ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

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

# FRAMEWORK E LIBRERIE PER LO SVILUPPO DI APPLICAZIONI DI REALTÀ AUMENTATA: METAIO SDK COME CASO DI STUDIO

Elaborata nel corso di Sistemi Operativi LA

Tesi di Laurea di:
ALBERTO SPERANDIO

Relatore:
Prof. ALESSANDRO RICCI

ANNO ACCADEMICO 2013–2014
SESSIONE I

# PAROLE CHIAVE

**Augmented Reality**

**SDK**

**Metaio**

A Longo, Fio, Costa, Mazzo, Stefe e Cangio

# Indice

# Introduzione

I moderni sistemi computazionali hanno reso applicazioni e dispositivi sempre più complessi e versatili, integrando in essi un numero crescente di funzioni. Da qui si avverte la necessità di un design d'interfaccia utente efficace e pratico che renda il rapporto uomo/macchina semplice ed intuitivo. Negli ultimi anni questo proposito è stato accolto da sviluppatori e progettisti che si sono affacciati nel mondo della "Realtà Aumentata", una nuova visione d'insieme nel rapporto tra mondo reale e virtuale. Augmented Reality (AR), propone infatti di sviluppare nuove interfacce uomo-computer, che invece di mostrare le informazioni digitali su display isolati, immergano i dati stessi nell'ambiente concreto. Sfuma così una distinzione marcata tra il reale e il virtuale, ma anzi si cerca di combinare in modo naturale la coesistenza di quest'ultimi, permettendo la creazione di interfacce utente semplici e intuitive anche per applicazioni complesse. Il proposito che la tesi vuole andare ad affrontare è proprio quello di indagare lo sviluppo di nuove applicazioni basate su questa tecnologia. Nel primo capitolo verrà analizzatala storia, i campi di applicazione, i device più importanti sui quali è implementata e le varie tecniche di Tracciamento. Nella seconda parte della Tesi andremo a interessarci del sistema vero e proprio sul quale regge questa tecnologia. Quindi nel successivo capitolo vedremo esempi di architetture e di piattaforme che offrono questa realtà di sviluppo, soffermandoci su un particolare caso di studio: Metaio; di cui nel terzo e ultimo capitolo indagheremo framework, SDK e API messe a disposizione.

# Capitolo 1

# Panoramica sulla Realtà Aumentata

## 1.1 Cenni storici

L'inizio della ricerca sulla AR può essere collocato nel 1957, anno in cui Morton Heilig, fotografo, brevetta un simulatore chiamato *Sensorama*. Si tratta del primo esempio di esperienza multi-sensoriale conosciuto. Il prototipo era sostanzialmente costituito da un dispositivo meccanico nel quale lo spettatore poteva vedere, ascoltare e nello stesso tempo perfino odorare cinque cortometraggi attraverso immagini, suoni, vibrazioni, e odori.

Sarà poi Ivan Sutherland nel 1966, professore ad Harvard, a concepire il primo vero sistema di realtà aumentata: l'*Head Mounted Display* (HMD). Un dispositivo di visualizzazione da indossare sul capo o come parte di un casco, con un piccolo display ottico di fronte a uno o a entrambi gli occhi . Il sistema era piuttosto rudimentale sia in termini di interfaccia che di realismo, e l'ambiente creato era molto schematico e semplice. La prospettiva che il software mostrava però non era assoluta e dipendeva dalla posizione dell'utente, qui nasceva uno dei primi problemi che accompagnerà tutta l'evoluzione della AR: il *Registration problem*; l'oggetto fisico e quello virtuale devono essere correttamente allineati tra loro.

Dalla fine degli anni 60 a oggi è stato un susseguirsi di innovazioni e release di prototipi, su questa nuova branca delle scienze *ICT* che ancora però non era stata ben definita, divenendo settore di ricerca solo agli inizi degli anni 1990. Anno in cui Tom Caudell, introduce per la prima volta il

termine Realtà Aumentata, per descrivere un display digitale utilizzato per la manutenzione elettronica degli aerei, che miscela grafici virtuali con una realtà fisica. Anche se oggi la definizione per il mondo informatico di realtà aumentata è più dettagliata, essa rimane essenzialmente la stessa: la realtà aumentata è l'interazione di elementi grafici, audio e altri miglioramenti sovrapposti ad un ambiente reale che vengono visualizzati in tempo reale.

Basatesi su questa definizione ne seguiranno ulteriori due principali che supereranno e miglioreranno la tesi proposta da Caudell, entrambi accettate dal mondo scientifico. La più accreditata è stata formulata da Ronald Azuma nel 1997 con la pubblicazione del suo elaborato "A survey of augmented realit", nella quale il ricercatore americano propone una definizione formale di realtà aumentata caratterizzata da tre requisiti essenziali:

- Unire il reale e virtuale
- Registrazione nel mondo reale in 3D
- Interazione in tempo reale

Al di là del significato che il primo punto può evocare, evidente e immediato, una notevole importanza ricoprono i successivi due requisiti. La definizione di Azuma difatti, si preoccupa di escludere tutte quelle tecnologie del tempo che non aggiornavano il contenuto virtuale in 3D e in tempo reale, distinguendo così l' AR da tutte le altre "*augmentations*" off-line, come la computer grafica 2D/3D.

Pochi anni prima però Paul Milgram e Fumio Kishino, nel 1994, avevano introdotto un nuovo concetto dello spazio-tempo, concependo il mondo reale e virtuale come i due estremi di una estensione continua (*continuum reality*) in cui al suo interno è posta una area di confine chiamata *mixed reality*: una nuova visione della realtà intesa come coesistenza di reale e virtuale in un unico display. Nella loro pubblicazione, Milgrame e Kishino non agiscono con l'intento di sostituire il mondo reale con un artificio umano, ma piuttosto quello di aumentare la vista dell'utente del mondo reale con informazioni aggiuntive.

Da qui nasce la netta distinzione che i due ricercatori si apprestano a rimarcare tra *augmented* e *virtual reality*: per realtà virtuale si intende la completa immersione in un ambiente artificiale da parte dell'utente, nel quale possono esserci o meno caratteristiche comuni con uno specifico mondo

reale; nel secondo caso, con il termine realtà aumentata, gli elementi che appartengono al livello virtuale sono in stretta correlazione con una proprietà o oggetto fisico del mondo reale, arrivando ad essere del tutto vincolati. Dalla seconda metà del 2000 la ricerca sul AR è migliorata ulteriormente portando innovazioni e sviluppo nel settore HW e SW su device che utilizzano questa tecnologia, ampliando così la vasta gamma di campi di applicazione di essa.

## 1.2 Devices

Se prima abbiamo parlato di quanto un ostacolo allo sviluppo AR sia rappresentato da problemi puramente coordinativi tra reale e virtuale, riscontrabili in un contesto software, non sono da meno le difficoltà di portabilità che provengono dai device e componenti che sfruttano queste tecnologie. La realtà aumentata è un complemento naturale alla ricerca del *mobile computing* , dal momento che un sistema AR mobile può aiutare direttamente l' utente sul posto di lavoro invece di richiedere l'uso di *workstation* fisse. Si ritiene pertanto ovvio che uno dei principali prossimi passi nello sviluppo AR *mobile* sarà un passaggio a dispositivi ergonomici sempre più piccoli e intuitivi. L'integrazione di strumenti come videocamere 3D, microproiettori, e sensori (posizione, profondità..) ha portato allo sviluppo di vari dispositivi con un elevato potenziale di Augmented Reality. Tali dispositivi che prendono il nome di *Wearable Device* sono classificabili nella categoria dei dispositivi elettronici integrati in vestiti o degli accessori indossabili.

### 1.2.1 Wearable Devices

#### Display Head-Mounted

La categoria dei Display Head-Mounted (HMD) è stata ampiamente utilizzata e analizzata nei sistemi di realtà virtuale partendo da quello che può essere catalogato come il primo prototipo mai rilasciato (1966). I ricercatori di AR hanno lavorato su due tipologia di HMD: *video see-through* e *optical see-through*. I primi nascono dal bisogno dell'utente di avere una visione immediata del mondo reale, grazie alle riprese delle videocamere montate sul dispositivo. L'HMD standard utilizzato in applicativi di realtà virtuale fornisce all'utente il completo isolamento visivo dall'ambiente circostante, di solito impiegati in contesti di controllo remoto di qualche dispositivo (e.g.

Robot). Dato che il display è isolato visivamente il sistema deve utilizzare videocamere allineate con il display stesso per avere la visione del mondo reale.

Un HMD ottico invece elimina il canale video che sta guardando la scena reale. La fusione del mondo reale e virtuale avviene otticamente di fronte all'utente, attraverso solitamente un vetro semi-trasparente inclinato (*smartglasses*). Questa tecnologia è simile agli *heads-up display* ( HUD ) che comunemente appaiono nelle cabine di pilotaggio di aerei militari e di recente in alcune automobili sperimentali. In questo caso però, la fusione ottica delle due immagini viene fatto sul display montato sulla testa o su particolari lenti poste singolarmente davanti a ogni occhio, piuttosto che sul vetro della cabina stessa o sul parabrezza. Soprattutto per una miglior qualità dell'immagine e della visione finale, i device ottici stanno conquistando mercato e notorietà, spingendo la ricerca scientifica a ipotizzare futuri prototipi come *Contact Lenses* [Fig 1.1] bioniche indossate direttamente sull'occhio dell'utente. D'altra parte però le tecnologie ottiche mancano di una immediata sincronizzazione tra mondo reale e virtuale; sincronizzazione che invece è più efficiente in un dispositivo video-see-through, in quanto il *matching* non viene elaborato in tempo reale ma prima che l'utente visualizzi l'effettiva ripresa.

### Dispositivi Handheld

Separatamente alla categoria Display device, abbiamo tutti quei dispositivi che sfruttano la tipologia video-see-through attraverso un piccolo display montato su un device portatile. I dispositivi cosiddetti *handheld* sfruttano la loro estrema praticità per una miglior portabilità dell'esperienza AR. Questa categoria di visualizzazione è la più pratica e la più diffusa, grazie sostanzialmente alla completezza di questi sistemi (non necessitano di hardware aggiuntivi) e alla diffusione commerciale che smartphone/glass/watch e tablet stanno avendo negli ultimi anni, considerati ottime piattaforme per queste semplici applicazioni.

### Sixthsense Program

Solo recentemente è stato messo appunto un nuova tipologia di device AR, che in controtendenza con quello che era lo standard fino a d'ora, non cerca

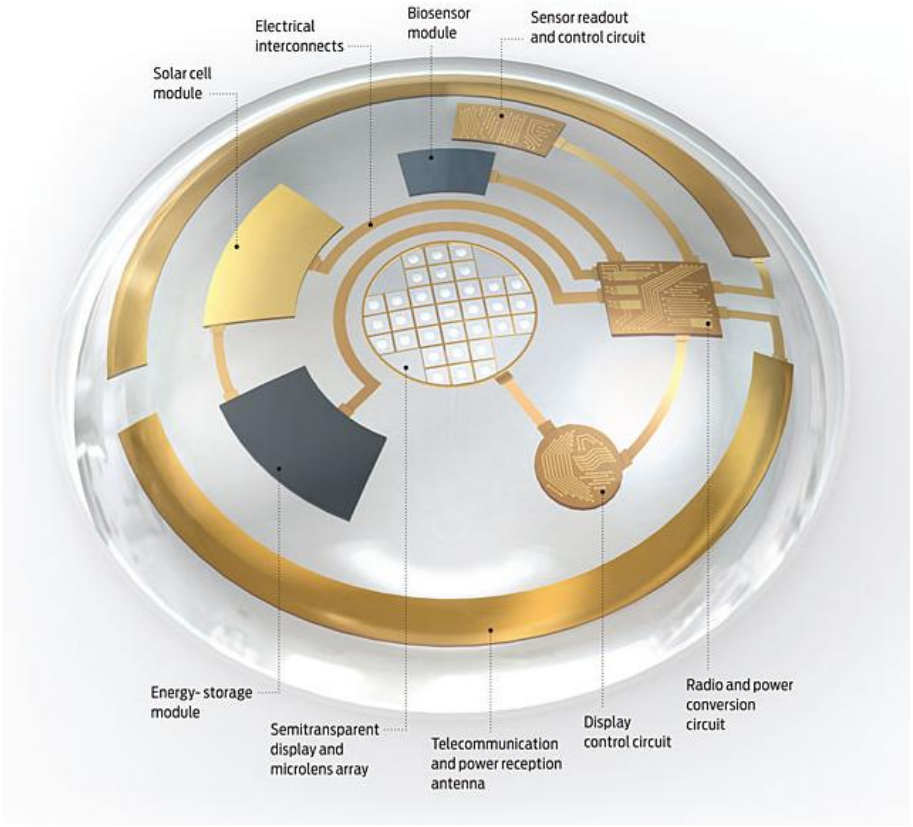


Figura 1.1: AR Lens

d'immergere l'utente in una esperienza virtuale attraverso la vista (relativamente), ma contrariamente grazie all'esperienza tattile. L'idea è quella di trasformare qualsiasi superficie in un display touch-screen.

*SixthSense* è uno strumento indossabile , un'interfaccia gestuale che aumenta il nostro mondo fisico con informazioni digitali, e ci permette di utilizzare i gesti naturali della mano per interagire con informazioni artificiali. Quello che fino a d'ora è un prototipo, comprende un proiettore (fissato al petto), uno specchio e una fotocamera uniti in un dispositivo di mobile computing legato al collo. Il sistema proietta le informazioni sulle superfici e sugli oggetti fisici che ci circondano, rendendo qualsiasi superficie una

interfaccia digitale; la fotocamera infatti riconosce qualsiasi movimento delle mani dell'utente grazie alla presenza di marker colorati sulle punte delle dita dell'utente (che vengono utilizzati per il monitoraggio visivo).

## 1.3 Ambiti Applicativi

La maggior parte delle applicazioni reali incentrate su AR sono create per dimostrazioni o scopi di ricerca. Per un tipo di tecnologia che è ancora in fase di sviluppo, quale essa è, qualsiasi tipo di implementazioni pratiche sono di importanza cruciale. Nel campo della realtà aumentata, questo lavoro di ricerca inizialmente è cominciata *indoor*, dove l'hardware solitamente è di grosse dimensioni e consuma una notevole quantità di energia senza imporre troppe restrizioni al suo utilizzo. Una volta che l'hardware è diventato di dimensioni più piccole e più potente, i ricercatori hanno sviluppato sistemi più complessi e hanno iniziato a muoversi *outdoor*. Questa sezione descrive varie applicazioni che sono state sviluppate sia per ambienti interni ed esterni, sollevando un' interesse generale per la tecnologia e rendendo concreto il potenziale commerciale dietro di essa.

