

Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Google ARCore

Marco Vettore
Matricola 1216433

Alberto Varini
Matricola 1227153

Mattia Tamiazzo
Matricola 1219603

Indice

1	Introduzione	1
2	Realtà aumentata	2
2.1	Storia	2
2.2	Applicazioni	3
3	Funzioni fondamentali	4
3.1	Light estimation	4
4	Augmented faces	7
4.1	Configurazione e utilizzo	8
5	Cloud anchors	9
5.1	Funzionamento	9
5.2	Configurazione e utilizzo	10
6	Geospatial API	12
6.1	Configurazione e utilizzo	12
	Bibliografia	15

1. Introduzione

ARCore è un kit di sviluppo lanciato da Google nel mese di marzo 2018, utilizzabile nella maggior parte degli smartphone con Android Nougat o superiore (API level 24+). Tramite esso è possibile sviluppare applicazioni con funzionalità in realtà aumentata, permettendo all'utente di interagire con l'ambiente che lo circonda. In questo documento, dopo una breve panoramica sulla realtà aumentata, verranno analizzate le principali funzionalità e caratteristiche dell'SDK.

Nei primi capitoli vengono presentate le tre funzioni fondamentali del framework ARCore, che permettono al dispositivo di integrare contenuti virtuali al mondo reale:

- Il rilevamento del movimento, che consente ad esso di tracciare la propria posizione nel mondo.
- La comprensione ambientale, che permette la rilevazione della posizione e della dimensione delle superfici.
- La stima della luce, che consente di valutare le condizioni di illuminazione dell'ambiente.

Nei capitoli successivi sono invece presentate le funzionalità aggiuntive del framework, che consentono di migliorare l'integrazione tra virtuale e reale, come ad esempio il rilevamento della profondità, il posizionamento istantaneo, le API *Augmented Images* e *Augmented Faces* e altre importanti funzioni aggiuntive.

2. Realtà aumentata

La Realtà Aumentata (Augmented Reality - *AR*) è una tecnologia che permette di compiere esperienze interattive, in cui l'ambiente reale viene arricchito da contenuti virtuali. Similmente alla realtà virtuale, vengono creati elementi grafici sintetici con cui l'utente può interagire attraverso i sensi. Tuttavia, come spiegato in [2], nell'*AR* l'ambiente reale gioca un ruolo fondamentale: lo scopo della realtà aumentata è proprio cercare di collegare il mondo reale con quello virtuale.

L'*AR* viene definita in [1] come un sistema che incorpora tre caratteristiche principali: la combinazione tra reale e virtuale, l'interazione real-time e la rappresentazione 3D.

2.1 Storia

Il primo rudimentale sistema di realtà aumentata è stato creato da Ivan Sutherland [19] nel 1968. Esso era composto da un display ottico trasparente che veniva montato sulla testa e che poteva mostrare semplici immagini in tempo reale. Nel 1993 George Fitzmaurice ha creato *Chameleon* [11], un dispositivo che tramite un piccolo schermo collegato a una videocamera poteva essere orientato per esplorare uno spazio virtuale 3D. Simile al prototipo di Fitzmaurice, nel 1995 Jun Rekimoto e Katashi Nagao creano *NaviCam* [16], che prendendo in input un flusso video poteva riconoscere in real-time dei marcatori colorati e sovrapporre al video delle informazioni testuali.

Dal 2000 vengono creati altri primi sistemi di realtà aumentata, con applicazioni soprattutto a giochi interattivi, come ad esempio l'estensione *ARQuake* [20] o *Human Pacman* [4], ancora vincolati alle scarse prestazioni dei dispositivi mobili. Solo con l'aggiunta ai cellulari della fotocamera, e poi di schermi touch, vengono quindi create le prime applicazioni commerciali in grado di sfruttare le potenzialità della realtà aumentata. Ne sono esempi *AR Tennis* [12], primo gioco in AR collaborativo per cellulare, e *ARhrrrr!*, primo gioco mobile in realtà aumentata con contenuti grafici di alta qualità. Vengono quindi sviluppate le prime librerie software per la realtà aumentata, come *ARToolKit*, implementata prima in linguaggio C e poi in C++ nelle versioni più recenti, *OpenCV*, che possiede anche funzionalità per l'*AR*, e dal 2018 la libreria per Android *ARCore* di Google.

2.2 Applicazioni

Le applicazioni della realtà aumentata possono riguardare diversi ambiti, ai quali questa tecnologia può apportare benefici economici o qualitativi, oppure creare servizi innovativi [3].

