestire un'elevata quantità di dati, è conveniente che l'elemento astratto *word model* corrisponda ad un database. Il sistema quindi richiederà il contenuto di una scena attraverso una query ottenendo in risposta il contenuto AR. Questo modello si applica specialmente nell'ambito mobile AR permettendo l'accesso dinamico al contenuto di un database online.

4.2.5 Context

Il *Context Subsystem* è una parte astratta del sistema non necessariamente in relazione con la tecnologia AR; tuttavia resta comunque un elemento fondamentale nell'architettura: per questo vengono elencati noti pattern di sviluppo per questo tipo di sottosistema.

Blackboard. Il pattern *blackboard* è applicabile nel caso risulti necessario gestire grandi quantità di dati da raffinare man mano nei vari passi del processo. I componenti produttori di informazioni scrivono sulla *blackboard*, i consumatori leggono, processano i dati e possono a loro volta scrivere informazioni più "raffinate". In sistemi AR complessi questo tipo di configurazione può rivelarsi un collo di bottiglia ma allo stesso tempo permette di disaccoppiare i dati da chi li produce.

Repository. Il *Context Subsystem* si comporta come un deposito di informazioni ben noto agli altri componenti del sistema che vi accedono per leggere o scrivere informazioni. La principale differenza dal pattern precedente lo spazio degli indirizzi usato per gestire gli accessi ai dati.

Publisher/Subscriber. I componenti che forniscono informazioni si connettono come *publisher* ad un sistema di scambio messaggi, mentre i consumatori come *subscriber*. I *publisher* immettono informazioni in un determinato canale il quale si occupa di recapitarle ai *publisher* "iscritti" ad esso. In questa configurazione, risulta facoltativa la memorizzazione delle informazioni, le quali vengono recapitate all'istante ai destinatari.

Ad Hoc Per casi più semplici potrebbe non essere necessario un sistema di scambio dati centralizzato: un elevato numero di connessioni dirette tra i componenti può d'altra parte compromettere la leggibilità del codice e l'estensibilità del sistema.

4.2.6 Output e Rendering

Siccome l'*User Output subsystem* si occupa principalmente del rendering di contenuto 3D, i seguenti pattern sono principalmente legati a questo aspetto.

AR Browser. Un AR Browser fa riferimento ad una classe di applicazioni che forniscono contenuto AR a dispositivi in grado di visualizzarlo. Solitamente gli AR browser offrono più funzionalità e forme di realtà aumentata di una applicazione *single-purpose*. Il pattern fornito sfrutta questo tipo di applicazioni come sistemi di rendering. I principali vantaggi di questo approccio sono la riusabilità del componente ed il formato standard per descrivere le esperienze AR.

OpenGL. Molte applicazioni commerciali e browser AR proprietari usano OpenGL come motore di rendering 3D in quanto risulta uno strumento di sviluppo particolarmente flessibile. Modificando le proprietà degli oggetti 3D in base alle informazioni ricevute dal *tracking subsystem*, la scena può essere visualizzata in maniera corretta.

Scene Graph. Al posto di usare una libreria come OpenGL, per scenari più complessi occorre affidarsi a motori di rendering che forniscono funzionalità con un livello di astrazione maggiore e maggiore libertà di configurazione rispetto ad un AR Browser.

Trasferimento Video. L'applicazione recupera il flusso video e le informazioni per il rendering e li trasferisce ad un server nell'ambiente di esecuzione con il compito di sovrapporre e codificare le immagini prima di restituirle al client. Particolarmente adatto nel caso di dispositivi *video-see-through*, questo pattern soffre delle classiche problematiche relative allo scambio di dati e latenze.

Classi Viewer Multiple. Per poter organizzare diverse tipologie di informazioni (3D, 2D, audio, testo ecc.) e gestire dispositivi di output di vario genere, conviene sfruttare diversi formati per i dati dedicati ad ognuno di essi dotandoli di una interfaccia astratta per facilitarne la gestione.