### 1.3.1 Indoor AR

Per realtà aumentata *indoor*, si intende quelle serie di applicazioni che sono state sviluppate per fornire agli utenti un ulteriore consapevolezza situazionale sui loro compiti. Progettando i dati sulla visione effettiva che ha l'utente, le informazioni sono visualizzate direttamente nell'ambiente e l'utente può meglio comprendere il rapporto che i dati virtuali hanno con il mondo fisico. La prima dimostrazione di AR fu fornita da Sutherland, e sull'onda di questa nuova scoperta scientifica ben presto l'interesse militare ha preso il sopravvento. L'uso di HMD dagli anni 70 infatti è stato impiegato per migliorare i già esistenti Head up Display (HUD) installati su *aircraft* militari (progetto *Super Cockpit*). Una tecnologia simile viene utilizzata per implementare display adibiti a mansioni di manutenzione 3D, un esempio e primo grande banco di prova è stato il progetto KARMA sviluppato dal ricercatore universitario Steven Feiner che invece di visualizzare direttamente grafica 3D da un database, utilizza la generazione automatica basata sulla conoscenza della produzione, a seconda di una serie di regole e vincoli definiti per ogni attività o caso di studio. Dal momento che l'uscita non

viene generato in anticipo, il sistema può personalizzare l'output a seconda di condizioni e di esigenze degli utenti. Dalla Columbia University, è stato sviluppato il progetto ARMAR ("Realtà Aumentata per la manutenzione e riparazione"). Il sistema attraverso un visore presenta informazioni visive all'utente, riguardanti i vari step necessari per il mantenimento o la riparazione di macchine industriali e non.

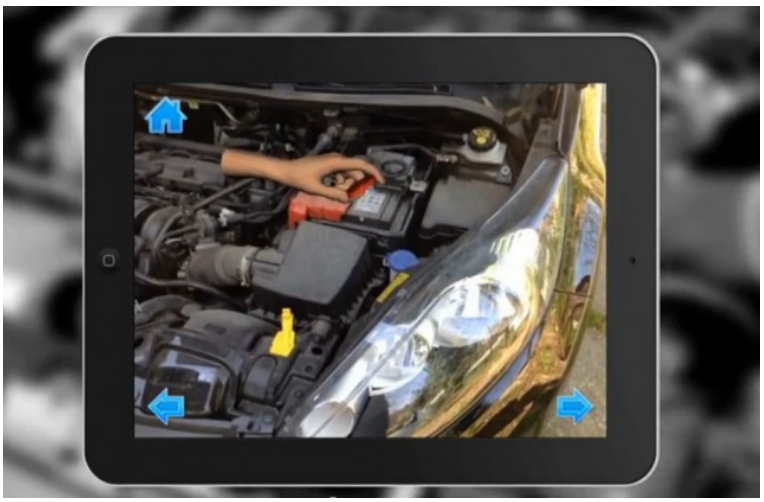

Figura 1.2: AR Maintenance Support

Questa ricerca è una buona dimostrazione di come la tecnologia AR può essere utilizzata per assistere i lavoratori con mansioni complesse del mondo reale.[Fig 1.2] Parallelamente le stesse nozioni di virtualizzazione sono state applicate nel campo medico, come formazione di operatori sanitari. Il sistema *CAE ProMIS* sviluppato nel 2012 per la formazione medica, permette di simulare situazione chirurgiche realistiche. Il sistema inoltre ha moduli diversi, che possono essere utilizzati per modificare l'ambiente di lavoro. Questo permette una formazione più dinamica e diminuisce i costi dell'avere un simulatore specializzato per ogni situazione. Ultimamente molte grandi aziende hanno iniziato ad utilizzare la realtà aumentata come strumento pubblicitario, solitamente utilizzando il dispositivo mobile del cliente. Sfruttando la fotocamera dei device messi in commercio si possono aggiungere informazioni 3D, video e altri contenuti multimediali interattivi a semplici riviste,prodotti,immagini o altri contenuti fisici.

### 1.3.2 Outdoor AR

L'obiettivo finale della ricerca AR è comunque quello di produrre sistemi che possono essere utilizzati in qualsiasi ambiente, senza alcun genere di restrizioni sull'utente. La prima dimostrazione di AR operante in un ambiente esterno è la cosiddetta *Macchina di Turing*, sviluppata sempre alla Columbia University. Questo sistema rudimentale si basa sull'utilizzo di un calcolatore *backpack* con tutte le attrezzature necessarie per il supporto AR (solitamente un visore HMD). La macchina di Turing fornisce agli utenti la posizione dei vari edifici e la specifica funzionalità di essi presso il campus della Columbia. L'interazione con il sistema è realizzata grazie all'uso di una bussola GPS e dell'HDM stesso per gestire la visione dell'esterno; guardando inoltre gli oggetti di interesse più a lungo di un tempo prestabilito, il sistema presenta ulteriori informazioni interattive. Le funzionalità della Macchina Turing sono state poi ulteriormente estese grazie ad un sistema di modellazione 3D che si sovrapponeva al mondo fisico, dando agli utenti la possibilità di vedere edifici che non esistevano più (o che non esistevano ancora). Non solo, a questo è stata aggiunta la possibilità di visualizzare filmati, rappresentazioni e altri contenuti multimediali riferiti ad ogni particolare posizione del campus. Possiamo quindi immaginare gli attuali prototipi e sistemi di visualizzazione outdoor, come delle moderne macchine di Turing, ai quali vengono migliorate le funzionalità (e.g. *Booking*, *Mapping*) e la portabilità (e.g dispositivi *handheld*).

## 1.4 Tracking e Registration

Le applicazioni di realtà aumentata richiedono sistemi di localizzazione di alta qualità in termini di precisione, *jitter* e *lag*. Il cosiddetto "*Tracking*", chiamato anche "*problem of real-time pose determination*", si occupa del recupero della posizione e dell'orientamento di un oggetto virtuale da visualizzare (o del display-utente), in modo da registrarlo rispetto al mondo reale. Affiancato ad esso (e da qui in poi sottinteso con esso) abbiamo la "*Registration*", ovvero l'allineamento del oggetto virtuale con l'utente reale, con un particolare attenzione alla resa prospettica della scena AR. Il monitoraggio deve essere eseguito in *real-time*, in genere richiedendo soluzioni che stimano poco meno di 50 millisecondi, e sotto qualsiasi condizione che potrebbero

indebolire il servizio. Nel caso in cui questo venisse perduto, il sistema deve essere in grado di recuperarlo in fretta.

Esistono varie tecnologie di tracking (acustico,magnetico,inerziale, vision-based) ad altrettanti strumenti che affiancano questa tecnologia (GPS, giroscopi,bussola elettronica..) ma i rigidi requisiti che dovrebbero avere le applicazioni di realtà aumentata sono ancora difficili da raggiungere :

- Elevata precisione spaziale ( 6 gradi di libertà - 6 DOF [1]) in posizione e orientamento.
- Assenza di jitter, cioè disturbo in uscita del sistema di tracking.
- Alta velocità di aggiornamento.
- Assenza di lag, ovvero il ritardo che intercorre dallo strumento di misurazione all'output del sistema.
- Completa mobilità degli utenti ( nessuna restrizione nel movimento e nella portabilità).

Qui di seguito la classificazione in base alle diverse tipologie e ai diversi hardware che accompagnano queste tecnologie:

### 1.4.1 Sensor Based Pose Determination

#### Sensori ad ultrasuoni

Il tracciamento ad ultrasuoni misura il tempo di volo (*time of flight*, TOF) di segnali *chirp* dalle fonti di ultrasuoni ai microfoni per il rilevamento, producendo direttamente i valori di posizione e orientamento assoluti.

Il rilevamento ultrasonico è piuttosto limitato dalle caratteristiche degli impulsi inviati e dalle interferenze ambientali che essi possono avvertire. Poichè gli impulsi viaggiano alla velocità del suono inoltre, possono aumentare ritardi di tracking con la lontananza del sensore dal trasmettitore. Non solo, effetti ambientali come temperatura, umidità e correnti d'aria possono influire sulla precisione delle misurazione .

[1] Si 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).

### Sensori magnetici

Il monitoraggio magnetico si basa sulla misurazione delle forze generate dal campo magnetico terrestre (*magnetismo passivo*) o in alternativa quando questo è troppo debole, da uno prodotto artificialmente (*magnetismo attivo*).

I sensori magnetici però non sono però largamente utilizzati in quanto sono in grado di rilevare unicamente l'orientamento dell'oggetto più che la posizione (in quanto richiederebbe l'utilizzo di altri strumenti); il rilevamento dei dati degli stessi *magnetometri* inoltre, potrebbe venir disturbato anche dalla presenza di altri dispositivi elettronici o metallici nelle vicinanze.

### GPS

Il Tracking GPS misura il tempo di volo di segnali inviati da satelliti posizionati nello spazio , producendo direttamente valori assoluti di posizione. Il Global Positioning System ( GPS ), inizialmente sviluppato unicamente con fini militari, è basato su una costellazione di 24 satelliti in orbita attorno alla Terra, ognuno dei quali trasmetta onde radio appositamente codificati che contengono informazioni di posizione estremamente precise. Un ricevitore ( con conoscenza della reale posizione di almeno 4 satelliti GPS ) può calcolare la propria posizione misurando il tempo di volo (*time of arrival - ToA*) di questi segnali dallo spazio. La qualità e l'affidabilità di questo tracciamento vengono perà fortemente messe in discussione dalla vicinanza di edifici o luoghi chiusi che potrebbero fare da schermo alle onde radio provenienti dai satelliti.

### Sensori inerziali

Rispetto al monitoraggio sensor based introdotto in precedenza, l'inertial tracking offre delle funzionalità complementari interessanti. Innanzitutto è completamente autonomo: non necessita di acquisire informazioni da fonti esterne; i sensori posizionati sull'oggetto misurano informazioni relative al movimento. I sensori inerziali forniscono inoltre un buon rapporto segnale-disturbo, soprattutto in caso di rapido cambio di direzione (accelerazione / decelerazione ) e per un alta velocità di rotazione dell'oggetto da tracciare. Tra questi sensori, i più comuni sono i *giroscopi* e gli *accelerometri*:

- **Accelerometri** Gli accelerometri sono piccoli e semplici dispositivi che misurano le forze di accelerazione applicate ad un oggetto lungo

un unico asse. Portano notevoli vantaggi tra i quali la non necessaria presenza di alcuna fonte di dati, l'economicità, il basso uso di potenza, e la semplicità con la quale vengono aggiunti a dispositivi *wearable*. Il principale svantaggio di questa tecnologia è che il processo di integrazione delle misurazioni soffre di accumulo di errori, portando così un alto fattore di imprecisione nella misurazione dei dati.

- **Giroscopi** Molto simili ai precedenti accelerometri in termine di prestazioni e affidabilitá, i *giroscopi* sono degli strumenti in grado di rilevare le forze rotazionali applicate al sensore, senza far riferimento a una fonte dati.

### 1.4.2 Vision Based Pose Determination

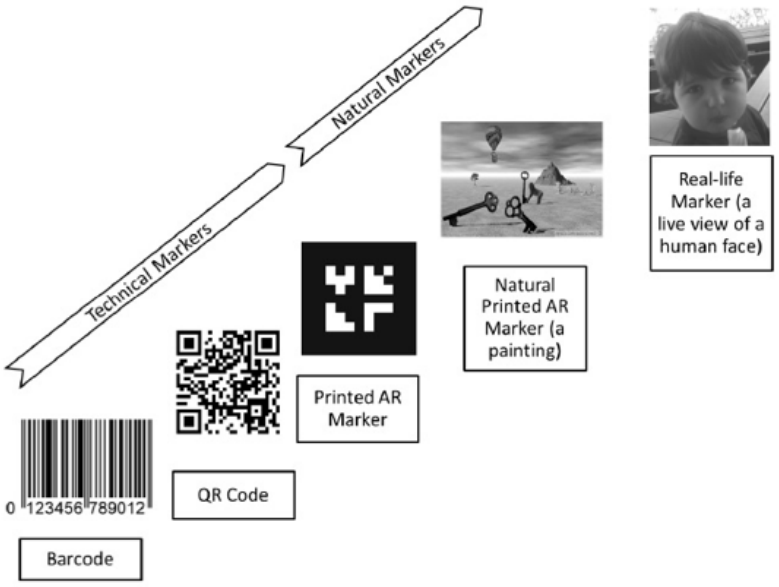


Figura 1.3: Marker Evolution

A differenza di un tracciamento sensoriale, in cui ci si basa prevalentemente sulla affidabilità di sensori autonomi, le tecniche sotto riportate

sfruttano un approccio puramente visivo per il completamento del tracking (*ottico*). Algoritmi di elaborazione di immagini e *computer graphic* analizzano un segnale video in ingresso per ricavare più informazione possibili sulla posizione e l'orientamento nello spazio dell'oggetto. Possiamo distinguere fondamentalmente due approcci radicalmente diversi per un tracciamento *vision-based*:

- *Outside-in tracking*: Le telecamere sono montate in posizione fissa e osservano la scena. C'è un rapporto fisso ( traslazione T + rotazione R ) tra la scena stessa e il sistema di telecamere coordinate: gli oggetti in movimento vengono osservati, e le loro traiettorie sono monitorate.

- *Inside-out tracking*: La telecamera è l'oggetto in sè, allegato a sua volta all'oggetto in movimento da tracciare. In questo caso sono le coordinate della telecamera a dover essere recuperate e monitorate attraverso una relazione variabile con la scena stessa (T(t), R(t)).

In entrambi i casi per'ò la relazione spaziale è stabilita dalla identificazione di caratteristiche comuni (punti, linee, macchie) e la conoscenza di queste caratteristiche (aspetto, dimensioni, posizione nello spazio 3D) è obbligatoria per essere in grado di misurare quantitativamente i 6DoF di libertà dell'oggetto.

Sostanzialmente però possiamo distinguere tre grandi gruppi d'interesse per una corretta determinazione vision-based:

### Marker Based Vision

Questo tecnica di tracciamento sfrutta il riconoscimento di un elemento ripreso dalla realtà e presente nel database delle applicazioni, trasformandolo in un evento digitale con cui l'utente interagisce. Questi simboli, perlopiù immagini in 2D, vengono definiti *marker*, e possono venire rilevati da una qualsiasi fotocamera con un soddisfacente livello di risoluzione. Punto focale di questa tecnologia, è l'intensità luminosa sfruttata dagli apparecchi di tracciamento per calcolare la posizione degli oggetti nello spazio reale. Integranti all'uso della telecamera vengono impiegati anche diversi ricettori, come i sensori CCD (*Charged Coupled Device*), utilizzati per il calcolo della posizione di elementi *passivi* o *fotodiodi* laterali, al fine di elaborare elementi *attivi*. Esempi tipici di dispositivi passivi sono appunto i marker,

elementi attivi invece sono le rappresentazioni tridimensionali che verranno sovrapposte successivamente al marker.