In particolare, nel corso degli anni la tecnologia AR è stata usata per scopi pubblicitari e commerciali, quali la prova di capi d'abbigliamento senza doverli indossare o l'integrazione al marketing cartaceo di video promozionali tramite riconoscimento delle immagini, o per l'intrattenimento, come nello sviluppo di videogiochi. Trova inoltre applicazioni nella produzione industriale, in cui vengono sovrapposte all'area di lavoro istruzioni virtuali, nell'ambito militare, come l'addestramento al volo dei piloti, o per la formazione e la pratica sanitaria.

3. Funzioni fondamentali

TODO creazione di sessione? Nel seguente capitolo vengono

3.1 Light estimation

Nel rendering in realtà aumentata è importante che gli oggetti virtuali siano il più possibile integrati con l'ambiente circostante. Una delle caratteristiche principali che permette all'occhio umano di percepire la posizione di un oggetto nello spazio è la luce, cioè il modo in cui esso viene illuminato e l'ombra che proietta. Proprio per questo motivo, il framework ARCore mette a disposizione il *Light estimation API*, che fornisce informazioni dettagliate riguardo l'illuminazione della scena, come spiegato nella documentazione ufficiale [8]. Tali informazioni sono necessarie per imitare i vari effetti che producono gli oggetti reali quando colpiti da una fonte di luce, che sono descritti dalla figura 3.1 nella pagina seguente:

- le ombre (*shadows*), che sono direzionali e suggeriscono dove è collocata la fonte di luce;
- l'ombreggiatura (*shading*), cioè l'intensità della luce che colpisce una certa faccia dell'oggetto;
- la lumeggiatura (*specular highlight*), la macchia luminosa che compare su un oggetto lucido quando viene illuminato;
- la riflessione (*reflection*), che può essere con proprietà speculari per oggetti completamente lucidi, come ad esempio uno specchio, oppure di diffusione, non dando un chiaro riflesso dell'ambiente circostante.

Le modalità per la gestione della stima della luce sono due, l'*Environmental HDR mode* e l'*Ambient intensity mode*. Durante la configurazione della sessione ARCore può essere scelta una delle due modalità, oppure disabilitare la stima della luce, come mostra il listing 3.1 nella pagina successiva tratto dalla guida ufficiale.



Figura 3.1: Esempio degli effetti prodotti dagli oggetti quando sono illuminati.

```

1 // Configura la sessione in modalità ENVIRONMENTAL_HDR
2 val config : Config = session.config
3 config.lightEstimationMode = LightEstimationMode.ENVIRONMENTAL_HDR
4 session.configure(config)
5
6 // Configura la sessione in modalità AMBIENT_INTENSITY
7 val config : Config = session.config
8 config.lightEstimationMode = LightEstimationMode.AMBIENT_INTENSITY
9 session.configure(config)
10
11 // Configura la sessione disabilitando la Light Estimation API
12 val config : Config = session.config
13 config.lightEstimationMode = LightEstimationMode.DISABLED
14 session.configure(config)

```

Listing 3.1: Configurazione della modalità di stima della luce.

3.1.1 Environmental HDR mode

La modalità *Environmental HDR* combina tre diverse API per replicare la luce reale, come descritti dalla figura 3.2 nella pagina seguente.

Main Directional Light Questa API calcola la direzione e l'intensità della fonte di luce principale, permettendo di posizionare correttamente l'ombra e la lumeggiatura dell'oggetto virtuale. Inoltre, questa funzionalità permette ad entrambi questi effetti ottici di venire corretti se cambia la posizione relativa dell'oggetto rispetto la fonte di luce

Ambient Spherical Harmonics Questa funzionalità permette di rappresentare la luce ambientale della scena, parametrizzando l'intensità della luce proveniente