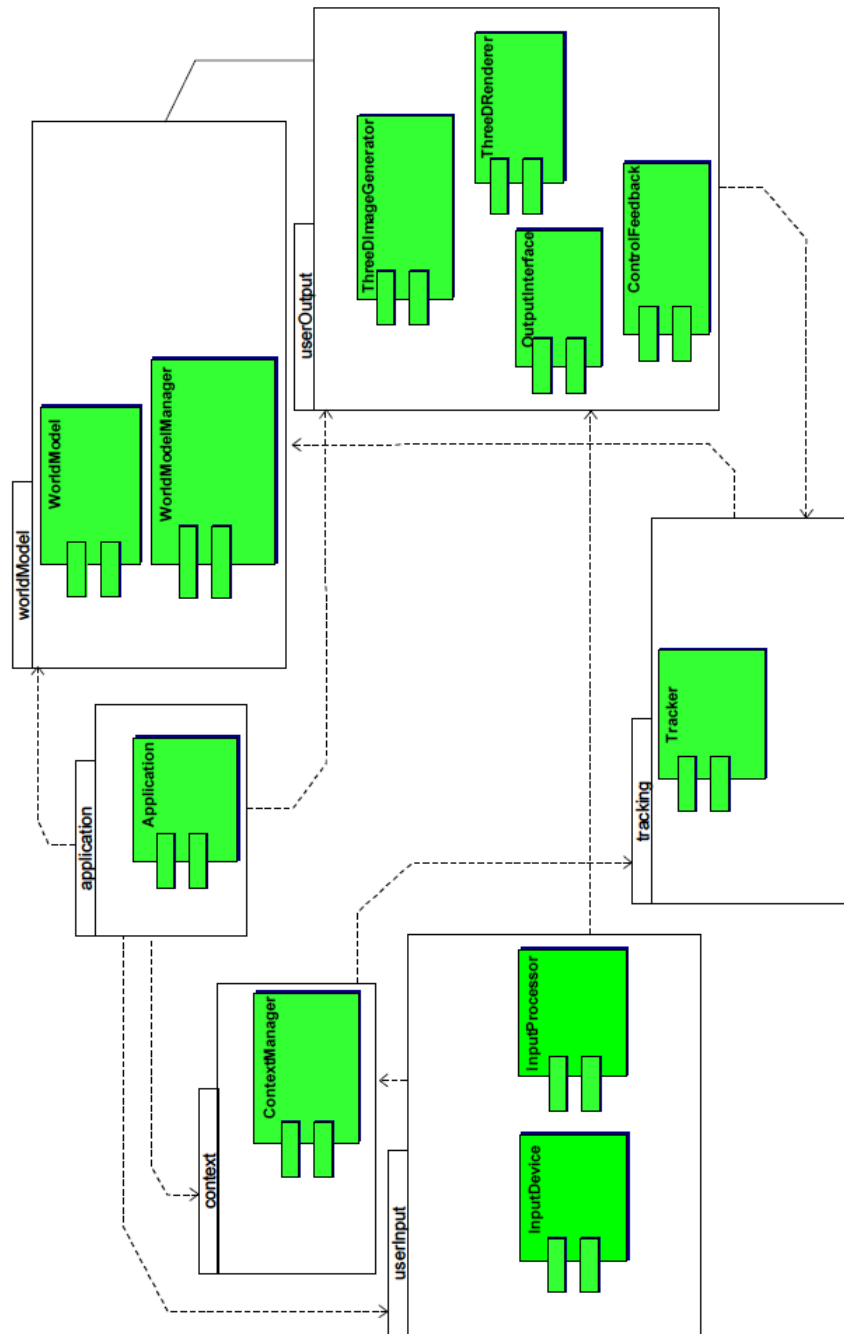


Figura 4.1: Architettura di riferimento per un'applicazione AR.

Capitolo 5

Strumenti Per Lo Sviluppo e per la Pubblicazione di Applicazioni AR

Nella prima sezione di questo capitolo viene fornita una breve descrizione di ARToolkit, una libreria *open source* usata per diverse applicazioni AR nel corso degli anni evidenziando la struttura del *framework* ed il processo di esecuzione di una semplice applicazione. Successivamente, vengono presentate le due offerte alternative per lo sviluppo di software e contenuto AR proposte oggi dalle più importanti aziende del settore: la prima si concentra sulla produzione e pubblicazione di esperienze AR sotto forma di collezione di elementi virtuali da aggiungere a determinate posizioni reali, il tutto sfruttando un sistema *web-based* descritto in maniera più approfondita nella seconda sezione; a questa proposta viene spesso affiancato un framework per lo sviluppo di applicazioni AR con diverse possibili funzionalità: più avanti in questo capitolo verranno brevemente confrontati gli SDK proposti da Layar e Metaio, Qualcomm (Vuforia) e Wikitude.

5.1 ARToolkit

Sviluppata nel 1999 da Kato [17], ARToolkit fu la prima libreria software realizzata appositamente per facilitare la creazione di applicazioni AR basate su *marker-tracking*: tutt'oggi è mantenuta come progetto open-source alla base di diverse soluzioni commerciali. Il tool multiplatforma consiste

in un set di librerie C/C++ le quali definiscono le principali funzioni necessarie al tracking ed alla integrazione con le librerie grafiche (GLUT) ed il motore di rendering OpenGL.