#### Natural Feature Tracking

I precedenti metodi di monitoraggio richiedono però che le immagini contengano caratteristiche fisiche intenzionalmente posizionate sulla scena. Nella categoria dei sistemi di puntamento ottici rientrano quindi anche quei sistemi di riconoscimento dell'immagine che utilizzano algoritmi di computer grafica per calcolare la posizione di oggetti reali, prevalentemente in una scena *outdoor*, non predicibili a priori. Questa categoria di riconoscimento (Natural feature tracking, NFT), senza quindi l'utilizzo di marker, è ben più sensibile a margine di errore. I dispositivi infatti una volta visualizzata la scena, devono riconoscere da diverse angolazioni elementi tipici ricorrenti negli oggetti interessati (linee, punti, angoli). Da questa necessità la NFT è spesso affiancata da altre tecnologie, quali complessi algoritmi di *computer vision* e di *computer graphic*.

#### Cad Based

Un nuovo modo di concepire il monitoraggio visivo da parte dell'utente è quello che viene definito negli ultimi anni *cad-based vision* o in modo più appropriato *model-based vision*: la realtà aumentata applicata all'ambito industriale. Il sistema si basa su tecnologie di visione per il riconoscimento oggetti, di sintesi e riconoscimento vocale, interfacciate con progetti CAD e computer graphic per la virtualizzazione di oggetti. Questa tecnologia si avvicina ancora una volta ad un tracciamento marker based, in quanto sfrutta la conoscenza a priori della forma degli oggetti così da poter virtualizzare al meglio il conseguente movimento dell'operatore. A differenza delle prime due tecnologie il cad based è infatti finalizzato puramente all'ambito industriale e al supporto di mansioni lavorative.

### 1.4.3 Hybrid Tracking

Come abbiamo potuto osservare dalle sezioni precedenti, le tecniche recenti di vision-based e sensor-based non possono provvedere autonomamente a un corretto tracciamento. Le nuove direzioni che stanno prendendo le applicazioni AR sono sempre più indirizzate verso una tecnologia ibrida,

ovvero l'Hybrid Tracking. La fusione infatti di sensori inerziali con l'utilizzo di un canale video che può essere rappresentato da un tracciamento ottico, è risultato soddisfacente in termini di robustezza , velocità di tracciamento e precisione , e per riduzione di jitter e di lag. Un esempio di questa nuova tecnica può essere rappresentato da un sistema vision-based assistito da sensori inerziali quali giroscopi e accelerometri, e da sensori di posizione quali il GPS. Questo offre al sistema completa mobilità e autonomia sul piano spaziale, nel quale la visione attraverso un canale video offre grande affidabilità, integrata con la precisione o la correzione delle misurazioni da parte dei sensori statici.

# Capitolo 2

# Sviluppo di Applicazioni di Realtà Aumentata

In questo capitolo verrà fornita una descrizione sullo sviluppo di augmented reality app, partendo da quelli che sono gli aspetti generali relativi alla progettazione e allo sviluppo di applicazioni AR (architettura, framework). Verranno comparati i 5 SDK maggiormente diffusi a livello commerciale, effettuando un confronto tra un sistema X e un sistema Y senza alcun intento di scegliere un framework migliore di un altro, in quanto ogni architettura software sulla quale si basano, è sviluppata per risolvere un particolare problema applicativo; anzi con il materiale messo a disposizione verrà argomentato ancora una volta una soluzione generale a quello che può significare lo sviluppo di applicazioni AR, prendendo in considerazione nel prossimo capitolo il caso di un framework specifico, *Metaio*. Fin dal momento della massima distribuzione di applicazioni e browser di AR sul mercato mondiale, verso la fine del 2008, si è sentito il bisogno di definire uno standard riguardante la Realtà Aumentata, per consentire una distribuzione di contenuti su più ecosistemi AR, piattaforme e applicazioni. Un numero sempre crescente di collaborazioni sono stati istituite per analizzare l'ambiente AR (e.g. W3C AR Comunità Group) e sono stati inoltre proposti dei primi formati standard (e.g. progetto ARML 1.0, proposto da Wikitude, o il KARML, proposto da Georgia Tech) per dare una prima panoramica a cosa un AR standard dovrebbe assomigliare. Molto recentemente, l'OGC (Open Geospatial Consortium) ha annunciato la formazione di un vero e proprio team, lo Standards Working Group ARML, con l'obbiettivo di definire un AR

Standard ufficialmente accettato all'interno del OGC stesso. Nelle sezioni successive, Martin Lechner ricercatore presso Wikitude, proporrà una guida step-by-step per la definizione di uno standard, analizzando vari punti da seguire.

## 2.1 Steps Standard AR

1. **Analisi degli Stakeholder** [1]**:** Identificare gli stakeholder dell'ecosistema AR per assicurare che i loro interessi vengano rispettati.

2. **Analisi dell'architettura:** Analizzare le architetture AR da un punto di vista tecnico ed estrarre modelli comuni per assicurarsi che lo standard possa essere utilizzato da applicazioni AR già esistenti.

3. **Definizione dello scope:** Sulla base dell'architettura e dell'analisi degli stakeholder, identificare le parti architettoniche ad alto livello che sono nello scope dell'applicazione, e quelle che sono fuori portata.

4. **Definizione dei casi d'uso:** Identificare casi d'uso che sono già realizzati in applicazioni AR, e sviluppare casi d'uso che possono essere usati in futuro (se possibile). Questo assicura che una misura di fattibilità sia disponibile per gli standard già esistenti, per determinare se essi hanno già provveduto ad una funzionalità necessaria.

5. **Separazione verso il basso:** in base alla definizione dei casi d'uso, suddividere l'ambito dell'applicazione in parti più piccole (Working items), così da facilitarne l'analisi.

6. **Analisi degli standard esistenti:** Per ognuno dei Working item, analizzare e studiare se ci sono degli standard che coprono e già lavorano su essi.

7. **Colmare le lacune:** Colmare le lacune per i Working item che non sono ancora coperti (completamente) da standard esistenti, proponendo ad essi nuove funzionalità.

[1] Nello sviluppo di un prodotto software sono i soggetti interessati all'andamento e/o ai risultati finali del progetto, in quanto andamento e risultati possono avere effetti positivi o negativi su di loro. [Fig 2.1]

8. **Check di Fattibilità:** Per il risultato finale, controllare se i casi d'uso sono rispettati e le parti interessate sono in grado di implementare e/o utilizzare lo standard proposto.

### Stakeholders

Il primo passo quando si definisce uno standard AR è analizzare chi fa parte dell'ecosistema preso in considerazione e perchè; quale stakeholder è già influenzato da uno standard e come, possa trarre vantaggio da questo nuovo standard.

- **Content producer:** Un entità (persona o azienda) che crea contenuti per diversi client, per consegnarlo alla applicazione AR che si intende sviluppare.
- **Content publisher:** Il publisher riceve i contenuti dal produttore e li mette a disposizione per l'eventuale applicazione o browser.
- **AR Software Developer:** Gli sviluppatori software che sono tenuti a fare in modo che la loro app recuperi i dati messi a disposizione dal publisher (e.g. interrogazione al server).
- **Device Platform Provider:** Il provider fornisce l'ambiente sul quale poi gli sviluppatori andranno a creare l'applicazione AR, in genere attraverso la creazione di sistemi operativi o di nuove piattaforme. Google (Android), Apple (iOS), Nokia (Symbian).
- **Produttori Hardware:** I produttore HW sono necessari per il corretto svolgimento dello sviluppo SW, provvedono infatti a fornire chip, sensori, telecamere, batterie.
- **Software Publisher:** Il Software Publisher infine rende disponibile il SW sul mercato. In un ambiente mobile questo è rappresentato da varie applicazioni di store (Android Market, App Store, Ovi Store, App World etc.) ma può essere anche una pagina web che permette il download dell'app.

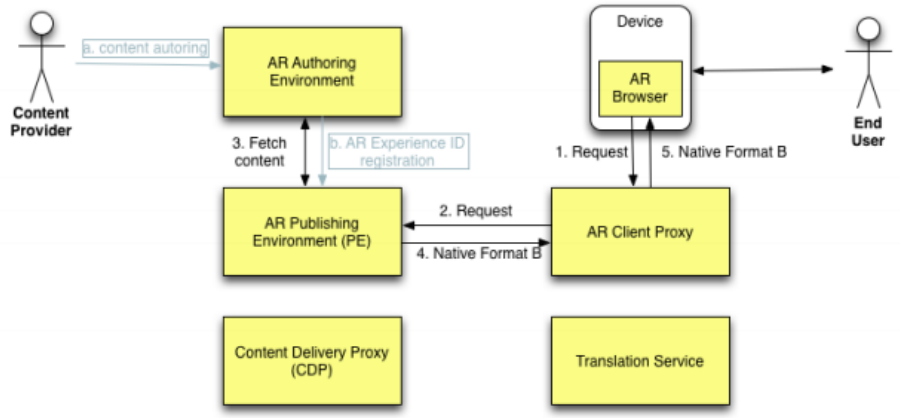


Figura 2.1: Struttura Stakeholder

## 2.2 Architettura

Nel 2004 T. Reicher, A. MacWilliams, e B. Bruegge in seguito allo studio di diverse applicazioni AR, hanno notato una architettura software generica e comune per questi sistemi. Nella loro pubblicazione "Study on software architectures for augmented reality systems", i tre ricercatori hanno fornito una primissima architettura di riferimento, a sua volta divisa in diversi sottosistemi.

### 2.2.1 Sottosistemi dell'Architettura

Spesso un'applicazione specifica appartiene ad un range più grande di applicazioni, chiamato dominio. Per ogni dominio, inoltre, ci sono dei requisiti funzionali e non, che sono mappati attraverso delle funzioni implementate a loro volta da sottosistemi [Fig 2.2]: *Application*, *Interation*, *Presentation*, *Tracking*, *Context* e *World Model*. Qui di seguito sono elencati i sei subsystem specificando i componenti interessati, le loro responsabilità e le relazioni attraverso le quali cooperano tra loro:

1. **Application subsystem:** "Application" è un sottosistema astratto che può essere considerato come il contenitore di tutto il codice specifico dell'applicazione, insieme ai dati di configurazione e ai contenuti.

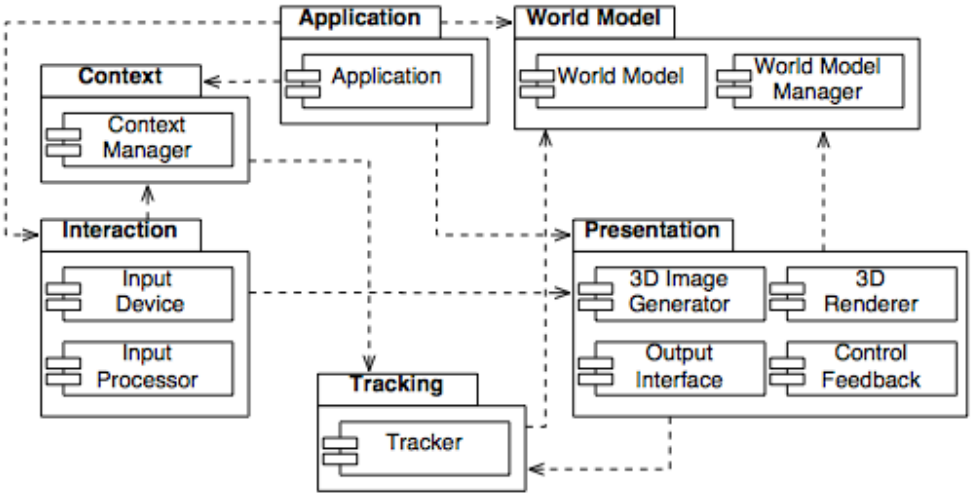


Figura 2.2: Sottosistemi dell'Architettura

2. **Interaction subsystem:** Il sottosistema Interaction raccoglie ed elabora qualsiasi input che l'utente fa o riceve. In questo sottosistema è doveroso fare una precisazione. Possiamo distinguere infatti le interazioni sia nello User Input subsystem, che raccoglie tutti gli input effettuati deliberatamente dall'utente (e.g. click, messaggi vocali o touch), e nello User output subsystem che invece controlla l'output del sistema mostrato all'utente (e.g. Immagini 3D, messaggi vocali o di testo).

3. **Presentation subsystem:** Il sistema provvede alla visualizzazione finale dell'ouput del sistema. Sia in formato 3D, sia in altri formati *media*, come semplici immagini in 2 dimensioni, testo, notifiche e suoni. Fanno parte di questo sistema tutti i componenti che provvedono alla fase di *registration* e di *rendering*.

4. **Tracking subsystem:** Il sottosistema "tracking" è uno dei più importanti di una architettura di realtà aumentata, si occupa appunto del tracciamento dell'applicazione, solitamente comunicando con altri

livelli del sistema come il *Context* o il *World model*, per una miglior esperienza AR.

5. **Context subsystem:** Il sottosistema context raccoglie differenti tipi di dati di contesto e li rende disponibili agli altri sottosistemi. Gli esempi possono essere dai più svariati dalle preferenze utente all'attività corrente, dalle informazioni di tracking alle risorse accessibili al sistema.

6. **World Model subsystem:** Nella concezione di realtà aumentata, l'utente non è statico, si sposta nel mondo reale, e dal mondo reale ottiene informazioni sugli oggetti fisici e sull'ambiente in generale. Tutte le informazioni provenienti da esso vengono registrate in un *world model* appunto, una struttura dove vengono raccolte informazioni su scene, oggetti virtuali, tracking e rappresentazioni degli oggetti reali, concependo così l'idea di un mondo virtuale immerso in quello fisico. A gestione del world model vi è un world model manager che controlla gli accessi ad esso.

### 2.2.2 Pattern

Sempre nella loro pubblicazione, Brugge, Reicher e MacWilliams, descrivono dei possibili approcci di implementazione dei singoli subsystem dell architettura, attraverso l'utilizzo di pattern. Verrà fornita una breve descrizione di ogni pattern, ordinati secondo il subsystem che andranno ad implementare.

#### *Application*

**Scene Graph Node pattern** Una *scene graph* è una struttura dati ad albero che cataloga gli elementi di una scena grafica. Formata da diversi nodi, ognuno dei quali rappresenta un oggetto grafico (e.g sfera), questa scene graph permette all applicazione di integrarsi perfettamente nell'ambiente reale; una sorta di necessità, in quanto è lo stesso utente a richiedere questo tipo di interazione con il mondo fisico. Questo pattern quindi collabora attivamente con un altro livello di subsystem, il world model, nel quale viene implementata la logica dell'applicazione grazie al codice grafico contenuto nei nodi.

