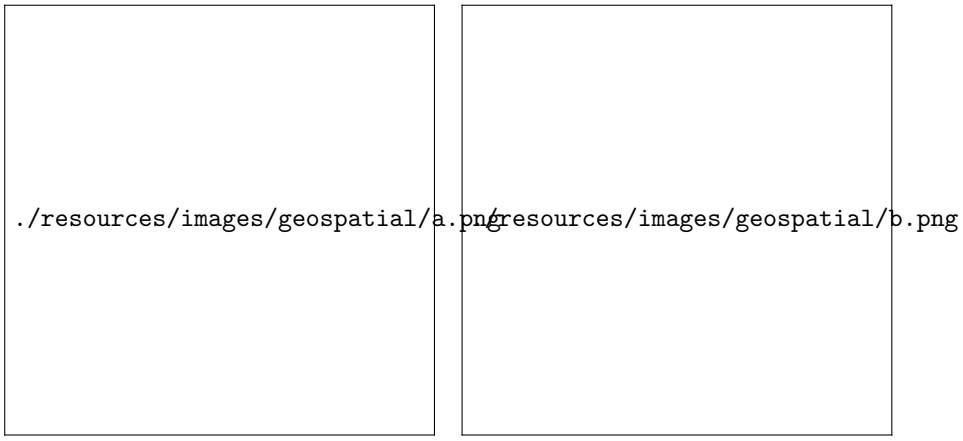
Chapter 1

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



(a) *Immagini della fotocamera del dispositivo*

(b) *Confronto con immagini Street View*

Figure 1.1: Ricostruzione delle funzionalità di Geospatial API.

1.1 Configurazione e utilizzo

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

[caption=Configurazione della modalità Geospatial API., label=lst:geo$_s$$ession, language = Kotlin]//AbilitailGeospatialAPI.session.configure(session.config.applygeospatialMode = Config

1.1.1 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 **??** 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 **??**.

[caption=Richiesta di permessi per l'uso di Geospatial API., label=lst:geo$_p$$erm, language = xml, morekeywords = android : name, android : value, keywordstyle = , alsodigit = −, stringstyle = , emph = manifest, uses − permission, emphstyle =] < manifest... >< uses − permissionandroid : name = "android.permission.ACCESS$_F$INE$_L$OCATION"/ >< uses − permissionandroid : name = "android.permission.ACCESS$_C$OARSE$_L$OCATION"/ >< uses − permissionandroid : name = "android.permission.INTERNET"/ >< /manifest >

1.1.2 Calcolo della posizione

La posizione può essere presa da un oggetto della classe `Earth`, ricevuto dalla sessione ARCore `session`, come descritto dal listing **??**. 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.

[caption=Calcolo della posizione corrente., label=lst:geo$_c$$urrent, language = Kotlin]valearth = session.earth//VerificochesiainstatoTRACKINGif(earth?.trackingState == TrackingState.TRACKING)valcameraGeospatialPose : GeospatialPose = earth.cameraGeospatia // Salvataggio della latitudine in gradi val latitude : Double = cameraGeospatialPose.latitude // Salvataggio della longitudine in gradi val longitude : Double = cameraGeospatialPose.longitude // Salvataggio dell'altitudine in metri val elevation : Double = cameraGeospatialPose.altitude // Salvataggio dell'orientamento in gradi val heading : Double = cameraGeospatialPose.heading

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

1.1.3 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 **??** per un esempio di creazione dell'anchor.

[caption=Posizionamento di anchor Geospatial., label=lst:geo$_p$os, $language = Kotlin]if(earth.trackingState == TrackingState.TRACKING)valanchor = earth.createAnchor(/ * Valoridellaposizione * /latitude, longitude, al

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.