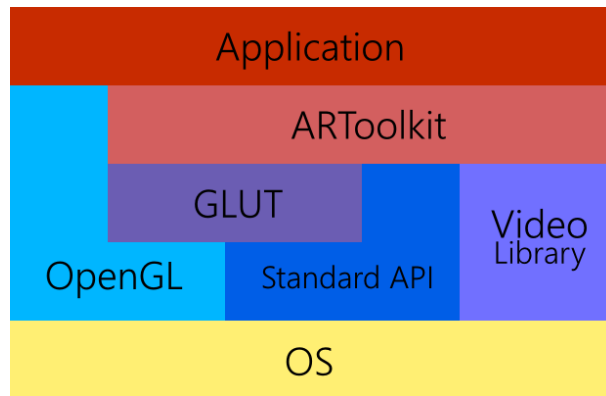


Figura 5.1: Struttura del framework ARToolkit

La libreria consiste in tre moduli principali: il primo (AR) fornisce tutte le funzioni necessarie al processo di tracking e di calibrazione della videocamera; seguono una collezione di routine standard per gestire i fotogrammi in input e la parte per la visualizzazione in finestra del risultato finale. Il processo di una semplice applicazione basata su ARToolkit viene descritto in Figura 5.2. Per ottenere un'interpretazione corretta dell'immagine occorre inoltre calibrare la videocamera attraverso l'analisi di particolari pattern stampati dotati di caratteristiche distinguibili note alla funzione di calibrazione. Il processo mostrato in figura riprende i passaggi descritti da un punto di vista più teorico precedentemente in questa tesi: durante la fase di inizializzazione (funzione `init`) vengono caricati in memoria i pattern da riconoscere ed i parametri ottenuti dal processo di calibrazione; nella funzione, viene infine inizializzato il flusso video in ingresso. I punti da 1 a 4 del *main loop* vengono ripetuti fino alla chiusura dell'applicazione: ottenuto un fotogramma dal dispositivo di input (`arVideoGetImage`), il programma deve rilevare la presenza di marker (`arDetectMarker`). Una volta analizzato il fotogramma è possibile identificare tutti i marker rilevati attraverso il parametro `marker_info`, inizializzato dalla funzione precedente. Nella fase di *registration*, vengono calcolate e memorizzate in una matrice di trasformazione le coordinate e l'orientamento della videocamera rispetto all'oggetto



Figura 5.2: Processo di esecuzione di una applicazione basata su ARToolkit

(`arGetTransMat`); segue la visualizzazione dell'oggetto grazie a OpenGL. L'ultima fase del processo si occupa di liberare gli spazi in memoria e di terminare l'acquisizione dati dal flusso video.

5.2 Tipica struttura di un Framework AR

Le applicazioni di *augmented reality* devono affrontare tipiche problematiche a partire dal processo di tracking, registration e visualization fino alla scelta dell'interfaccia grafica e delle tecniche di interazione: queste funzionalità possono essere implementate indipendentemente dalle varie esperienze ed applicazioni AR attraverso lo sviluppo di un *framework* apposito. L'uso di questi strumenti permette una veloce integrazione di dispositivi hardware, la realizzazione di interfacce grafiche ottimizzate alle diverse modalità di visualizzazione ed in generale l'implementazione di funzionalità AR all'interno di applicazioni. A differenza del sistema di creazione e gestione del contenuto descritto nella sezione successiva, simile in tutte le proposte analizzate, gli SDK e le API proposti offrono diversi gradi di libertà nell'uso delle funzionalità dei framework proprietari: possiamo tuttavia descriverne i principali scopi e le caratteristiche comuni.