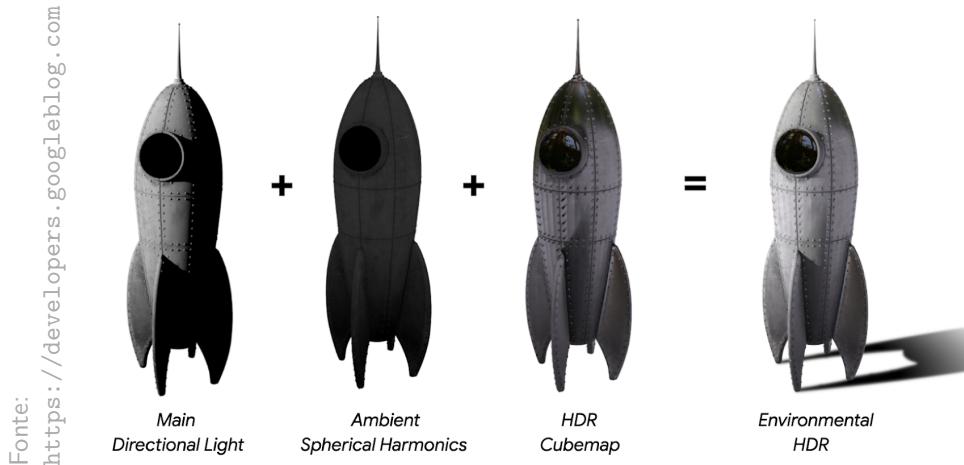


Figura 3.2: Composizione della modalità Environmental HDR.

dalle varie direzioni.

HDR Cubemap Essa permette di riprodurre la riflessione di oggetti con superfici lucide. Tramite questa API viene modificata anche l'ombreggiatura e il colore dell'oggetto, che dipenderanno dalla tonalità dell'ambiente circostante.

3.1.2 Ambient intensity mode

La modalità *Ambient intensity* determina l'intensità media dei pixel e la correzione del colore di una data immagine. Dopo aver filtrato l'intensità media di un insieme di pixel e il bilanciamento del bianco per ogni frame, vengono corretti la luce e il colore dell'oggetto virtuale, affinché si integri meglio con la scena [18]. Questa modalità può essere utilizzata se la stima della luce non è critica, come per oggetti che possiedono già una propria illuminazione integrata.

4. Augmented faces

L'API *Augmented Faces* permette di identificare i volti umani e le varie parti che lo compongono tramite Intelligenza Artificiale, per sovrapporre ad essi modelli 3D come maschere, occhiali, cappelli utilizzando solo la fotocamera frontale [5]. Questa libreria permette ottenere un *face mesh*, una rappresentazione virtuale composta da una maglia di punti che riproduce il profilo del volto [14]. Oltre ad essa, l'API fornisce un *center pose* e tre *region pose*, come descritti dalla figura 4.1 tratte dalla documentazione ufficiale.

Face mesh Consiste in una rete di 468 punti, che permette di posizionare una texture sul volto. Essa viene tracciata come un piano, per permettere all'immagine virtuale di seguire il volto anche se in movimento, come spiegato in [9].

Center pose Rappresenta il centro del volto, posizionato dietro il naso. Utile per il rendering di oggetti virtuali da posizionare sopra la testa.

Region pose Identifica una regione rilevante del volto, come i lati destro o sinistro della fronte, oppure il naso. Sono utili per il rendering di oggetti virtuali da posizionare sul naso o attorno agli orecchi.