**Scripting pattern** Questo pattern provvede al rapido sviluppo di un applicativo software grazie all'utilizzo di un linguaggio di scripting, più veloce a livello implementativo. Le parti dell'applicazione interessate sono solamente quelle che presentano vincoli di performance (e.g che operano sul *real time*) e solitamente sono scritte in linguaggi compilativi come il C++ allo stesso tempo fornendo però anche interfacce di scripting. Questa soluzione fornisce una più rapida release di prototipi, ma solitamente non si utilizza questo pattern per quei componenti che implementano importanti funzionalità. L'approccio script non è indicato infatti per le applicazioni complesse.

**Central Control pattern** Riscrive l'applicazione in un più alto linguaggio di programmazione, descrivendo esplicitamente cosa accade e quando questo accade.

**Tracking Rendering Loop pattern** Per semplificare lo sviluppo di applicazioni AR, alcuni librerie forniscono funzionalità di *basso-livello* necessari all'aggiornamento regolare della *view utente*. Il compito del pattern è quello di fornire punti di controllo che possono essere richiamati all'interno del ciclo di aggiornamento e conseguentemente modificare la scena da presentare.

**Web Service pattern** Grazie a questo pattern l'esperienza AR viene intesa come un contenuto multimediale soggetto alla classica relazione client-server. Infatti il flusso di controllo di questo pattern è posto su un server web e attuato all'interno di un servizio web. Questo servizio web risiede in un determinato indirizzo web e la risposta del servizio è fornita da un client web. Se la risposta contiene contenuti di Realtà Aumentata allora interviene una componente di visualizzazione AR che si occupa di mostrare il contenuto in esame.

**Multimedia Flow Description pattern** Per semplificare la creazione di contenuti AR nella nuova applicazione, ci si può appoggiare a dei linguaggi di markup ad alto livello, che forniscono componenti e concetti in grado di aiutare a creare nuovi contenuti in modo rapido. Un esempio classico è una sequenza di scene AR e altri documenti, a supporto di uno

scenario che illustra passo passo le mansioni per lavoratori. Solitamente i linguaggi più diffusi a livello commerciale sono l'HTML5 e l'XML.

***Interaction*** Questa sezione si sofferma sull'uso di pattern fini alla gestione delle tecniche d'interazione del sistema e su come vengono gestiti gli input utente.

**Handle-In Application pattern** Mantiene la gestione degli input nell'applicazione all'interno del codice applicativo, con un esplicito riferimento ai tipi di input device. Rimane il modo più semplice e intuitivo di gestione degli input.

**Networked Input Devices pattern** Fornisce un layer astratto specifico per gli input device, e si occupa di fornire anche una descrizione di come questi input possono essere combinati. Solitamente il pattern viene affiancato dall'uso del middleware per trovare dinamicamente nuovi input device.

**Use Browser Input Functions pattern** Con questo pattern si tende a sfruttare le potenzialità e funzionalità di una applicazione esterna (e.g. Browser AR) utilizzata solitamente per il rendering degli oggetti virtuali. Si sfrutta la possibilità di mandare eventi da parte dell'applicativo scelto, attraverso la sua l'interfaccia utente. Semplici click o gesti dell'utente all'interno dello scenario in esecuzione possono essere interpretati come veri e propri segnali di input.

**Modality Fusion pattern** Modalità di input individuali, come ad esempio gesti e segnali audio da parte dell'utente, vengono uniti e interpretati come un unico singolo evento.

***Presentation*** I pattern descritti in questa sezione si occupano principalmente della visualizzazione finale dell'output, facendo particolare attenzione al rendering delle immagini 3D o di quelle 2D.

**3D Markup pattern** Il pattern 3D Markup si occupa di visualizzare un contenuto multimediale (grafico) 3D attraverso un linguaggio ad alto livello come il VRML. Parallelo all'impiego di quest'ultimo, viene affiancato anche l'utilizzo di interfacce esterne integrate nel VRML, come l'External Authoring Interface (EAI), per modificare la scena e impostare i *viewpoint* secondo i criteri determinati dalle informazioni del tracking.

**Low-Level Graphics Primitives pattern** Parallelamente al precedente linguaggio High-Level ora ci troviamo ad utilizzare una libreria grafica 3D per astrazioni low-level, ovvero rendering di facili costrutti 3D. Motori grafici come OpenGL risultano particolarmente flessibili ed efficaci per quelle astrazioni grafiche non particolarmente high level required; se poi integrata con le informazioni sul tracking, la scena può essere renderizzata con la corretta direzione e distanza di visualizzazione.

**Video Transfer pattern** Un client recupera un flusso video attraverso una o due telecamere *head-mounted*, lo codifica (ad esempio in MPEG 2) lo comprime, e lo trasferisce ad un server. Il server a sua volta decomprime le immagini video, le elabora (calcolando la posizione e l'orientamento della fotocamera), applica algoritmi di AR se necessario, ricodifica e ricomprime le immagini. Quest'ultime a loro volta vengono inviate al client, per poi essere decompresse e visualizzate nuovamente sul display utente iniziale.

**View Manager pattern** Gestisce l'interfaccia utente con un componente centrale che filtra o mette in coda le richieste di output.

**Adaptation to Error Level pattern** Livello molto importante per la gestione degli errori, adatta infatti l'output finale (presentazione) al livello di errore registrato dagli altri subsystem, in particolare da quello di tracking. Per esempio, se il tracking non è preciso o supera una certa soglia, il pattern cambia la visualizzazione finale da 3D a 2D, limitando gli errori grafici che ne conseguono.

**Multiple Viewers pattern** Fornisce un livello astratto specifico per diversi tipi di formati output (AR, testi, audio) che è in grado di gestire la visualizzazione dei conseguenti documenti/informazioni in quel particolare formato.

**2D/3D Mapping pattern** Mappa informazioni grafiche 2D in scenari finali 3D, così da poter integrare autonomamente applicazioni 2D in uno ambiente fisico tridimensionale.

***Tracking*** In questa sezione vengono descritti particolari tecniche e approcci per la gestione del servizio di tracking, e per l'interazione dei sensori o device con il sistema.

**Inside-out Tracking pattern** L' inside-out tracking (o Tracking Interno) è una tecnica grazie alla quale un sistema di tracciamento che opera a livello user, traccia la posizione e l'orientamento dell'utente. Classico esempio è il tracking effettuato da un telecamera posta solitamente sul capo dell'utente che traccia le sue coordinate e direzioni nell'ambiente circostante.

**Outside-in Tracking pattern** Diversamente da quanto descritto precedentemente, l'Outside-in Tracking (o Tracking Esterno) effettua il tracciamento attraverso dei device appositi all'interno della ambiente fisico, per poi restituire in un secondo momento le informazioni al sistema utente.

**Tracking Server pattern** Questo pattern gestisce le numerose richieste di tracking spostando il carico delle attività computazionali a lato server, fornendo al client solo il risultato finale.

**Networked Trackers pattern** Ogni device o sensore di tracciamento, viene incapsulato in una specifica interfaccia, la quale andrà a comunicare con uno specifico middleware (e.g CORBA) e a registrare il sensore nella rete. Il fine ultimo del middleware sarà quello di mettere in comunicazione dinamica gli stessi sensori con i componenti che ne richiedono l'uso, così come accade nel sottosistema Interaction.

**Direct Access pattern** Grazie all'accesso diretto, si può accedere ai sensori di tracking direttamente grazie a particolari drivers per il sistema operativo.

***World Model*** Nella sezione World Model verranno descritti le varie tecniche di accesso e soprattuto salvataggio delle informazioni e dei dati, per un rapido riutilizzo quando richiesto.

**Scene Graph Stream pattern** Con un tool grafico uno sviluppatore di contenuti crea il modello di una scena virtuale. Questo pattern è particolarmente utile quando queste scene devono essere memorizzate. Ad esempio viene largamente utilizzato in un contesto industriale, quando le scene create con strumenti CAD possono essere semplificate e riutilizzate. Inoltre la descrizione di una scena solitamente viene salvata nel file system e data successivamente al sistema AR per l'elaborazione.

**Object Stream pattern** Lo step successivo allo Scene Graph Stream Pattern precedentemente descritto, è l' Object Stream pattern, ovvero la serializzazione e il salvataggio su disco di particolari oggetti in diversi formati o di intere scene. Alla prossima occasione quando verrà richiesta qualche informazione precedentemente salvata, questa verrà ricarica direttamente dal disco o deserializzata.

**Dynamic Model Loading pattern** Lo step finale del loading data, iniziato con i precedenti due pattern, è rappresentato dal Dynamic Model Loading, ovvero una gestione del caricamento delle informazioni attraverso un Database vero e proprio. Parte delle informazioni sono soprattutto riguardante la grafica 3D degli oggetti virtuali o riguardante il monitoraggio del tracking. Queste informazioni vengono accedute grazie alla richiesta da parte del sistema al database attraverso delle query, ottenendo in risposta il contenuto AR. Questo pattern si adatta particolarmente al caso specifico mobile, ovvero quando si vuole sfruttare l'accesso dinamico a un database online.

**Marker File pattern** All'avvio del sistema o ogni qualvolta quest'ultimo viene a rapportarsi con un nuovo ambiente, è utile e preferibile creare un *Marker File* (sostanzialmente un file di configurazione) nel quale vengono salvate le caratteristiche e i tipi di marker che i sensori o i tracker device devono ricercare.

***Context*** Il context subsystem provvede al raccoglimento e alla distribuzione dei dati attraverso i diversi sosttosistemi del sistema centrale, qui di seguito l'elenco dei vari pattern che agevolano questa operazione.

**Blackboard pattern** I componenti *producer* scrivono le informazioni su una parte centrale (*Blackboard*), le quali vengono successivamente accedute e lette dai componenti *consumer*. I dati a questo punto vengono processati e potrebbero venire anche sostituiti con informazioni *high-level*.

**Repository pattern** Piuttosto similare al precedente pattern, questo pattern permette il salvataggio delle informazioni in un deposito comune (*repository*), nel quale i producer possono scrivere e dal quale i consumer possono leggere. A differenza però del Blackboard, qui le informazioni possiedono ognuna un proprio indirizzo di accesso, accessi gestiti dalla stessa repository.

**Publisher/Subscriber pattern** Con questa configurazione viene superata l'idea di un luogo condiviso dove salvare e immagazzinare informazioni. I producer infatti si connettono a un centro di messaggistica come publisher, e a loro volta i consumer come subscriber. I publisher scrivono le nuove informazioni in un canale apposito che andrà a consegnare direttamente al subscriber iscritto ad esso, il dato corretto.

## 2.3 Piattaforme AR

### 2.3.1 Vuforia

La piattaforma Vuforia utilizza tecniche di riconoscimento immagine computer-vision superiori, stabili e tecnicamente efficienti, mettendo a disposizione la più ampia serie di caratteristiche e funzionalità, e offrendo agli sviluppatori la libertà di estendere le loro visioni senza limitazioni tecniche. Con il supporto per iOS, Android e Unity 3D, la piattaforma Vuforia consente di scrivere una singola applicazione nativa che può raggiungere la maggior parte degli utenti nella più ampia gamma di smartphone e tablets. Tecnologie fondamentali che fanno dell'SDK Vuforia (sviluppato dall'azienda Qualcomm) un top di gamma per la programmazione mobile, sono l'Extended Tracking, motore

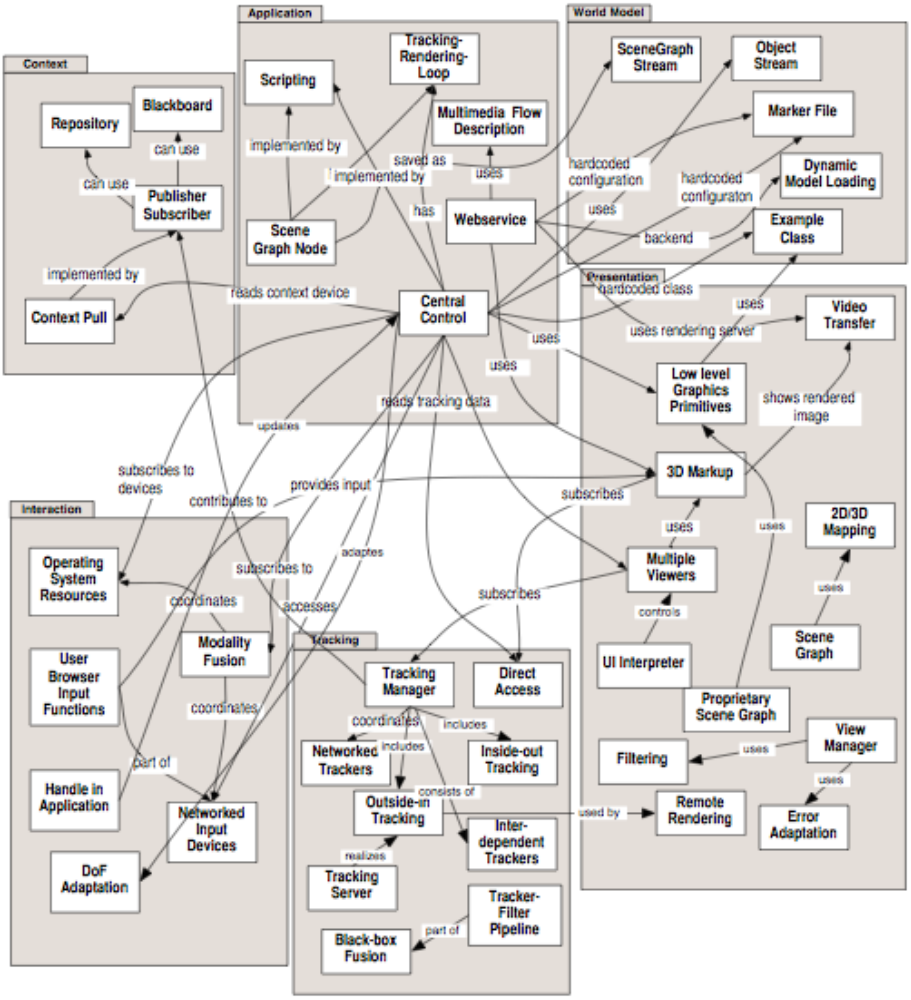


Figura 2.3: Struttura Pattern implementativi

di tracciamento che si basa sul riconoscimento visivo di immagini reali e tridimensionali (e non limitandosi a semplici marker), e il Cloud Storage, ovvero la possibilità di salvare le immagini target su un database Cloud a cui si connette il servizio, in modo da risparmiare spazio di storage locale.