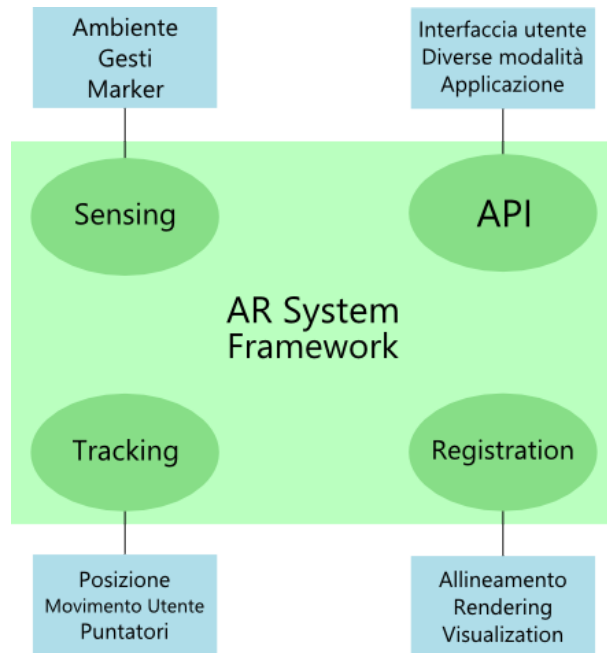


Figura 5.3: Rappresentazione schematica delle funzionalità di un framework AR

Tracking e Sensing Come prevedibile, la maggior parte dei framework analizzati supporta più modalità di tracking come tecniche *sensor-based*, *marker-based* e *NFT*: è possibile quindi inserire elementi virtuali in specifiche coordinate GPS o in posizioni relative a immagini note all'applicazione. Inoltre i sensori e le videocamere di nuova generazione permettono rilevazioni di alta qualità anche in caso di ambienti non noti a priori: nascono quindi nuove tecniche di *instant-tracking* come SLAM 3D di Metaio e Extended Tracking di Vuforia, entrambe descritte nella prossima sezione.

Registration e Visualization Il calcolo della posizione dell'utente e delle relative coordinate degli oggetti virtuali è una funzione fondamentale per qualunque libreria software dedicata alla *augmented reality*: nel caso di *location-based registration* e quindi di tecniche di tracking basate sulla posizione GPS del dispositivo, la posizione degli elementi virtuali è corretta in tempo reale grazie a dati provenienti da sensori inerziali (IMU, *Inertial*

Movement Unit) e magnetici. Gli SDK commerciali permettono di affidarsi a motori di rendering esterni come OpenGL per la visualizzazione degli elementi sovrapposti: in alcuni casi (Metaio e Wikitude) ne possiedono uno integrato per garantire maggiore supporto e compatibilità.

API Le API dedicate agli sviluppatori permettono di includere il browser AR direttamente all'interno della propria applicazione (ad eccezione di Vuforia che non offre un browser *stand-alone*). Solitamente viene effettuata la distinzione tra realtà aumentata basata su computer vision (*Visual-based Augmented Reality*) e l'inserimento di elementi in specifiche coordinate GPS (*GeoSpatial-based Augmented Reality*): nel caso di quest'ultima, l'utente seleziona la tipologia di contenuto desiderato, il sistema restituisce quindi i *Points of Interest* (POI) vicini alla sua posizione e l'elemento multimediale legato ad essi. Una funzionalità offerta dagli SDK di Layar e Metaio è la cosiddetta Visual Search: questo processo consiste nell'ottenere informazioni relative ad un oggetto reale attraverso l'invio di immagini o informazioni ad un *visual search engine*. Questa operazione non rientra nell'ambito della realtà aumentata ma la possibilità di riconoscere elementi del mondo reale può permettere l'identificazione di *trigger* per esperienze AR o addirittura di sfruttare gli oggetti identificati come marker. I sistemi operativi target sono solitamente iOS ed Android, essendo questi attualmente i più diffusi in campo mobile, più avanti in questo capitolo verranno mostrate le piattaforme supportate da ciascuno degli SDK.

5.3 Architettura del sistema offerto dai principali provider

L'esperienza AR viene solitamente gestita come contenuto multimediale visualizzabile dal browser proprietario, rispecchiando il pattern descritto nel capitolo precedente (*content-based*, Servizio Web). La figura 5.4 illustra i componenti dell'architettura definita in questa sezione. I singoli elementi verranno analizzati in seguito alla figura. Un *AR browser* o un'applicazione di terze parti richiede il contenuto attraverso l'*AR client proxy* fornito dal provider, le richieste vengono processate da un servizio proprietario (*AR Publishing Environment (PE)*) e reindirizzate verso il provider del contenu-

to: l'esperienza AR restituita viene validata ed inviata al client incaricato di visualizzarla in maniera corretta.

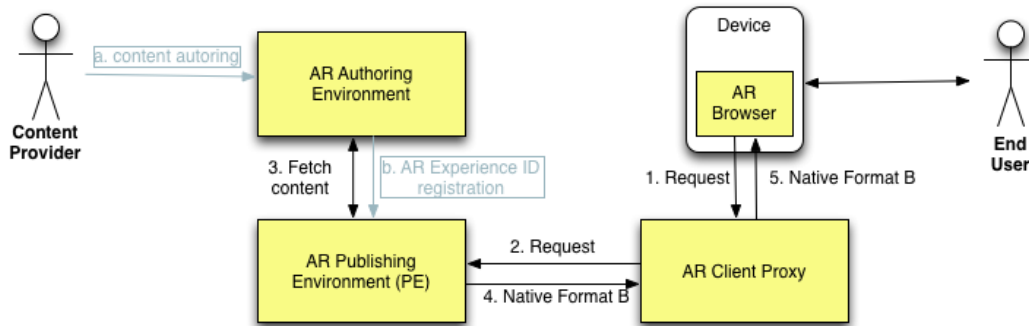


Figura 5.4: Rappresentazione schematica del sistema di pubblicazione ed accesso al contenuto