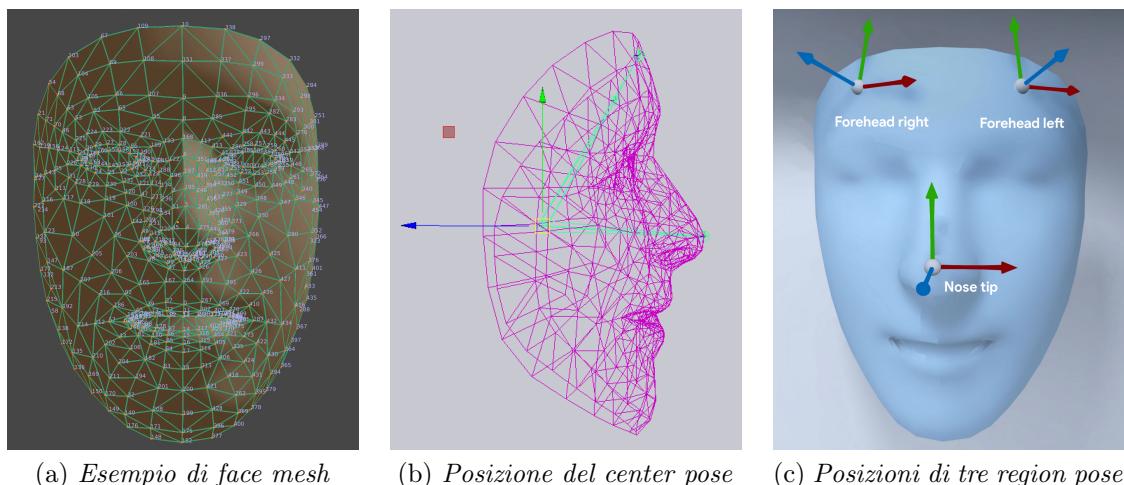


Figura 4.1: Elementi ottenuti tramite l'API Augmented Faces

4.1 Configurazione e utilizzo

La configurazione della sessione ARCore deve essere effettuata selezionando la fotocamera frontale ed abilitando la modalità Augmented Face, come mostrato nel listing 4.1 tratto dalla guida ufficiale Google.

```

1 // Configura la sessione utilizzando la camera frontale.
2 val filter = CameraConfigFilter(session).setFacingDirection(CameraConfig.
3   FacingDirection.FRONT)
4 val cameraConfig = session.getSupportedCameraConfigs(filter)[0]
5 session.cameraConfig = cameraConfig
6
7 // Abilita la modalità Augmented Face.
8 val config = Config(session)
9 config.augmentedFaceMode = Config.AugmentedFaceMode.MESH3D
10 session.configure(config)

```

Listing 4.1: Configurazione della modalità Augmented Face.

Da ogni frame è possibile ricavare un oggetto **Trackable**, che può essere tracciato e a cui possono essere collegate degli **Anchor**. Verificando lo stato di ogni oggetto **Trackable** restituito, è possibile ricavare i region pose, il center pose e i vertici del face mesh, per poi procedere con il rendering degli oggetti virtuali. Si veda il listing 4.2 tratto dalla documentazione ufficiale per un possibile utilizzo.

```

1 // Ricava gli oggetti trackable dalla sessione ARCore
2 val faces = session.getAllTrackables(AugmentedFace::class.java)
3
4 // Verifica lo stato di ogni oggetto contenuto nella lista di Trackable
5 faces.forEach { face ->
6   if (face.trackingState == TrackingState.TRACKING) {
7     // Ricava il center pose
8     val facePose = face.centerPose
9
10    // Ricava i region pose
11    val foreheadLeft = face.regionPose(AugmentedFace.RegionType.FOREHEAD_LEFT)
12    val foreheadRight = face.regionPose(AugmentedFace.RegionType.FOREHEAD_RIGHT)
13    val noseTip = face.regionPose(AugmentedFace.RegionType.NOSE_TIP)
14
15    // Ricava i vertici del face mesh
16    val faceVertices = face.meshVertices
17
18    // Rendering dell'oggetto virtuale
19  }
20}

```

Listing 4.2: Utilizzo della modalità Augmented Faces.

5. Cloud anchors

L'API ARCore *Cloud Anchor* introduce i cloud anchor, un tipo speciale di anchor che permettono di condividere con altri utenti l'esperienza AR. In particolare, un dispositivo può posizionare oggetti virtuali nello spazio, e altri utenti possono vedere l'oggetto virtuale ed interagire con esso trovandosi nella stessa posizione, come spiegato da [13].

Questo tipo di anchor, come spiegato dalla documentazione ufficiale [6], trova applicazione ad esempio per creare oggetti virtuali che persistano nel mondo reale, cioè che mantengano nel tempo la posizione in cui sono stati creati, oppure per creare giochi virtuali multigiocatore in cui è importante la collaborazione tra utenti in tempo reale.

5.1 Funzionamento

La creazione e la diffusione dell'anchor avviene tramite connessione internet in quattro passaggi, descritti ad alto livello come segue. La figura 5.1 nella pagina seguente tratta dalla guida ufficiale Google descrive visivamente le quattro fasi del funzionamento dei cloud anchor.

1. **Creation:** l'utente crea un anchor localmente;
2. **Hosting:** ARCore carica i dati della mappa 3D dello spazio circostante all'anchor locale nell'ARCore Cloud Anchor, che a sua volta restituisce al dispositivo un Cloud Anchor ID univoco;
3. **Distribution:** l'app distribuisce l'ID univoco agli altri utenti;
4. **Resolving:** gli utenti possono utilizzare l'ID ricevuto per ricreare l'anchor quando si trovano nello stesso ambiente. L'API compara la scena del dispositivo con la mappa 3D caricata precedentemente per determinare la posizione dell'utente e visualizzare correttamente l'oggetto virtuale.