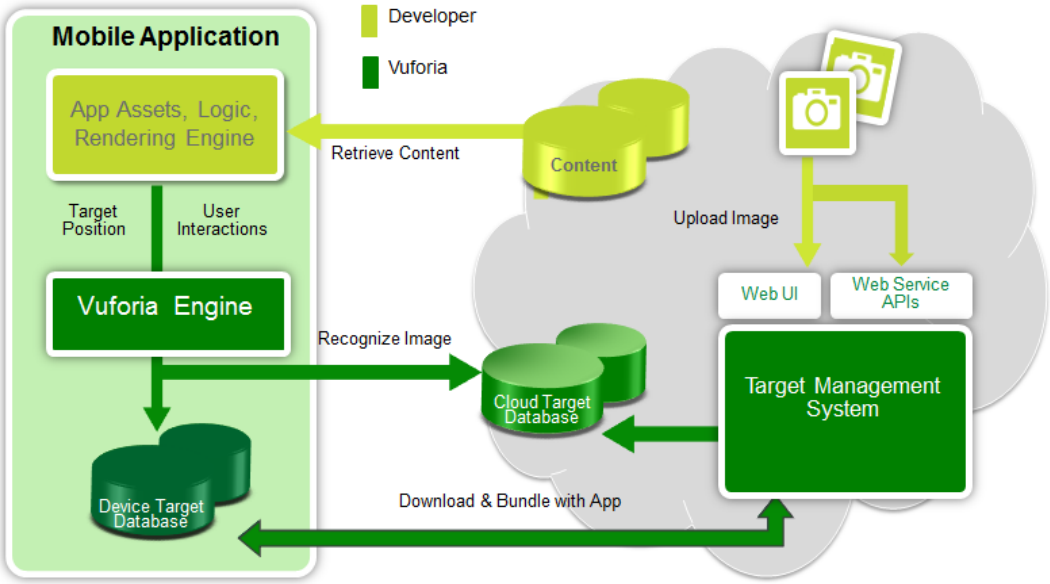


Figura 2.4: Vuforia Platform

### 2.3.2 D'Fusion

D'Fusion è la soluzione commerciale di Realtà Aumentata più usato al mondo. Disponibile per un gran numero di dispositivi e sistemi operativi, consente lo sviluppo di soluzioni AR per un'ampia varietà di usi, come il marketing interattivo, eventi live e applicazioni industriali. Oltre a fornite un SDK fornisce anche una suite di prodotti che agevolano lo sviluppo, garantisce il supporto ad android/iOS oltre che ad Adobe Flash e Unity 3D. La suite si compone di 4 prodotti:

- *D'Fusion Mobile* Appositamente studiata per il settore mobile offre il supporto per android e iOS.
- *D'Fusion Home* È studiata per la produzione via web o su supporti ottici, inoltre fornisce la possibilità di integrazione con Facebook.
- *D'Fusion Pro* Utilizzato per la realizzazione di applicazioni professionali, aggiunge il supporto a video HD, videocamere multiple, videocamere infrarossi e sensori specifici.

- *D'Fusion Studio* È la soluzione gratuita che utilizza la versione free dell'SDK, consente la creazione di modelli 3D e la loro integrazione nell'ambiente tramite l'utilizzo di tracker.

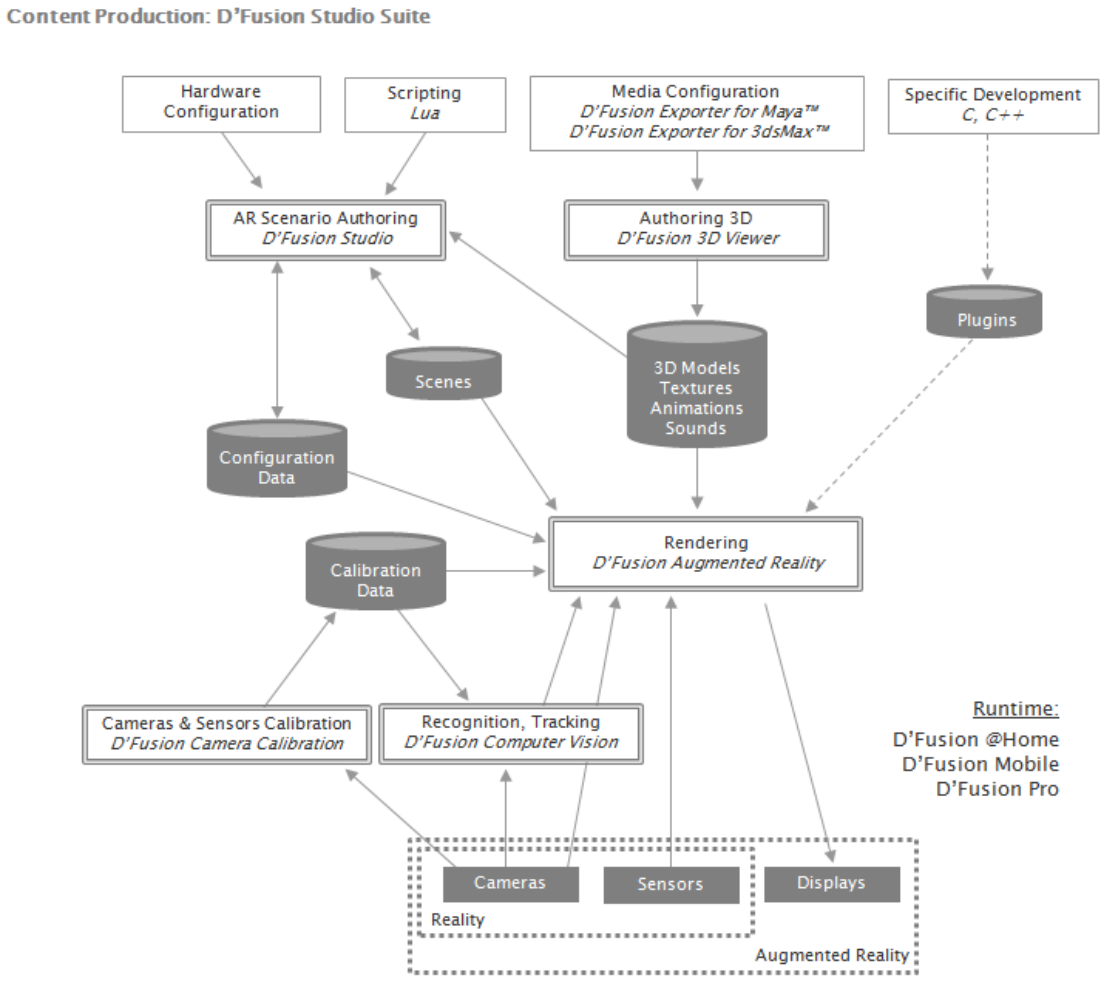


Figura 2.5: D'Fusion Platform

### 2.3.3 Wikitude

Wikitude è stata votata come miglior AR Browser dal 2012, dando inoltre anche l'opportunità per sviluppatori di creare applicazioni utilizzando l'SDK proprietario. Non solo, una particolarità di Wikitude è quella di poter creare contenuti (chiamati "Worlds") utilizzando le interfacce e i linguaggi stessi di Google Maps e Google Earth, ovvero il Keyhole Markup Language (KML) e l'Augmented Reality Markup Language (ARML), entrambi basati su XML. Gli sviluppatori possono inoltre operare su più piattaforme quali Android, iOS e Blackberry, creando contenuti (texts, sound e anche immagini 3D) utilizzando linguaggi come JavaScript, CSS e HTML5. Il suo punto di forza infatti è proprio quello di essere basato su linguaggi di programmazione e tecnologie web, rendendo il sistema molto versatile e integrabile per nuove piattaforme. Tra le altre, un altra grande caratteristica di Wikitude è, che a differenza degli altri browser AR, i contenuti una volta creati sono ospitati su server proprietari.

### 2.3.4 String

Framework per ios, rende possibile realizzare applicazioni in realtà aumentata che materializzano e animano oggetti tridimensionali in sovrapposizione a marker naturali. Ultimamente sono state avviati anche progetti di sviluppo per una tecnologia Tracking Natural Feature, mantenendo sempre la propria linea ufficiale di portabilità multipiattaforma, ovvero servizio di AR unicamente per sistemi iOS; questo ha reso String un vero e proprio brand esclusivo per un unico e mirato target di mercato. I motori grafici e di sviluppo 3D sono i più efficienti e sviluppati del settore, basati su un linguaggio Unit3D e OpenGL riescono a tracciare 10 immagini simultaneamente, rendendo la qualità immagine 3D del servizio AR il vero e proprio punto di forza dell'azienda, che mira appunto maggiormente ad un settore pubblicitario e ludico (mobile).

### 2.3.5 Metaio

L'azienda Metaio viene fondata in Germania, precisamente a Monaco, nel 2003 da Thomas Alt e Peter Meier. Ben presto le potenzialità della SW house vengono riconosciute, collaborando prima ad un progetto con Volkswagen e poi ricevendo un sovvenzionamento da parte dello stato. I primi risultati

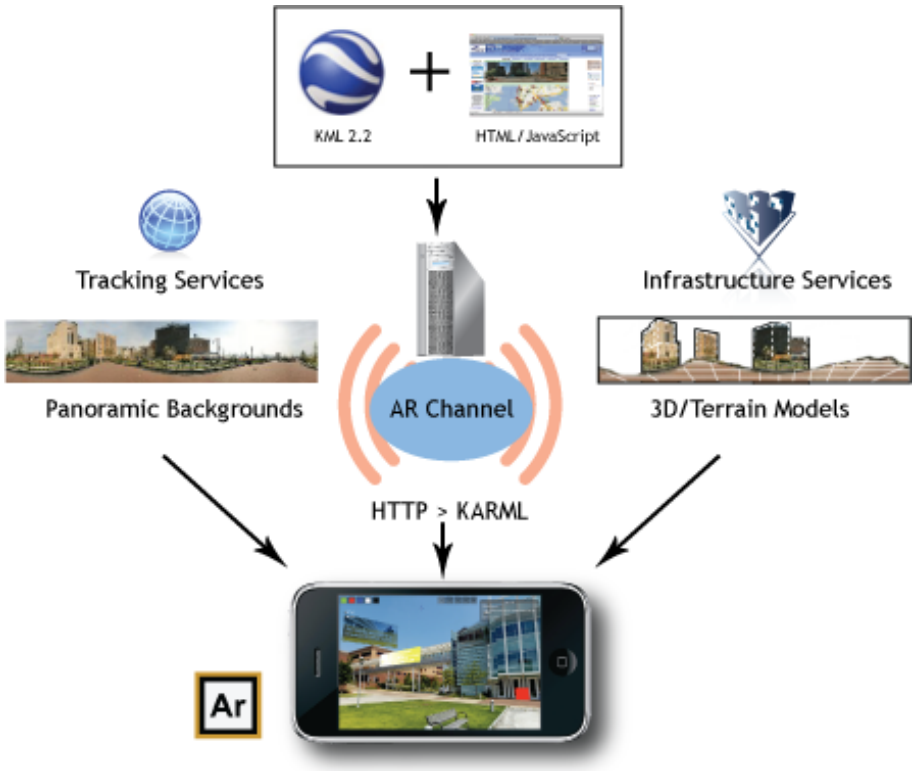


Figura 2.6: Wikitude Platform

non tardano ad arrivare, la società tedesca rilascia infatti nel giro di pochi anni una delle prime applicazioni AR chiamata *KPS Click and Design* nel 2005 e diventa leader (tutt'ora) del settore augmented reality, mettendo a disposizione gratuitamente (e non) diversi prodotti per lo sviluppo. Ad oggi Metaio può vantare anche l'app browser AR maggiormente diffusa sul mercato: Junaio.

# Capitolo 3

# Framework e SDK: caso di studio METAIO

In questo capitolo riprenderemo alcuni termini e concetti per un ulteriore descrizione di una possibile architettura AR, parallelamente finalizzata all'analisi di un possibile *Augmented Reality Framework*, concluderemo discutendo il caso di studio che la tesi si è prefissata di analizzare, il progetto Metaio e relativo SDK.

## 3.1 Framework

Alla base di ogni sviluppo di applicazioni AR ci sono sempre le solite problematiche architetturali a cui andare a far fronte. Ogni applicazione infatti deve provvedere a modo suo ai vari problemi discussi precedentemente, che spaziano dal Tracking, al multimedia, all'interfaccia grafica fino alla gestione degli input hardware. Tutto questo attraverso lo sviluppo di un framework specifico, solitamente proprietario. [Fig 3.1] Se infatti è vero, che gli SDK e la API delle diverse aziende offrono diverse funzionalità che puntano a sfruttare al meglio le potenzialità del framework prioritario, è anche vero che questi vengono accomunati da scopi e caratteristiche simili. Il Framework sviluppato dalla Metaio è un framework modulare che include vari componenti al suo interno:

- *Capturing Component.*
- *Rendering Component (contents creation).*

- *Tracking Component.*
- *Sensor Interface Component.*

Il livello superiore è formato da un layer di interfaccie che provvedono all'interazione tra le applicazioni ultime e i 4 componenti precedentemente citati, nascondendo così all'utente i dettagli dell'implementazione di quest'ultimi. L'SDK è compatibile con tutte le più importanti piattaforme di sviluppo (Android, IOS, Unity 3D, Windows), e supporta allo stesso modo i più importanti linguaggi di programmazione (Java, Obj-C, C++) affiancando ad essi un linguaggio Javascript proprietario chiamato *AREL(Augmented Reality Experience Language).*

## 3.2 SDK

L'SDK dell'azienda tedesca è uno dei pochi attualmente free disponibili in rete, risultando uno dei migliori sul mercato per quanto riguarda prestazioni e personalizzazione. Attraverso questo SDK si ha a disposizione la tecnologia necessaria che una normale piattaforma metterebbe a disposizione per un completo sviluppo di AR app. Qui di seguito una breve descrizione dei punti più importanti su cui si basa l'SDK, che verranno poi argomentati più avanti.

**Tecnologia**

- L'unico SDK che offre la capacità di riconoscere e tracciare qualsiasi immagine, oggetto o ambiente del mondo reale;
- L'unico SDK che offre *continuous* e *client-based visual search*;
- L'unico SDK sia con 3-D *point cloud tracking* e sia con tecnologia *SLAM*, per un tracking markless in tempo reale.

**Sviluppo**

- Consente lo sviluppo nativo di app su iOS, Android e Windows PC.
- La struttura modulare di SDK permette di scrivere una sola volta il codice AR, per poi distribuirlo su più piattaforme ( Android, iOS, PC).