AR Browser Gli AR Browser sono applicazioni sviluppate e mantenute dal provider del servizio, in grado di visualizzare un'esperienza AR pubblicata attraverso i loro sistemi. L'applicazione è spesso distribuita gratuitamente attraverso i diversi *app store* come parte di una piattaforma *closed service*. Il compito di un AR browser è quello di interpretare gli input dell'utente e quelli provenienti dal mondo reale (in generale i dati rilevati dai sensori di vario tipo) controllando la ricezione e la presentazione dell'esperienza.

Content Provider Un'entità (persona o azienda) che crea il contenuto e lo rende disponibile sotto forma di esperienza AR.

AR Publishing Environment Un servizio dove tutte i contenuti AR compatibili con il browser vengono identificati e registrati.

AR authoring Environment Un termine ùttosto generale per descrivere un qualunque server/servizio/piattaforma dove i provider possono creare e/o salvare il contenuto AR. Nel contesto di questa sezione, esso è un endpoint dal quale il PE ottiene il contenuto.

AR Client Proxy Il client richiede il contenuto sfruttando questo componente il quale restituisce l'esperienza in un formato nativo per il client.

5.4 Layar

Ideata nell'estate del 2009, Layar guadagna rapidamente fama internazionale come uno dei primi browser per realtà aumentata disponibili sul mercato. L'applicazione per iOS ed Android permette di accedere a contenuto digitale visualizzato tramite AR semplicemente inquadrando con la fotocamera del dispositivo un'immagine registrata nel database dell'azienda.

Terminologia Le *AR Experience* Layar sono chiamate *layer*, si tratta di collezioni di oggetti e informazioni digitali pubblicate da un *service provider* e gestite dai server Layar. I *layer* sono distinguibili in due categorie: *vision layer* e *GEO layer* corrispondono rispettivamente ad esperienze *visual-based* e *geospatial-based*.

5.4.1 SDK API

Lo scopo dell'SDK fornito da Layar è quello di integrare il browser proprietario tramite un set di semplici metodi: nel caso di Android, la classe principale è `LayarSDK` la quale, dopo l'inizializzazione, permette di lanciare una *view* con l'interfaccia tipica dell'applicazione Layar in modalità ricerca se non viene specificato alcun layer. La classe `LayarSDKFragment` permette invece di integrare la vista in un'interfaccia personalizzata selezionando la modalità di visualizzazione tra AR e MAP corrispondenti alla visuale di realtà aumentata ed una dall'alto utilizzabile nel caso di un Layer GeoSpatial AR. La classe `LayarSDKClient` opera da interfaccia per la notifica e la gestione di eventi tra cui gli input dell'utente sugli *augment* ovvero sugli elementi digitali e sui menu. I metodi corrispondenti agli eventi possono essere sovrascritti estendendo la classe.

5.4.2 Creator, Content-based API e AR Browser

Il software per la creazione di contenuto è Layar Creator, questo programma permette di aggiungere elementi digitali a particolari immagini marker

nel mondo reale tramite la pubblicazione di un layer: l'esempio più usato è quello di una campagna pubblicitaria su una rivista che, se guardata "attraverso" uno smartphone con il browser AR mostra un video o un modello 3D relativo alla pubblicità. Le offerte proposte da Layaar, SDK compreso, si concentrano su questo aspetto limitando l'accesso alle funzionalità di base della piattaforma. Il sistema di pubblicazione e gestione di layer ha una struttura simile a quella del caso generale: è rilevante notare il fatto che la piattaforma Layaar comprendente i servizi di pubblicazione e validazione dei layer è un elemento necessario al funzionamento di una qualunque applicazione, per adesso infatti non è possibile creare esperienze AR funzionanti offline. Le richieste del browser o della applicazione sviluppata attraverso l'AR Client Proxy fornito dall'azienda avvengono sotto forma di *request* HTTP (`getPOIs`). I parametri obbligatori della richiesta `getPOIs` sono i seguenti:

- `userID`: l'identificatore dell'utente dell'applicazione generato all'installazione sul dispositivo;
- `layerName`: il nome del layer selezionato dall'utente o dall'applicazione;
- `version`: versione dell'API utilizzata;
- `lat,lon,countryCode,lang`: coordinate GPS del dispositivo e informazioni generali sulla localizzazione.
- `action`: specifica se la richiesta riguarda un intero layer od un aggiornamento.