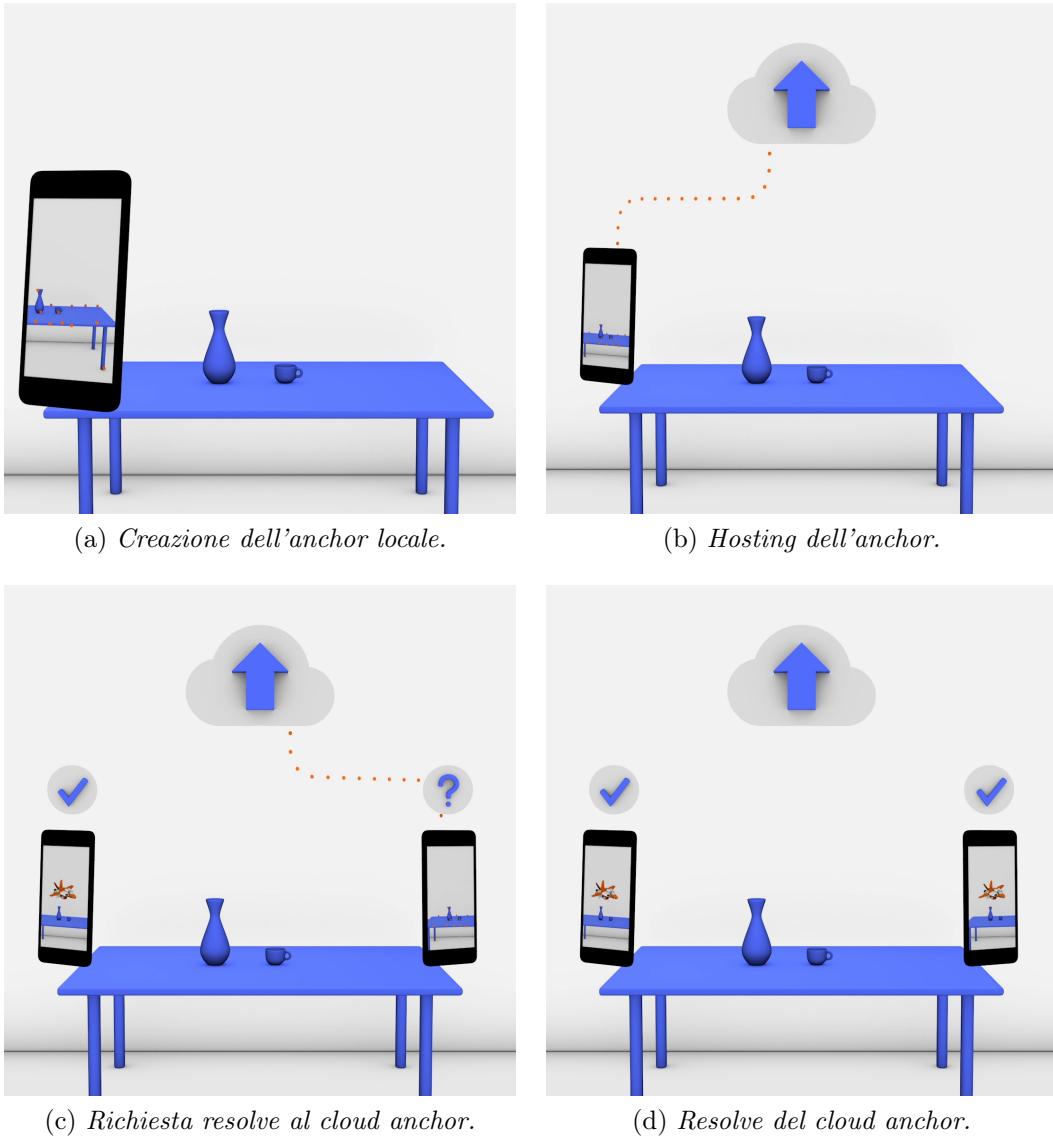


Figura 5.1: Funzionamento ad alto livello di Cloud Anchor API.

5.2 Configurazione e utilizzo

Per implementare un'applicazione che utilizzi i Cloud Anchor è prima necessario creare un progetto in *Google Cloud Platform* e abilitare il ARCore Cloud Anchor API per l'hosting, il salvataggio e il resolving degli anchor.

La sessione ARCore deve essere impostata per poter utilizzare l'API Cloud Anchors, come descritto dal listing 5.1 tratto dalla documentazione ufficiale.

```

1 val config = Config(session)
2 config.cloudAnchorMode = Config.CloudAnchorMode.ENABLED
3 session.configure(config)

```

Listing 5.1: Configurazione della modalità Cloud Anchor.

5.2.1 Autenticazione

Se il cloud anchor deve poter avere processi host/resolve di durata massima 24 ore, come ad esempio per giochi virtuali multigiocatore, è necessaria un'autenticazione tramite chiave dell'app. Tale autenticazione si compie generando una *API key* per il progetto cloud dal *Google Cloud Console* e aggiungendola nel campo `android:value` in un tag `meta-data`, nel campo `application` del file `AndroidManifest.xml` dell'applicazione, come spiegato dal listing 5.2.

```

1 <meta-data
2   android:name = "com.google.android.ar.API_KEY"
3   android:value = "API_KEY"/>

```

Listing 5.2: Autenticazione con API key.