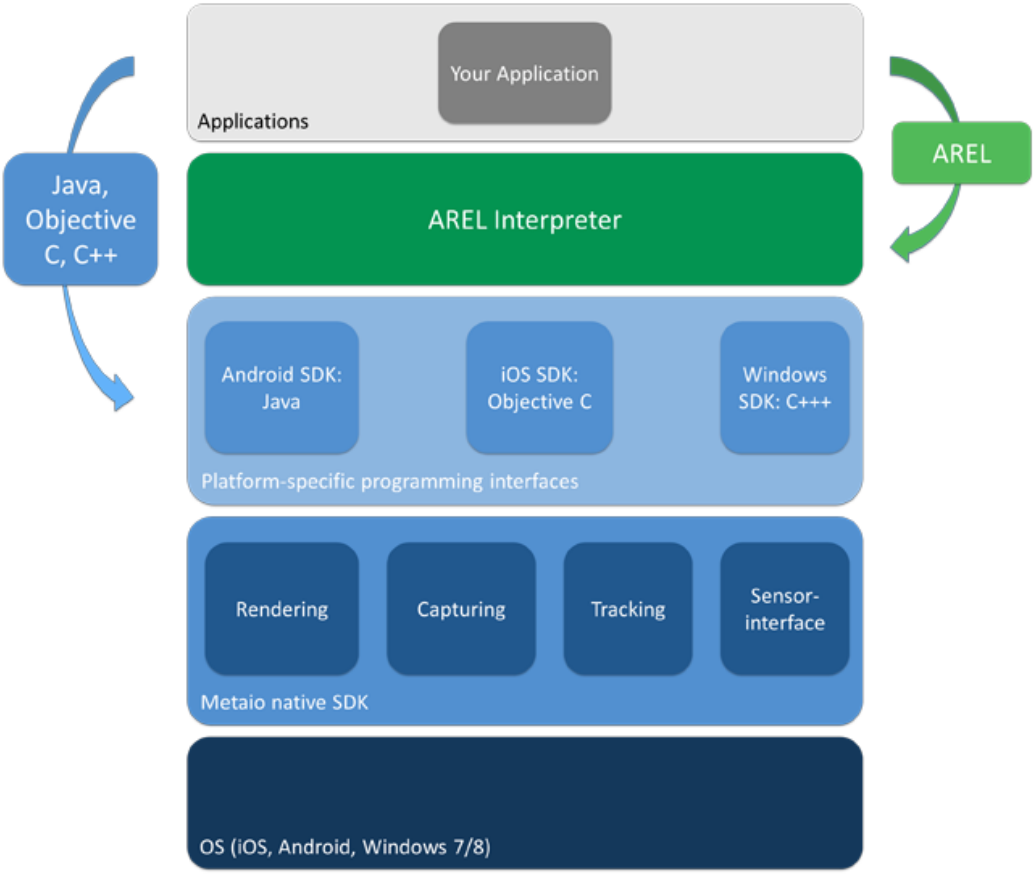


Figura 3.1: Metaio Framework

- Supporta lo sviluppo HTML5 e Javascript.

**Contenuto**

- Supporta client offline e memorizza il contenuto AR o sull'app o direttamente sul cloud.
- L'SDK è dotato di un proprio motore di rendering, non è necessario quindi acquistarne uno diverso.
- Supporto principali formati 3D.

## 3.3 Prodotti

### 3.3.1 Visual Search

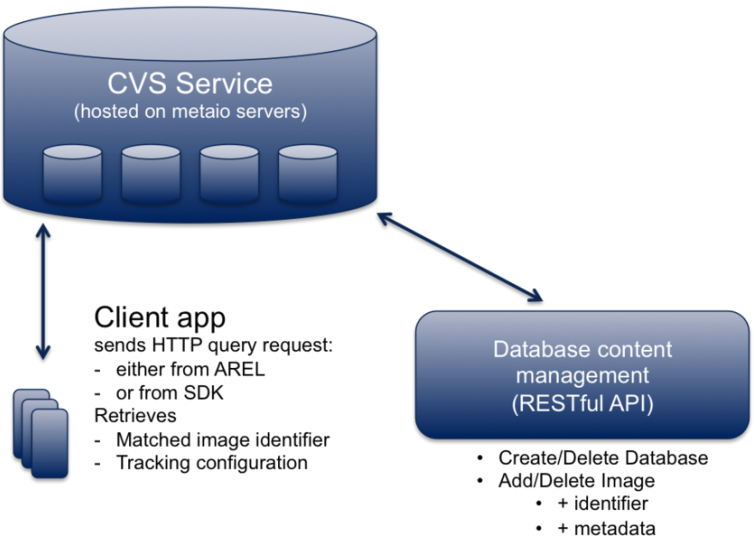


Figura 3.2: Metaio CVS

Continuous Visual Search (CVS) [Fig 3.2] è una tecnologia e servizio offerto da Metaio con il quale è possibile creare il proprio database di immagini tracker, e velocizzare quindi il riconoscimento di esse e la conseguente operazione di tracciamento. CVS è utile per creare progetti in cui centinaia di modelli possono essere gestiti automaticamente tramite script. Dal momento che il riconoscimento avviene sul server, il client non deve effettuare

alcun sforzo di local storage, in quanto tutte le informazioni sono ospitate su server proprietari.

### 3.3.2 ToolBox

ToolBox è un altro importante strumento di creazione e modifica contenuti messo a disposizione dalle Metaio. Grazie a ToolBox si possono infatti generare o editare mappe di tracciamento 3D partendo unicamente dagli elementi dell'ambiente circostante. Utilizzando la fotocamera del device viene infatti fatta una scansione della porzione di spazio che si vuole rendere tracciabile e attraverso tecniche di computer grafica vengono posizionati dei breakpoint che aiutano a delimitare angoli, contorni e superfici degli oggetti fisici.

### 3.3.3 Arel

AREL (Augmented Reality Experience Language) è un linguaggio di scripting a supporto dell'SDK Metaio, basato su tecnologie Web come HTML5 , XML e JavaScript [Fig 3.3]. Simile ad un normale sito web, un'esperienza AR basata su AREL è costituita da una parte statica, AREL XML, che definisce tutti i contenuti e link che si vogliono inserire, e da una parte dinamica, AREL JavaScript, che definisce le interazioni e comportamenti dei singoli oggetti o di un'intera scena. Naturalmente non è necessario definire tutti i contenuti staticamente in un file XML, ma si può anche caricarlo nel codice JavaScript quando necessario, proprio come si farebbe con l'SDK dal codice nativo. Arel inoltre sposa a pieno l'idea di interoperabilità e multipiattaforma che gli standard AR tanto ricercano; si è in grado infatti di creare una piattaforma indipendente da linguaggi di programmazione specifici (Objective-C, Java, C++). Più nel dettaglio:

- La parte XML definisce quali contenuti devono essere caricati e in che modo, come i modelli 3D o immagini pubblicitarie; essa definisce infatti le proprietà iniziali di questi oggetti come il formato, il sistema, le trasformazioni, le coordinate ecc.

- Il layer HTML5 fornisce invece l'interfaccia utente e interagisce con il ponte JavaScript instaurato con l'SDK Metaio; è possibile progettare questa parte grafica come qualsiasi sito web mobile, con l'eccezione che

lo sfondo è trasparente per permettere all'utente di vedere la fotocamera e qualsiasi contenuto AR reso sopra l'immagine della fotocamera.

- Tutte le callback e chiamate provenienti direttamente dal'SDK sono inoltrate alla logica dell'applicazione JavaScript, in modo da poter reagire a queste in maniera dinamica.

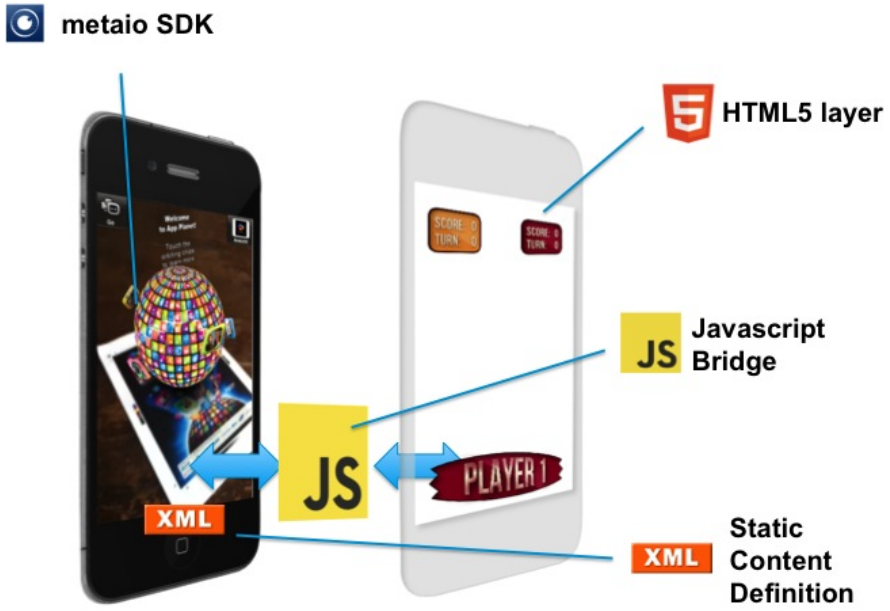


Figura 3.3: Arel Structure

## 3.4 Sviluppo di un App Metaio: Hello World App

Abbiamo capito che l'SDK Metaio permette uno sviluppo di applicazioni AR multipiattaforma, e mette a sua volta a disposizione i vari ambienti per il relativo linguaggio di programmazione. Qui di seguito verranno fornite le istruzioni e le caratteristiche per lo sviluppo di un applicazione su Android e iOS.

### 3.4.1 Attività

Dopo aver correttamente settato l'ambiente di sviluppo, passiamo ad analizzare una primissima applicazione di Realtà Aumentata, denominata in maniera classica "HelloWorld". Fine ultimo dell'app è quello di mostrare un immagine 3D sovrapposta ad un marker correttamente ripreso e riconosciuto dalla fotocamera (per questo motivo le funzioni di simulazioni device Android o iOS sono da disabilitare). Una volta ottenuto il codice d'esempio direttamente dal sito produttore di riferimento per gli sviluppatori (dev.metaio.com), iniziamo a dare un occhiata alle cartelle forniteci e a cercare quali attività o elementi specifici del codice avremo bisogno per preparare l'applicazione demo. Passiamo alla cartella *TutorialHelloWorld/Assets* in iOS o *assets/TutorialHelloWorld/Assets* se si utilizza la piattaforma Android. Analizziamo ora i seguenti file:

- **metaioman.md2** è la geometria 3D o semplicemente il modello scelto per una primissima rappresentazione di Augmented Reality. Il formato è in MD2 ma l'SDK è in grado di caricare anche modelli OBJ e FBX.

- **metaioman.png** è una Texture. Abbiamo anche bisogno di una texture per fornire al modello un aspetto migliore. Quindi usiamo un'immagine PNG che deve avere lo stesso nome del modello.

- **metainmantarget.jpg** questa è l'immagine scelta per il tracking, che porterà ad una corretta visualizzazione del modello in 3D desiderato.

- **TrackingDataMarkerlessFast.xml** è un file XML che contiene tutte le configurazioni e caratteristiche per il tracking. È essenziale nel progetto di un applicazione.

- **arelConfig.xml**, **arelTutorial.html** e **arelGlue.js** sono i file necessari per eseguire questo esempio anche in HTML5 e JavaScript senza una sola riga di codice nativo.

### 3.4.2 Implementazione

Ora, passiamo a dare uno sguardo d'insieme al codice. L'implementazione avviene all'interno di una sola classe chiamata *HelloWorldViewController* per iOS e *TutorialHelloWorld* per Android. Questa classe è rispettivamente una subclass della *MetaioSDKViewController* e della *MetaioSDKViewActivity*, che contengono a loro volta il numero delle chiamate di sistema necessarie. Tra le altre annotiamo:

- **onGeometryTouched/onTouched**: chiamata in reazione all'interazione fisica touch di una geometria sul display.
- **startCamera()**: avvia la fotocamera impostata di Default.
- **onSurfaceCreated()**: fondamentale per la creazione dei contenuti. Questa chiamata si poggia direttamente sulla libreria grafica OpenGL, che si occuperà di settare e renderizzare l'interfaccia utente.

I due metodi fondamentali per entrambe le piattaforme rimangono il *viewDidLoad()* per iOS e *LoadContent()* per Android, che si occupano di tre aspetti essenziali.

1. **Assegnare file Tracking di riferimento all'SDK**.

   Dove trackindatafile è il path per il file tracking di configurazione in formato XML, nel quale viene specificata l'immagine campione, il tipo di tracking adottato (optical, markerbased, markerless..) e le modifiche attuate sui parametri (subtype fast, numero di frame tracciabili, impostazioni fotocamera..) Tutte specifiche che verranno analizzate nella prossima sezione riguardante il Tracking.

```
metaioSDK.setTrackingConfiguration(trackingDataFile);
```

Figura 3.4: Android Tracking sample

```
m_metaioSDK->setTrackingConfiguration(trackingDataFile);
```

Figura 3.5: iOS Tracking sample

2. **Caricare il modello 3D da visualizzare**. Dove IGeometry è l'interfaccia che gestisce la creazione di oggetti 3D nell'SDK e dove metaioManModel è il path del file che contiene il modello 3D da visualizzare (le texture verranno caricate automaticamente).

```
// Loading 3D geometry
mModel = metaioSDK.createGeometry(metaioManModel);
if (mModel != null)
```

Figura 3.6: Android geometry creation

```
virtual IGeometry* createGeometryFromImage(const std::string& filepath, const bool displayAsBillboard = false) = 0;
```

Figura 3.7: iOS geometry creation

3. **Settare le dimensioni, la rotazione del modello e l'animazione**. Una volta che il modello è stato caricato con successo possiamo passare a modificarlo a nostra discrezione: in modo simile si possono ottenere gli stessi risultati di traslazione e rotazione invocando semplicemente setTraslation() e setRotation().

```
theLoadedModel->setScale(metaio::Vector3d(0.8,0.8,0.8));
```

Figura 3.8: iOS setScale

```
public void onAnimationEnd(IGeometry geometry, String animationName)
{
    // Play a random animation from the list
    final String[] animations = {"meow", "scratch", "look", "shake", "clean"};
    final int random = (int)(Math.random()*animations.length);
    geometry.startAnimation(animations[random]);
}
```

Figura 3.9: Android animation

### 3.4.3 Ciclo di vita AR App Android Metaio

Ogni applicazione Metaio (caso Android) è formata da una classe che si estende da quella padre ARViewActivity, a sua volta contenuta nel progetto MetaioSDK, dal quale l'applicazione dipende. ARViewActivity è nel pacchetto com.metaio.SDK nella directory/src. Le callback importanti su Android (in ordine di attivazione) che gestiscono l'attività AR sono:

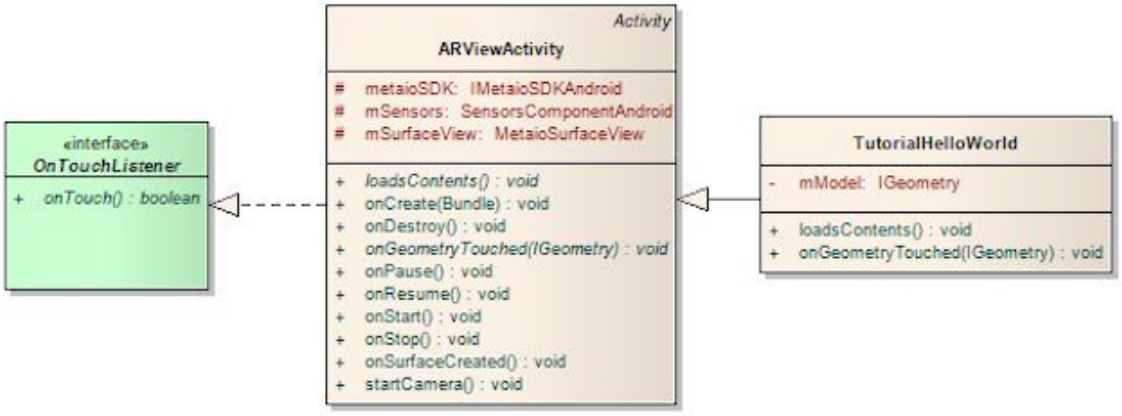


Figura 3.10: Schema UML di ARViewActivity e TutorialHelloWorld

- **onCreate():** Metodo della classe ARViewActivity chiamato alla creazione della prima activity; pertanto, la maggior parte della inizializza-

zione della applicazione dovrebbe essere invocata qui. Per questo sia i sensori che l'istanza Metaio SDK sono creati in questa callback.