La *response* del provider, dopo essere stata validata dalla piattaforma Layaar (*ARPublishingEnvironment*), consiste in una struttura javascript contenente le informazioni relative agli oggetti da aggiungere alla view sul dispositivo.

5.5 Metaio

Fondata in Germania nel 2003, la società fu originata da una collaborazione con Volkswagen e successivamente finanziata dallo Stato. Nel 2005, Metaio fu tra le prime aziende del settore a rilasciare una applicazione commerciale

basata sulla tecnologia AR chiamata KPS Click and Design con la quale era possibile posizionare pezzi di arredamento in un'immagine. Oggi il loro prodotto junaio è tra i più celebri esempi di browser AR per smartphone.

Terminologia Il browser junaio permette la visualizzazione di esperienze AR con il nome di *AR channel*: come vedremo, a differenza di Layaar, non è necessario appoggiarsi a questa rappresentazione delle scene per implementare una applicazione AR.

Metaio propone a fianco del sistema di creazione e pubblicazione di contenuti multimediali AR un framework *all-purpose* che permette la realizzazione di applicazioni di *augmented reality* indipendenti dal browser proprietario junaio.

5.5.1 SDK API

Come per Layaar, l'accesso alle funzionalità dell'SDK avviene attraverso l'interazione con una classe principale che implementa l'interfaccia `IMetaioSDK`: i metodi pubblici permettono di inizializzare e configurare i diversi componenti del sistema, di creare e gestire gli oggetti virtuali della *AR experience*, i sensori, più videocamere ed in generale il flusso dell'applicazione.

Sistema di tracking Tramite l'interfaccia principale è possibile selezionare una tra le tecniche di tracking supportate dal framework:

- *ID marker*: classico marker 2D, quadrato, in bianco e nero;
- *Picture*: marker 2D realizzato con una qualunque immagine;
- *Image tracking*: markerless feature tracking per immagini 2D;
- *3D Map*: mappa di punti tridimensionale costruita durante il processo di calibrazione (o con il tool apposito fornito da Metaio)
- *Instant 2D/3D tracking*: permette la generazione in tempo reale di feature 2D o mappe di punti 3D (SLAM) per una tecnica di tracking markerless.
- *LLA marker* (*Latitude, Longitude, Altitude*): simili agli ID marker ma con informazioni aggiuntive sulle coordinate globali del marker.

- **QR:** È disponibile una modalità di lettura di codici QR.
- **3D model:** tecnica di tracking *model-based* che sfrutta modelli CAD precaricati nel sistema.
- **Non optical tracking:** usa una combinazione di GPS, giroscopio, bussola e accelerometro per tracciare la posizione e i movimenti dell'utente.
- **Dummy:** tecnica usata in fase di test, restituisce una posizione fissa.

Gestione dei sensori e videocamera L'interfaccia `IMetaioSDK` e le sue derivate dedicate ai diversi sistemi operativi, permettono di controllare direttamente il set di sensori e la videocamera del dispositivo, inviando comandi e richiedendo dati (screenshot, fotogramma, stato dei sensori ecc.). Il framework supporta inoltre più videocamere e l'aggiunta dinamica di sensori.

IGeometry Gli elementi virtuali implementano l'interfaccia `IGeometry`: una `Geometry` può essere creata a partire da immagini 2D, testo, modelli 3D o video; questa classe contiene tutte le informazioni riguardo a posizione, scala e rotazione dell'oggetto.

Rendering L'SDK di Metaio permette di usare un motore di rendering integrato o, alternativamente, uno esterno: l'interfaccia della classe principale fornisce metodi per gestire la visualizzazione degli elementi, effetti grafici, luci, *shader* e *reflection map*.

Controllo flusso dell'applicazione È possibile fermare temporaneamente il processo di tracking o l'intero main loop dell'applicazione tramite appositi metodi.

registerCallback Nel caso risulti necessario gestire eventi relativi al flusso dell'applicazione, il metodo `registerCallback` consente di registrare *handler* che implementano l'interfaccia `IMetaioSDKCallback`.

5.5.2 Creator, Content-based API e AR Browser

Metaio offre un tool per la creazione di *channel* chiamato Metaio Creator, il software si occupa della conversione e dell'integrazione di contenuti multimediali nel formato standard visualizzabile dall'*AR Browser* junaio. I canali sono descritti in un linguaggio di scripting appositamente creato chiamato AREL (*Augmented Reality Experience Language*, basato sulle tecnologie web HTML5, XML e JavaScript. La GUI specifica per ogni canale è implementata come un qualunque sito web *mobile* con sfondo trasparente, gli elementi della scena virtuale sono descritti in formato AREL XML mentre la logica dell'applicazione e di interazione con i singoli oggetti è implementata in JavaScript grazie ad una libreria ad hoc (AREL.js).