Per cloud anchor che devono persistere per una durata compresa tra 1 e 365 giorni invece, è necessaria un'autenticazione tramite *OAuth client*, che associa l'applicazione android al progetto di Google Cloud Platform. Tale autenticazione richiede una chiave *SHA-1 fingerprint* che si può generare con il task `signingReport` di Gradle.

5.2.2 Hosting

Il metodo `hostCloudAnchorWithTtl(anchor:Anchor, ttlDays:Int)` della classe `Session` permette di iniziare l'hosting di `anchor` con una durata di `ttlDays` giorni, un numero positivo compreso tra 1 e 365; per hosting con durata fino a 24 ore è invece necessaria l'invocazione del metodo `hostCloudAnchor(anchor:Anchor)`. Entrambi i metodi restituiscono il nuovo anchor, con la stessa posizione di `anchor` e con stato `Anchor.CloudAnchorState.TASK_IN_PROGRESS`. Il listing 5.3 tratto dal codelab [17] realizza l'hosting tramite un oggetto della classe wrapper `CloudAnchorManager`.

```

1 val cloudAnchorManager : CloudAnchorManager = CloudAnchorManager()
2
3 // Realizza l'hosting per l'anchor currentAnchor con durata 300 giorni
4 cloudAnchorManager.hostCloudAnchor(session, currentAnchor, 300, this ::
    onHostedAnchorAvailable);

```

Listing 5.3: Hosting di un cloud anchor.

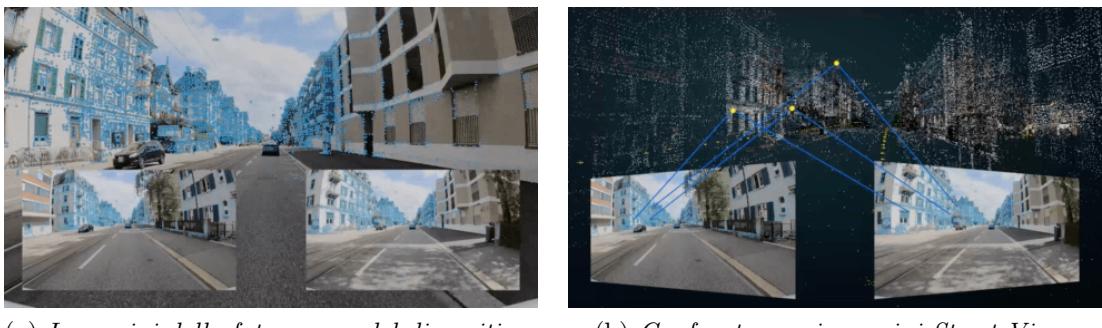
5.2.3 Resolving

Il metodo `resolveCloudAnchor(String cloudAnchorId)` della classe `Session` compie il resolving, creando un anchor sulla base delle informazioni 3D inviate e compatte con la mappa 3D caricata durante l'hosting, per poter posizionare correttamente l'oggetto rispetto al dispositivo che sta compiendo il resolving.

6. Geospatial API

L'API *Geospatial* è una funzionalità aggiunta a maggio 2022 al framework ARCore, che utilizza i dati di *Google Earth 3D* e *Google Maps Street View* per creare contenuti AR basati sulla posizione geografica. L'API sfrutta il *global localization*, il quale combina il *VPS - Visual Positioning Service*, un servizio Google che analizza l'ambiente circostante attraverso la fotocamera per determinare la posizione, *Street View*, che fornisce un database di immagini di luoghi, e il machine learning per migliorare la determinazione della posizione del dispositivo [15].

Il Geospatial API compara le informazioni provenienti dalla fotocamera (figura 6.1a) e dai sensori del dispositivo, come il GPS, con miliardi di immagini 3D estratte tramite machine learning da Street View (figura 6.1b) per determinare la posizione e l'orientamento del dispositivo, per poi mostrare contenuti AR posizionati correttamente rispetto all'utente, come spiegato in [10].



(a) Immagini della fotocamera del dispositivo (b) Confronto con immagini Street View

Figura 6.1: Ricostruzione delle funzionalità di Geospatial API.

6.1 Configurazione e utilizzo

La sessione ARCore deve abilitare l'utilizzo di Geospatial API, come descritto dal listing 6.1 tratto dalla documentazione ufficiale [7].

```
1 // Abilita il Geospatial API.  
2 session.configure(session.config.apply{ geospatialMode= Config.GeospatialMode.  
    ENABLED })
```

Listing 6.1: Configurazione della modalità Geospatial API.