- **onStart():** Il metodo successivo alla onCreate() (o alla onRestart() ovvero quando l'activity è stata sospesa) è l' onStart(), che si occupa di creare e settare la GLSurfaceView e far avviare il processo di capturing della fotocamera.

- **onSurfaceCreated():** Questa callback si occupa di dare al via all'ultima parte della applicazione, la creazione contenuti. Viene chiamata al momento della creazione della GLSurfaceView, e ad essa mette in coda attraverso la funzione queueEvent(), i metodi per la content creation.

- **loadsContent():** Metodo fondamentale per il tracking e il render dei contenuti. Mentre le callback fino ad ora citate provenivano dalla classe padre ARViewActivity , questa è specifica per ogni applicazione e caso d'uso. Qui vengono differenziati infatti i vari approcci di tracking e content creation, grazie ai metodi getAssetPath(), setTrackingConfiguration, e createGeometry().

- **onTouch():** In questa callback viene gestita l'interazione con il contenuto aumentato a seguito di una risposta ad un evento Touch, attraverso il metodo onGeometryTouched; il flusso di controllo viene avviato nella classe padre con la chiamata a onTouch(), metodo di ARViewActivity che si occupa appunto di richiamare onGeometryTouched. In ogni classe specifica verrà poi fatto l'override di quest'ultimo, in base alle specifiche date.

- **onPause/onResume:** Punto in cui l'istanza Metaio SDK e la surfaceViewGL sono in pausa / resume.

- **onStop:** Callback che viene chiamata quando la view non è più visibile all'utente; da qui si procede con onRestart() o onDestroy().

- **onDestroy:** Con questa callback è si effettua l'arresto dell'applicazione, e si elimina sia l'istanza Metaio SDK e sia i sensori;

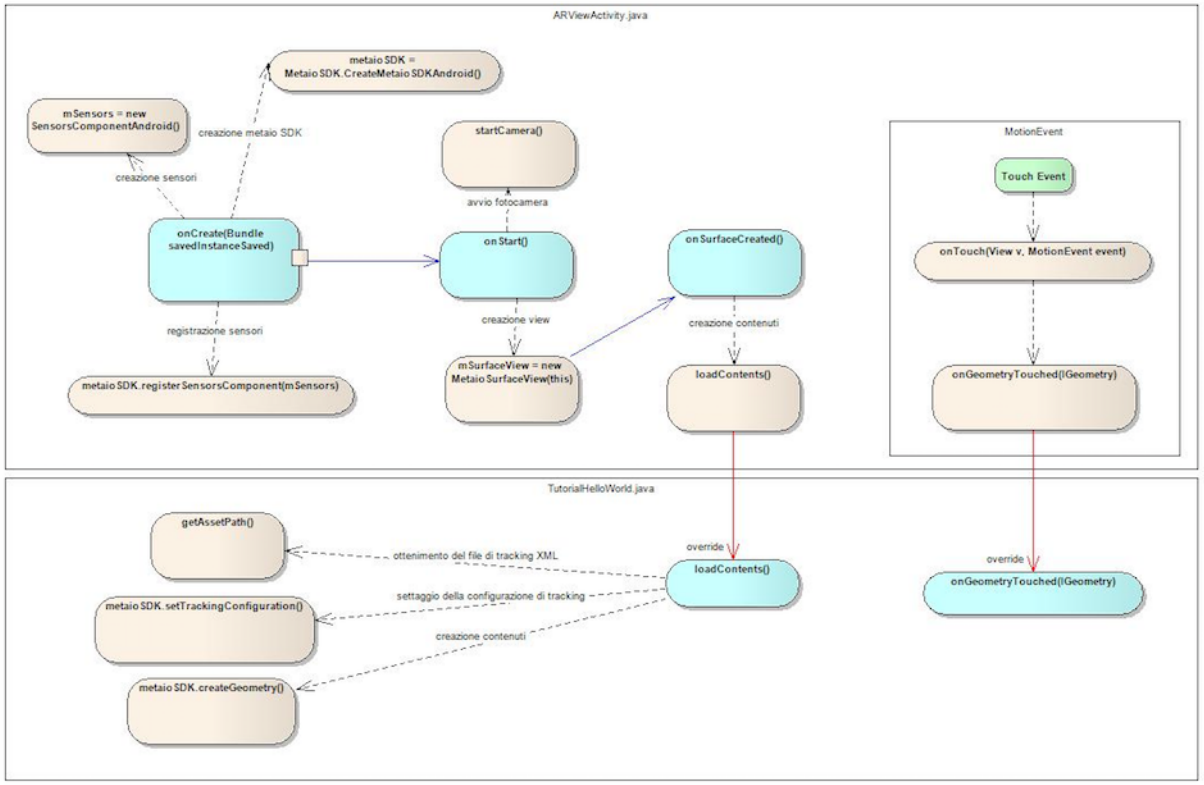


Figura 3.11: Ciclo di vita dei metodi di un app Metaio

## 3.5 API References

La documentazione ufficiale presenta innanzitutto due interfacce principali per lo sviluppo SW: *IMetaioSDK* e *IGeometry*. Qui si seguito una descrizione delle principali funzioni raggruppate secondo campo d'applicazione.

### 3.5.1 Tracking

- **setTrackingConfiguration:** Imposta una particolare configurazione di tracking partendo da un file XML.
- **startInstantTracking:** Provvede ad avviare una specifica configurazione di tracking.
- **pause/resume Tracking:** Avvia/Ferma il tracciamento.
- **getTrackingValues:** Permette di ottenere lo stato del sistema di tracciamento date le coordinate specifiche.
- **getTrackingDuration:** Ottiene la durata di tracking necessaria in millisecondi per ogni frame.
- **getNumberOfValidCoordinateSystems:** ottiene il numero delle coordinate di tracking utilizzate.
- **get/setFreezeTracking:** ottiene/blocca lo stato del tracking.

Inoltre l'interfaccia IMetaioSDK supporta diverse tecniche di tracking messe a disposizione nella libreria. Ogni metodo di monitoraggio deve essere memorizzato in un file XML chiamato *tracking configuration file*. Questi file determinano l'impostazione iniziale del tracking e possono essere facilmente modificati attraverso l'uso di un semplice editor di testo. Generalmente ci sono due configurazioni principali di tracking: *marker-based* e *markerless*. Come può facilmente suggerire il nome il primo caso indica una tracciamento basato su dei marker facilmente riconoscibili dalla fotocamera, rendendo il tracking più veloce e sicuro del secondo. Oltre ai metodi di rilevamento ottico, l'SDK supporta anche tecniche di tracciamento non ottico, che impiega l'utilizzo di altri sensori e non solo di telecamere.

### 3.5.2 Optical Tracking

#### Marker Based

**ID Marker** Classico marker quadrato in 2 dimensioni, bianco e nero. Esistono circa 512 modelli differenti e due modalità di funzionamento, chiamate anche *tracking quality: fast* e *robust*. La prima, basata su una soglia di valori preimpostata, è adattata soprattuto per scenari con visibilità favorevole, più rapida nell'individuare il marker, ma meno precisa. La seconda viene impiegata nelle situazioni di poca luce in quanto è previsto un numero variabile di iterazioni per la ricerca del dato; impiega più tempo, ma è più affidabile.

**Parametri**

- **TrackinQuality:** Fast o Robust.
- **TresholdOffset:** soglia di numeri di pixel che serve per configurare la fotocamera nel rilevamento del marker. Se la configurazione è impostata su Fast, allora il treshold non viene cambiato. Differentemente avviene con l'impostazione robust, dove la soglia del treshold varia tra 0 e 255.
- **NumberOfSearchInteractions:** numero di tentativi di ricerca del marker, a seconda del treshold impostato.
- **Size:** dimensione del marker che si vuole riconoscere (in mm).
- **MatrixID:** numero identificativo assegnato per ogni marker.

**Picture Marker** Questi marker sono in qualche modo posti a metà tra tecnologie marker-based e markerless. Sono formati da qualsiasi immagine o stampa con abbastanza contenuto visivo, con l'unico vincolo che l'immagine di riferimento deve avere un bordo scuro e deve essere stampata davanti ad uno sfondo luminoso. Valgono le stesse considerazioni fatte per il precedente IDMarker, riguardante il tracking quality e il treshold offset.

**Parametri**

- **TrackinQuality:** Fast o Robust.
- **TresholdOffset:** soglia di numeri di pixel che serve per configurare la fotocamera nel rilevamento del marker. Se la configurazione è impostata su Fast, allora il treshold non viene cambiato. Differentemente avviene con l'impostazione robust.
- **NumberOfSearchInteractions:** numero di tentativi di ricerca del marker, a seconda del treshold impostato.
- **Size:** dimensione del marker che si vuole riconoscere (in mm).
- **ReferenceImage:** Immagine di riferimento per individuare il marcatore. L'indirizzo del file è relativo (non assoluto) alla cartella dove è contenuto il file XML con la configurazione del tracking. I formati supportati sono ppm, pgm, png e jpeg e si consiglia di mantenere una risoluzione 64x64 pixel.

**Markerless**

**ImageTracking** Prima tecnica markerless che analizziamo. Come le precedenti due anche questa si divide in due configurazioni Fast e Robust. La fast lavora fluentemente sulla maggior parte degli smartphone e con la maggior parte delle immagini. La Robust è raccomandabile per le immagini in alta definizione.

**Parametri**

- **FeatureDescriptorAlignment:** regular, upright, gravity, rectified.
- **Regular:** impostazione di default.
- **Upright:** impostazione che specifica che la fotocamera non è allineata con l'asse ottico (e.g. Ribaltata)
- **Gravity:** utilizzata per registrare oggetti fissati su una superficie verticale.

- **Rectified:** utilizzata per registrare oggetti fissati su una superficie orizzontale.
- **MaxObjectstoDetectPerFrame:** Numero massimo di rilevamenti di oggetti diversi che possono essere registrati per frame. Si cerca di prevenire il rischio di rallentamento del sistema dovuto alla ricerca di troppi oggetti virtuali nello stesso momento.
- **MaxObjectstoToTrackinParallel:** Numero massimo di rilevamenti di oggetti diversi che possono essere registrati in parallelo.
- **ReferenceImage:** Immagine di riferimento per individuare il marcatore. L'indirizzo del file è relativo (non assoluto) alla cartella dove è contenuto il file XML con la configurazione del tracking.

**3D Maps** Questa tecnica di tracking markerless permette di utilizzare qualsiasi oggetto reale come riferimento per il tracciamento. Il formato di uscita è il .3Dmap file, che può essere utilizzato sia con il tool Creator sia dallo stesso SDK.

**Parametri**

- **MinMatches:** è la soglia che indica quanto livello di accuratezza e precisione la fotocamera deve avere per identificare il tracciamento. È un valore che varia da 15 a 40.
- **NumExtensibleFeatures:** Questo parametro è molto semplice e specifica il numero di nuove caratteristiche che si vuole estendere durante l'utilizzo delle funzioni già estratte per il monitoraggio.
- **MinTriangulationAngle:** Questo parametro è utile quando viene utilizzato il rilevamento estendibile. Va a modificare la velocità di estrazione della nuova caratteristica. I valori possibili vanno da 3 (gradi) a 10 (gradi).

**LLA** LLA è l'acronimo di latitudine, longitudine e altitudine ed è un formato per la definizione delle posizioni geografiche globali. Mentre l'altitudine viene misurata in metri sopra il livello del mare, la latitudine e la longitudine sono misurate in gradi.

**Parametri**

- **TrackinQuality:** Fast o Robust.
- **TresholdOffset:** soglia di numeri di pixel che serve per configurare la fotocamera nel rilevamento del marker. Se la configurazione è impostata su Fast, allora il treshold non viene cambiato. Differentemente avviene con l'impostazione robust.
- **NumberOfSearchInteractions:** numero di tentativi di ricerca del marker, a seconda del treshold impostato.
- **Size:** dimensione del marker che si vuole riconoscere (in mm).
- **MatrixID:** numero identificativo assegnato per ogni marker.

**QR and BarCode Reader** L'SDK Metaio include inoltre un lettore QR e di uno di codice a barre.

**Parametri** Le seguenti stringhe di configurazione di tracking predefiniti possono essere utilizzate:

- **CODE**: Riconosce tutti i possibili codici a barre e codici QR.
- **QRCODE**: Riconosce solo i codici QR.

**CadBased** Il tracking markerless basato su dati CAD in 3D è una nuova funzionalità beta del SDK 5.0 per consentire una precisa localizzazione sulla base di un determinato modello 3D di un ambiente, ad esempio un piccolo oggetto che deve essere monitorato.

**Instant Tracking** È una tecnica che permette di scattare un'istantanea dalla macchina fotografica per poi essere usata come immagine di riferimento per il tracking markless nello spazio 2D o 3D (comunemente chiamata SLAM).

### 3.5.3 Not Optical Tracking

**GPS/Inertial Sensors** L'SDK Metaio fornisce anche una tecnica tracking basata su GPS e sensori inerziali sul dispositivo. Attualmente supporta sensori come accelerometri, giroscopi e sensori magnetici sia su dispositivi Android che iOS.

**Parametri**

- **LocationProvider:** se il valore del parametro è su ÒnoneÓ, verrà disattivato il GPS.
- **IgnoreCompass:** se il valore è settato su TRUE, la bussola verrà disattivata. Ciò è utile negli scenari a 360¡, lasciando il compito del tracking al giroscopio.
- **Translation/Rotation Offset:** parametri per settare la translazione e la rotazione.
- **HandEyeCalibration:** permette di calibrare l'effettiva distanza tra i 2 sensori posti sul sistema AR (e.g. 2 fotocamere)

**Dummy** il sensore dummy restituisce una posizione fissa (rotazione/traslazione) ad una velocità di aggiornamento specifico definito in millisecondi. Ad esempio può essere utilizzato per collegare facilmente una immagine da sovrapporre al componente del rendering.