5.5.3 CloudPlugin e AREL Interpreter

È possibile integrare il supporto di visualizzazione di *AR channel* grazie al plugin a pagamento *Metaio CloudPlugin*. L'SDK è dotato inoltre di un interprete AREL che permette lo sviluppo di applicazioni direttamente in questo linguaggio di scripting. Uno dei vantaggi di questa tecnica di sviluppo è la possibilità di effettuare facilmente il porting da un sistema operativo all'altro, essendo il linguaggio indipendente dalla piattaforma.

5.6 Vuforia e Wikitude

In questa sezione presentiamo brevemente Qualcomm Vuforia e Wikitude ARchitect, mostrandone brevemente le caratteristiche non presenti negli SDK analizzati precedentemente.

5.6.1 Vuforia





L'SDK fornito da Qualcomm è concentrato su esperienze *video-see-through marker-based*. Un elemento notevole è il supporto nativo al riconoscimento testuale come marker. Il motore di tracking sfrutta come default la cosiddetta tecnica di *extended tracking*, ovvero una combinazione di un marker e SLAM attorno ad esso per garantire un allineamento consistente anche nel caso il marker esca dal campo visivo della telecamera.

5.6.2 Wikitude

Wikitude offre un content creator, un SDK dedicato a device specifici (EPSON moverio, Google Glass, VuZIX) ed un browser mobile: le esperienze AR vengono chiamate ARchitect World e sono codificate in pagine HTML le quali sfruttano le API ARchitect per generare gli oggetti da visualizzare. Un elemento virtuale (ARObject), è direttamente associabile ad una posizione fisica ed attivabile automaticamente quando l'utente entra in una determinata area geografica. Il principale punto di forza dell'SDK di Wikitude è l'essere interamente basato su tecnologie e linguaggi web e, per questo, facilmente integrabile in nuove piattaforme.

5.7 Confronto

Seguono due tabelle che confrontano caratteristiche e sistemi operativi supportati dai quattro SDK analizzati: Metaio risulta attualmente l'offerta più completa, permettendo di sfruttare il sistema *content-based* per la rapida produzione di contenuto AR, la rappresentazione *web-based* delle esperienze ed allo stesso tempo la possibilità di creare sistemi più complessi.

	iOS	Android	Blackberry	Web
ARToolkit				
				
				
Qualcomm 				
 wikitude				





	GPS	IMU	Marker	NFT	Visual Search	Face Tracking	Content API	Framework
ARToolkit								
								
								
Qualcomm 								
 wikitude								

Figura 5.5: Confronto delle piattaforme supportate e delle funzionalità degli SDK analizzati.

Conclusioni

Le *AR experience* nelle applicazioni oggi diffuse sono considerate contenuti multimediali con una semplice logica di interazione basata su linguaggi di scripting. Con l'avvento dei nuovi dispositivi specializzati nel campo della *augmented reality*, si aprono nuove possibilità per lo sviluppo di applicazioni dedicate e per l'evoluzione del concetto di *software* stesso: l'idea di sviluppare un sistema che presenti componenti e logica integrati con oggetti reali o associati ad una posizione fisica è strettamente correlata al paradigma di *ubiquitous computing*. La metafora della scrivania o *desktop* fu proposta per aiutare gli utenti a comprendere il modo in cui operava un computer [16]: con l'introduzione di un livello virtuale interattivo integrato in maniera trasparente nell'ambiente reale non sarà più necessario un processo di apprendimento né un particolare sforzo per accedere ad informazioni digitali. Sebbene rimanendo in un'ottica *content-based*, gli SDK analizzati implementano già il concetto: gli elementi visivi virtuali infatti (GeoLocated Object) sono legati ad una specifica posizione reale ed integrano una logica di interazione. Un primo passo verso il superamento delle metafore di implementate dagli attuali sistemi software è passaggio concettuale da un GeoLocated Object o equivalenti ad un vero e proprio componente software situato dotato di uno stato e di un comportamento che va al di là della semplice reazione a input dell'utente.

Ringraziamenti

Desidero ringraziare tutti coloro che mi hanno aiutato nella stesura della mia Tesi con suggerimenti, critiche ed osservazioni, anche se è mia responsabilità l'eventuale presenza di errori o imprecisioni in questo documento.

Prima di tutto ringrazio il mio Relatore, Alessandro Ricci, sempre disponibile ad ascoltare ed interpretare le mie esigenze, facilitando la mia ricerca del materiale e fornendo utili consigli per la realizzazione della Tesi.

Un ringraziamento particolare a Federica, alla quale questa Tesi è dedicata, ai miei familiari ed a tutte le persone che mi hanno incoraggiato e hanno speso parte del proprio tempo per discutere con me le bozze del lavoro.