6.1.1 Configurazione di VPS

L'utilizzo di Visual Positioning System impone che l'app sia associata a un progetto Google Cloud Project con abilitato il ARCore API. È inoltre necessaria un'autenticazione tramite *OAuth client*, oppure con *API key*. Si veda il paragrafo 5.2.1 a pagina 11 del capitolo Cloud Anchor per specifiche su entrambi i tipi di autenticazione.

Sono necessari inoltre i permessi per accedere alla posizione e ad internet per comunicare con il servizio online Geospatial API, da dichiarare nel campo `manifest` del file `AndroidManifest.xml`, come descritto dal listing 6.2.

```

1 <manifest ...>
2   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
3   <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
4   <uses-permission android:name="android.permission.INTERNET"/>
5 </manifest>
```

Listing 6.2: Richiesta di permessi per l'uso di Geospatial API.

6.1.2 Calcolo della posizione

La posizione può essere presa da un oggetto della classe `Earth`, ricevuto dalla sessione ARCore `session`, come descritto dal listing 6.3. Se l'oggetto di tipo `Earth` ha stato `TrackingState.TRACKING`, la posizione può essere ottenuta tramite un oggetto di tipo `GeospatialPose`, che contiene la latitudine e la longitudine, l'altitudine e un'approssimazione della direzione verso cui il dispositivo è rivolto.

```

1 val earth = session.earth
2 // Verifico che sia in stato TRACKING
3 if (earth?.trackingState == TrackingState.TRACKING) {
4   val cameraGeospatialPose : GeospatialPose = earth.cameraGeospatialPose
5
6   // Salvataggio della latitudine in gradi
7   val latitude : Double = cameraGeospatialPose.latitude
8   // Salvataggio della longitudine in gradi
9   val longitude : Double = cameraGeospatialPose.longitude
10  // Salvataggio dell'altitudine in metri
11  val elevation : Double = cameraGeospatialPose.altitude
12  // Salvataggio dell'orientamento in gradi
13  val heading : Double = cameraGeospatialPose.heading
14 }
```

Listing 6.3: Calcolo della posizione corrente.

L'oggetto `GeospatialPose` specifica anche l'accuratezza *A* dei dati ricevuti, tramite i metodi `getHeadingAccuracy()`, `getHorizontalAccuracy()` e `getVerticalAccuracy()`. I valori restituiti dai metodi hanno stessa unità di misura dei valori stimati *E* di cui specificano la precisione, cioè gradi per la latitudine, la longitudine e l'orientamento.

e metri per l'altitudine, e specificano che la posizione reale R è compresa con una probabilità del 68% nell'intervallo:

$$R \in [E - A, E + A].$$

Ad esempio, se il metodo `GeospatialPose.getHeading()` restituisce il valore $E = 60^\circ$ e il metodo `GeospatialPose.getHeadingAccuracy()` ritorna una precisione di $A = 10^\circ$, il valore reale sarà con probabilità del 68% nell'intervallo $R \in [50^\circ, 70^\circ]$. Un valore alto di accuratezza quindi garantisce una precisione minore.

6.1.3 Posizionamento di un anchor Geospatial

Per il posizionamento di un anchor, i valori di latitudine e longitudine devono essere dati rispettando le specifiche WGS84, mentre l'altitudine è definita come la distanza in metri dall'elissoide definito dallo stesso standard. L'orientamento dell'anchor invece viene fatto con l'utilizzo di un quaternione (qx, qy, qz, qw). Si veda il listing 6.4 per un esempio di creazione dell'anchor.

```

1 if (earth.trackingState == TrackingState.TRACKING) {
2     val anchor = earth.createAnchor (
3         /* Valori della posizione */
4         latitude, longitude, altitude,
5         /* Valori della rotazione */
6         qx, qy, qz, qw)
7         // ...
8 }
```

Listing 6.4: Posizionamento di anchor Geospatial.

L'altitudine dell'anchor, se esso viene posizionato vicino all'utente, può avere lo stesso valore dell'altitudine restituita dal metodo `GeospatialPose.getAltitude()`. Se invece l'anchor deve avere un'altitudine diversa da quella dell'utente, essa può essere ricavata dall'API Google Maps, forzando la prospettiva 2D e convertendo il valore restituito, basato sullo standard EGM96, nella codifica WGS84.