**Parametri**

- **Translation/Rotation**: Parametri per settare la translazione e la rotazione del sensore.

### 3.5.4 Content Creation

L'SDK Metaio offre una vasta gamma di possibilità nella creazione di contenuti. Al fine di liberare la creatività degli sviluppatori, l'SDK articola la content creation in tre grandi aree: immagini, film e animazione 3D. E anche possibile applicare texture e altre features ad un target scelto per il tracking. Semplice come la creazione di geometrie di immagine, gli sviluppatori

possono anche esplorare diverse configurazioni per dei contenuti cinematografici attraverso poche e semplici righe di codice. La Metaio SDK offre agli sviluppatori gli strumenti per convertire i file video in modo che possano essere riprodotti senza problemi in un ambiente AR. Infine per ottenere un'esperienza totale di AR, il Metaio SDK può essere pienamente funzionale in collaborazione con la modellazione 3D e il software di animazione.

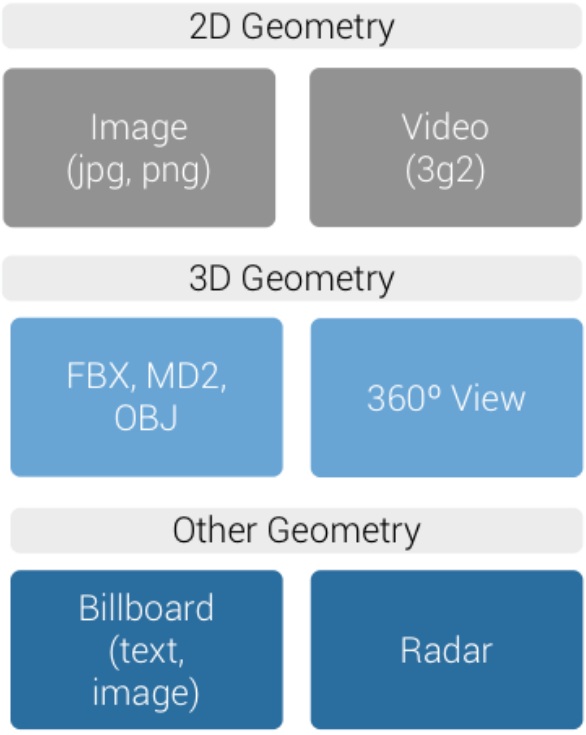


Figura 3.12: Creation types

**Image** La creazione di contenuti attraverso le immagini per le applicazioni mobile è la forma più semplice di Realtà Aumentata. Non solo creando l'immagine in sè, ma offrendo informazioni aggiuntive o modificando l'aspetto del target originale. Ad esempio, in un'applicazione basata su GPS, utilizzando sensori GPS per fornire informazioni sulla posizione geografica al tracker, lo sviluppatore può semplicemente allegare una descrizione più

dettagliata della posizione al target. Questo è molto utile in un sistema di navigazione che consente all'utente finale di identificare i luoghi che sono di grande interesse. Attualmente gli SDK Metaio supportano tre formati di immagine: Jpg, Png e Bmp. La risoluzione dell'immagine della texture è variabile in quanto l'autore può modificare la scala dell'immagine usando l' SDK stesso.

**Movie** Il contenuto Movie Texture rappresenta la tecnologia più evoluta per la grafica 2D. Con le funzionalità dell SDK è possibile specificare il percorso di un film e farlo apparire sovrapposto sopra un target 3D. Da appuntare però che il formato video è inteso anche offline, e non in streaming. Ciò significa che anche senza una connessione internet, il contenuto multimediale deve essere accessibile.

**Requisiti per la codifica video e il formato** L'SDK Metaio impone specifiche rigorose sul codec e sul formato del film, al fine di garantire una riproduzione fluida sui dispositivi mobili. Ecco le specifiche per i file video:

Compressione video: codec MPEG4.
Risoluzione video: 176x144px consigliato @ 20fps (e.g. 288kbps).
Dispositivi dual core: in grado di gestire risoluzioni più elevate.
Compressione Audio: AAC LC.
Audio Risoluzione: 22050kHz Stereo (e.g. 48kbps).

Stringa di codice per il caricamento da file di un particolare video, posto poi su un piano tridimensionale:

AREL Javascript

```
arel.Object.Model3D.createFromImage(_id, _imagePath)
```

Figura 3.13: Creation from movie

**3D Modelling** La modellazione 3D è basata sull'utilizzo di poligoni geometrici in un ambiente grafico tridimensionale, solitamente il conteggio di queste unità è calcolato in triangoli: vi possono essere modelli con un gran

quantitativo di poligoni "high-poly-model", oppure il contrario "low-poly-model". A volte queste categorie non si distinguono chiaramente, si basano sul dispositivo e sull'utilizzo del modello: in caso di applicazioni real time ad esempio, si tende ad utilizzare tecnologie a basso conteggio di poligoni; rendono più veloce e aumentano le prestazioni di una applicazione. Ogni volta che si progetta un applicazione infatti, si deve tenere conto di utilizzare il minor numero di poligoni geometrici possibili, ma allo stesso tempo anche necessari. Considerazione a parte si deve fare per le texture e le features aggiuntive a questi elementi. Bisogna essere consapevoli infatti che se si utilizzano molte texture, il sistema potrebbe risentirne in termini di prestazioni e velocità. Solitamente è consigliabile non utilizzare texture che sono più grandi di 2048 x 2048 pixel. Attualmente l'SDK Metaio supporta tre formati di oggetti 3D per la modellazione:

1. **OBJ:** Utilizzato prevalentemente per oggetti statici di alta qualità. Supporta infatti molto bene l'utilizzo di texture, anche pesanti, ma non supporta le animazioni.

2. **MD2:** Formato non più del tutto recente, ma ancora piuttosto efficacie per l'animazione 3D.

3. **FBX:** L'FBX è il formato più recente e utilizzato di modellazione, supportato anche da una vasta gamma di strumenti grafici: può infatti contenere tutto, dalla classifica modellazione grafica, d'animazione e di filmati; è stato progettato proprio per superare i precedenti due formati sia in termine di qualità d'immagine statica che d'animazione.

Qui di seguito alcune funzioni dall'interfaccia IGeometry per la modellazione delle forme geometriche:

- **setTraslation/Rotation/Scale/Transparency**: imposta la traslazione, la rotazione, la dimensione o la trasparenza della geometria.

- **getTraslation/Rotation/Scale/Transparency**: ottiene la traslazione, la rotazione, la dimensione o la trasparenza della geometria.

- **setRenderOption**: setta particolari opzioni per il rendering dell'immagine.

- **startAnimation**: avvia una particolare animazione dell'immagine.

- **setTexture**: setta una particolare texture all'immagine, data attraverso un file dato.
- **setMovieTexture**: setta una particolare texture per un contenuto movie.
- **getBoundingBox**: ottiene il bounding box dell'immagine
- **setOcclusionMode**: setta la modalità di occlusione dell'immagine.

Altre funzioni di modellazione sono definite nell'interfaccie:

**ILight**: per la gestione dei colori e della luce dinamica. Tra i quali abbiamo le funzioni (tutti i set e i get operano su parametri riguardanti il colore):

- **getAmbientColor**
- **getDiffuseColor**
- **getDirection**
- **getType**
- **getAttenuation**

**ImetaioSDK**: solita interfaccia principale del sistema, che operano su più campi d'applicazione dell'SDK.

- **getRendererType**: ottiene il tipo di render.
- **getRenderingDuration**: ottiene il tempo di rendering per una geometri 3D in millisecondi per frame.
- **createGeometry**: crea una geometria 3D dato un file specifico.
- **createLight**: crea un nuovo oggetto luce.
- **get3DLocalPositionFromScreenCoordinates**: converte le coordinate dello schermo al corrispondente oggetto 3D, sul piano dell'oggetto tracciato.

o nelle classi **MovieTextureStatus**,**ImageStruct** (per la creazione di una immagine tramite struttura apposita), **correspondence2D3D**(per la mappatura di un immagine 3D in un ambiente 2D e viceversa) e infine le classi di creazione base di vettori: **vector2d**,**vector3d**,**vector4d** ognuna delle quali va ad agire sulle coordinate del vettore della dimensione corrispondente.

### 3.5.5 Capturing and Sensor Handler

L'ultimo campo di applicazione di cui l'SDK Metaio fornisce delle librerie per una possibile gestione, è la parte riguardante il capturing e la gestione dei sensori. Come sempre 'interfaccia *ImetaioSDK* fornisce alcuni metodi sulla gestione della foto/videocamera per l'acquisizione dell'immagine visiva, oltre alla vera e propria classe di riferimento *metaio::Camera* che si occupa di definire le possibili fotocamere disponibili nel sistema, attraverso il costruttore principale Camera, che va a modificare attributi pubblici come *direction*, *resolution* and *fps*.

**ImetaioSDK**

- **set/getCameraParameters**: ottiene (o setta dato un file XML) i parametri per la fotocamera.
- **startCamera**: avvia il processo di acquisizione dell'immagine.
- **stopCamera**: arresta il processo di acquisizione dell'immagine.
- **requestScreenshot**: richiede un salvataggio dello screen da memorizzare in un file.
- **requestCameraImage**: chiama una callback che andrà a salvare in un file l'immagine di alta risoluzione scattata dalla fotocamera.

Per quanto riguarda invece la parte di gestione dei sensori abbiamo a disposizione l'interfaccia *ISensorComponent*, che gestisce oltre alla creazione dell'oggetto sensore, le informazioni riguardante i sensori di posizione, gravità e bussola. La classe *SensorComponentDummy* si occupa invece di gestire la parte riguardante i sensori dummy precedentemente descritti; e infine la solita interfaccia *IMetaioSDK* ci mette a disposizione funzioni base come:

- **registerSensorComponent**: registra (o de-registra) sensori ne sistema.
- **pause/resumeSensor**: avvia o mette in pausa il sensore.
- **sensorCommand**: invia un comando al sensore specifico dipendentemente dal comando di input dell'utente.

# Capitolo 4

# Conclusioni

Abbiamo visto le principali caratteristiche di un sistema AR, i principali campi di applicazioni, in cosa e come è presente uno standard per lo sviluppo di essi. Abbiamo affrontato il tema dello sviluppo mobile, prendendo in esame vari SDK disponibili sul mercato approfondendo il caso di studio di Metaio SDK. Da qui siamo passati a osservare e analizzare il codice, soffermandoci anche sulle interazioni e i metodi che accomunano ogni applicazione. Metaio SDK offre grandi potenzialità per quanto riguarda la programmazione AR, avendo a disposizione un offerta completa rispetto alle altre piattaforme. Non solo infatti, offre una rapida produzione di contenuto AR, sfruttando il sistema content-based dell'azienda tedesca, ma permette di sfruttare al meglio anche l'esperienza mobile (Junaio browser) senza tralasciare tecnologie di tracking basate su GPS, Face Recognition e Visual Search. Attualmente difatti, Metaio rappresenta la migliore realtà che abbiamo a disposizione. La ricerca condotta fino ad ora sulla tecnologia AR, ha portato la consapevolezza che una visione del rapporto reale-virtuale, è possibile. L'idea classica che vede l'inizio del mondo virtuale nel momento esatto in cui finisce il reale non è più logicamente concepibile, in quanto, seguendo il paradigma di *ubiquitos computing*, il rapporto uomo-macchina non può più essere confinato ad un livello limitato come quello *desktop*. I progettisti AR si occuperanno di sviluppare sistemi user friendly che sapranno integrarsi così, in una dimensione reale, della vita di tutti i giorni. Informazioni facilmente accessibili e sempre disponibili, strettamente correlate a posizione fisiche e oggetti reali, porteranno a una totale integrazione tra questi due mondi.

# Bibliografia


[1] B. MacIntyre and S.Feiner, *Language-level support for exploratory programming of distributed virtual environments*, Seattle, WA, USA.

[2] W.Pierarski and B.Thomas, *An Object Oriented Software Architecture for 3D Mixed Reality Applications.*

[3] T.Reicher and A.Mac Williams *Study on Software Architectures for Augmented Reality Systems, report for the ARVIKA consortium*,Technische Universitat Munchen, 2002

[4] T.Reicher and A.Mac Williams and B.Bruegge *Towards a System of Patterns for Augmented Reality Systems*,in International Workshop on Software Technology for Augmented Reality Systems

[5] D.C.Schmidt and M.Stal and H.Rohnert and Buschmann *Pattern-Oriented Software Architecture, Vol. 2: Patterns for Concurrent and Networked Objects*, Wiley, New York, NY, 2000.

[6] A.MacWilliams, T.Reicher, G.Klinker, B.Bruegge *Design Patterns for Augmented Reality Systems*, Institut fur Informatik, Technische Universitat Munchen.

[7] Martin Lechner, *A Proposed Step-By-Step Guide to an AR Standard*, Wikitude GmbH.

[8] Daniel Wagner, *Handheld Augmented Reality*, Graz University of Technology Institute for Computer Graphics and Vision

[9] Ronald Azuma and Gary Bishop, *Improving Static and Dynamic Registration in an Optical See-through HMD*, Department of Computer Science University of North Carolina at Chapel Hill

[10] Hirokazu Kato and Mark Billinghurst, *Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System*, Faculty of Information Sciences, Hiroshima City University, Human Interface Technology Laboratory, University of Washington.

[11] Ronald T. Azuma, *A Survey of Augmented Reality*, Hughes Research Laboratories 3011 Malibu Canyon Road, MS RL96.

[12] Caudell and Mizell, *Augmented reality: An application of heads-up display technology to manual manufacturing processes*, Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences.

[13] Paul Milgram and Fumio Kishino, *A taxonomy of mixed reality visual displays.*, IEICE TRANSACTIONS on Information and Systems.

[14] Mirko Falavigna and Paolo Piazzi, *Realtà aumentata per applicazioni industriali.* Technology Transfer Team

[15] A.Dunser, R.Grasset, and H.Farrant., *Towards immersive and adaptive augmented reality exposure treatment. 2011.*

[16] I. Sutherland., *A head-mounted three dimensional display* In Proceedings of Fall Joint Computer Conference, 1968.

[17] M. Livingston , *Military applications of augmented reality. 2011.*

[18] Vuforia, *https://www.vuforia.com*

[19] D'Fusion, *http://www.t-immersion.com/*

[20] Wikitude, *http://www.wikitude.com*

[21] String, *http://www.poweredbystring.com*

[22] Metaio, *http://www.metaio.com*

[23] Metaio, *http://dev.metaio.com*