Bibliografia

- [1] T. Ahonen. Tedxtalks, 2012. <http://tedxtalks.ted.com/video/TEDxMongKok-Tomi-Ahonen-Augment>.
- [2] APX-Labs. Ar keyboard concept, 2014. <http://www.apx-labs.com/products/skylight/>.
- [3] APX-Labs. Skylight, 2014. <http://www.apx-labs.com/products/skylight/>.
- [4] R. Azuma. A survey of augmented reality. In *Presence: Teleoperators and Virtual Environments*, 1997.
- [5] R. Azuma, Y. Baillet, R. Behringer, S. Feiner, S. Julier, and B. McIntyre. Recent advances in augmented reality. In *IEEE Computer Graphics and Applications*, 2001.
- [6] D. Baricevic, C. Lee, M. Turk, T. Holler, and D. Bowman. A hand-held ar magic lens with user-perspective rendering.
- [7] B. Brügge. Study on software architectures for augmented reality systems. 2002.
- [8] T. Caudell and D. Mizell. Augmented reality: An application of heads-up display technology to manual manufacturing processes. In *Proceedings of 1992 IEEE Hawaii International Conference on Systems Sciences*, 1992.
- [9] S. Corbett-Davies, R. Green, and A. Clark. Physically interactive tabletop augmented reality using the kinect. 2012.
- [10] E. Costanza, A. Kunz, and M. Fjeld. Mixed reality: A survey. 2009.

-
- [11] A. Damala and et al. Adaptive augmented reality for cultural heritage: Artsense project. 2012.
- [12] R. D. Desale and V. S. Ahire. A study on wearable gestural interface. a sixthsense technology. In *IOSR Journal of Computer Engineering (IOSR JCE)*, 2013.
- [13] A. Dünser, R. Grasset, and H. Farrant. Towards immersive and adaptive augmented reality exposure treatment. 2011.
- [14] I. Getting. Perspective/navigation- the global positioning system. In *Spectrum*, vol 30, 1993.
- [15] A. Henrysson, M. Billinghamurst, and M. Ollila. Artennis. 2006.
- [16] K. Hoshi, F. Öhberg, and A. Nyberg. Designing blended reality space: Conceptual foundations and applications. 2011.
- [17] H. Kato and M. Billinghamurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. 1999.
- [18] H. Kato, M. Billinghamurst, I. Poupyrev, K. Imamoto, and K. Tachibana. Virtual object manipulation on a table-top ar environment. In *ISAR 2000*, 2000.
- [19] KIA. Kia ces 2014, 2014. <http://kia-buzz.com/ces-2014-wrap-up/>.
- [20] C. Lai and C. Wang. Mobile edutainment with interactive augmented reality using adaptive marker tracking. 2012.
- [21] M. Livingston and et al. Military applications of augmented reality. 2011.
- [22] META. Spaceglasses, 2014. <http://kia-buzz.com/ces-2014-wrap-up/>.
- [23] Metaio. Custom enterprise solutions, 2014. <http://www.metaio.com/services/custom-solutions/>.
- [24] P. Milgram and F. Kishino. Taxonomy of mixed reality visual displays. In *IEICE Transactions on Information and Systems*, 1994.

-
- [25] Mitsubishi. Mitsubishi presents meview ar, 2014. <http://www.youtube.com/watch?v=EJGscLyptP0>.
- [26] J. Newmann, D. Ingram, and A. Hopper. Augmented reality in a wide area sentient environment. In *ISMAR 2004*, 2004.
- [27] OnTheGo. Onthego gesture recognition sdk for smartglasses, 2014. <http://www.otgplatforms.com/developers/>.
- [28] L. Optical. Lumus smartglasses, 2014. <http://http://www.lumus-optical.com/>.
- [29] J. Reikimoto. Augmented reality using the 2d matrix code. In *In Proceedings of the Workshop on Interactive Systems and Software (WISS'96)*, 1996.
- [30] K. Satoh, S. Uchiyama, and H. Yamamoto. A head tracking method using bird's eye view camera and gyroscope. In *ISMAR 2004*, 2004.
- [31] K. Satoh, S. Uchiyama, H. Yamamoto, and H. Tamura. Robust vision-based registration utilizing bird's eye view with user's view. In *ISMAR 2003*, 2003.
- [32] A. Stafford and W. Piekarski. User evaluation of god-like interaction techniques. 2008.
- [33] I. Sutherland. A head-mounted three dimensional display. In *Proceedings of Fall Joint Computer Conference*, 1968.
- [34] F. Zhou, H. Duh, and M. Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. 2008.