Bibliografia

- [1] Ronald T. Azuma. «A Survey of Augmented Reality». In: *Presence: Teleoperators and Virtual Environments* 6.4 (ago. 1997), pp. 355–385. DOI: 10.1162/pres.1997.6.4.355.
- [2] Bimber et al. *Spatial Augmented Reality Merging Real and Virtual Worlds*. Ago. 2005. ISBN: 9780429108501. DOI: 10.1201/b10624.
- [3] Julie Carmigniani et al. «Augmented Reality Technologies, Systems and Applications». In: *Multimedia Tools Appl.* 51.1 (gen. 2011), pp. 341–377. ISSN: 1380-7501. DOI: 10.1007/s11042-010-0660-6.
- [4] Adrian David Cheok et al. «Human Pacman: A Sensing-Based Mobile Entertainment System with Ubiquitous Computing and Tangible Interaction». In: *Proceedings of the 2nd Workshop on Network and System Support for Games*. NetGames '03. Redwood City, California: Association for Computing Machinery, 2003, pp. 106–117. ISBN: 1581137346. DOI: 10.1145/963900.963911.
- [5] Google developers. *Augmented Faces introduction*. Feb. 2022. URL: <https://developers.google.com/ar/develop/augmented-faces>.
- [6] Google developers. *Cloud Anchors allow different users to share AR experiences*. Apr. 2022. URL: <https://developers.google.com/ar/develop/cloud-anchors>.
- [7] Google developers. *Geospatial developer guide for Android (Kotlin/Java)*. Giu. 2022. URL: <https://developers.google.com/ar/develop/java/geospatial/developer-guide>.
- [8] Google developers. *Get the lighting right*. Mag. 2022. URL: <https://developers.google.com/ar/develop/lighting-estimation>.
- [9] Google developers. *New UI tools and a richer creative canvas come to ARCore*. Feb. 2019. URL: <https://developers.googleblog.com/2019/02/new-ui-tools-and-richer-creative-canvas.html>.
- [10] Google developers. *Using Global Localization to Improve Navigation*. Mag. 2022. URL: <https://developers.googleblog.com/2022/05/Make-the-world-your-canvas-ARCore-Geospatial-API.html>.

- [11] George W. Fitzmaurice. «Situated Information Spaces and Spatially Aware Palmtop Computers». In: *Commun. ACM* 36.7 (1993), pp. 39–49. ISSN: 0001-0782. DOI: 10.1145/159544.159566.
- [12] Anders Henrysson, Mark Billinghurst e Mark Ollila. «AR Tennis». In: SIGGRAPH '06 (2006), 1–es. DOI: 10.1145/1179133.1179135.
- [13] Ida Bagus Kerthyayana Manuaba. «Mobile based Augmented Reality Application Prototype for Remote Collaboration Scenario Using ARCore Cloud Anchor». In: *Procedia Computer Science* 179 (2021). 5th International Conference on Computer Science and Computational Intelligence 2020, pp. 289–296. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2021.01.008>.
- [14] Zainab Oufqir, Abdellatif El Abderrahmani e Khalid Satori. «ARKit and ARCore in serve to augmented reality». In: *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*. 2020, pp. 1–7. DOI: 10.1109/ISCV49265.2020.9204243.
- [15] Tilman Reinhardt. *Using Global Localization to Improve Navigation*. Feb. 2019. URL: <https://ai.googleblog.com/2019/02/using-global-localization-to-improve.html>.
- [16] Jun Rekimoto e Katashi Nagao. «The World through the Computer: Computer Augmented Interaction with Real World Environments». In: *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*. UIST '95. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1995, pp. 29–36. ISBN: 089791709X. DOI: 10.1145/215585.215639.
- [17] Fred Sauer e Dereck Bridie. *ARCore Cloud Anchors with persistent Cloud Anchors*. Ott. 2021. URL: <https://codelabs.developers.google.com/codelabs/arcore-cloud-anchors>.
- [18] Aleksi Suonsivu. «RGBD SLAM Based 3D Object Reconstruction and Tracking: Using Google ARCore». B.S. thesis. 2020.
- [19] Ivan E. Sutherland. «A Head-Mounted Three Dimensional Display». In: *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*. AFIPS '68 (Fall, part I). San Francisco, California: Association for Computing Machinery, 1968, pp. 757–764. ISBN: 9781450378994. DOI: 10.1145/1476589.1476686.
- [20] B. Thomas et al. «ARQuake: an outdoor/indoor augmented reality first person application». In: *Digest of Papers. Fourth International Symposium on Wearable Computers*. 2000, pp. 139–146. DOI: 10.1109/ISWC.2000.888480.