

Chapter 1


Depth understanding

1.1 Depth API

ARCore Depth API aggiunge un livello di realismo attraverso l'uso di algoritmi che generano immagini o mappe di profondità. Questi algoritmi sono in grado di ottenere stime di profondità fino a 65 metri. Un'immagine di profondità (figura ??) offre una visualizzazione 3D del mondo reale, ogni pixel è associato alla distanza dalla scena e attraverso l'uso di colori differenti è possibile riconoscere quali aree dello spazio sono più vicine al dispositivo. La profondità viene acquisita con piccoli spostamenti dai quali si ottengono misure più efficienti (fino a 5 metri). Per ciascun frame può essere recuperata l'immagine di profondità corrispondente dopo che l'utente abbia iniziato a muoversi con il dispositivo. Depth API richiede una grande potenza di elaborazione ed è supportata solo da alcuni dispositivi; è necessario controllare se il dispositivo è compatibile e nel caso lo fosse attivare la profondità manualmente nella configurazione della sessione ARCore.

Le principali funzionalità offerte da Depth API sono tre:

- **Copertura dei contenuti:** permette di posizionare accuratamente dei contenuti virtuali di fronte o dietro degli oggetti reali.
- **Immersione:** permette di decorare una scena con oggetti virtuali che interagiscono tra di loro.
- **Interazione:** i contenuti virtuali sono in grado di interagire con il mondo reale attraverso cambiamenti fisici e collisioni.



`./resources/images/depthAPI/DepthMap.png`

Fonte: <https://developers.google.com/ar/develop/depth>

Figure 1.1: Esempio di mappa di profondità.

1.1.1 Sessione ARCore con Depth API

Prima di iniziare una nuova sessione ARCore è necessario controllare se il dispositivo supporta Depth API. A volte questa opzione può essere disattivata oppure non supportata nonostante il dispositivo supporti ARCore. Dopo aver definito la sessione con le opportune configurazioni è possibile controllare se il dispositivo e la fotocamera supportano una determinata modalità di profondità invocando il metodo `isDepthModeSupported(Config.DepthMode mode)` sull'istanza della sessione. Se la modalità è supportata viene configurata la sessione e sarà possibile sfruttare depth API [?]. Si veda il listing ?? per un esempio di configurazione.

```
[caption=Controllo supporto depth API., label=lst: check-api-depth, language=Kotlin]
val config = session.config
// Verifica se il dispositivo supporta Depth API.
val isDepthSupported = session.isDepthModeSupported(Config.DepthMode.AUTOMATIC)
if (isDepthSupported) config.depthMode = Config.DepthMode.AUTOMATIC
session.configure(config)
```


Per ottenere l'immagine di profondità relativa al frame corrente viene invocato il metodo `acquireDepthImage16Bits()`, come mostrato dal listing ??.

```
[caption=Estrazione di un'immagine profonda., label=lst: depth-image, language=Kotlin]
val frame = arFragment.arSceneView.frame
// Estrai il depth image per il frame corrente, se disponibile.
try {
    frame.acquireDepthImage16Bits().use { depthImage -> // Uso del depth image
        // I dati sulla profondità non sono disponibili se non ci sono feature point // tracciati. Questo può succedere se non c'è movimento o quando la fotocamera // non è più in grado di tracciare gli oggetti dell'ambiente circostante.
    }
}
```

```
[caption=Estrazione di informazioni da un'immagine profonda., label=lst:
inf-depth-img, language=Kotlin] // Restituisce la profondità in mil-
limetri di depthImage alle coordinate ([x], [y]). fun getMillimeters-
Depth(depthImage:Image, x:Int, y:Int): Int
// Il depth image ha un singolo piano, che salva la profondità per ogni
pixel // in una variabile unsigned integer a 16 bit. val plane = depthIm-
age.planes[0]
// Distanza in byte tra campioni di pixel adiacenti. val pixelStride= x *
plane.pixelStride
// Row stride in byte per il piano. val rowStride= y * plane.rowStride
val byteIndex = pixelStride + rowStride
// Recupera l'ordine in byte del buffer. val buffer =
plane.buffer.order(ByteOrder.nativeOrder())
// Leggi due byte all'indice dato, componendoli in un valore pic-
colo // in base al corrente ordine in byte. val depthSample =
buffer.getShort(byteIndex)
return depthSample.toInt()
```

1.2 Raw Depth API

Le *ARCore Raw Depth API* forniscono informazioni più precise sulla profondità di alcuni pixel di un'immagine e permettono di rappresentare la geometria della scena generando delle **immagini di profondità grezze**. Ad esse vengono associate delle **immagini di affidabilità** che stabiliscono il grado di confidenza di ciascun pixel (associato ad un valore di profondità) [?]. In particolare, ogni pixel è un numero intero senza segno a 8 bit che indica la stima della confidenza con valori compresi tra 0 (più bassa) e 255 (più alta) inclusi. I pixel senza una stima della profondità valida hanno un valore di confidenza pari a 0 e un valore di profondità nullo.



`./resources/images/depthAPI/deptRawAPI.png`

Fonte: <https://stackoverflow.com/questions/59088045/arcore-raw-depth-data-from-rear-android-depth-camera>

Figure 1.2: Differenze tra immagini in API Raw Depth.

L'uso di Raw Depth API è molto utile nei casi in cui c'è il bisogno di avere una maggiore precisione, ad esempio nella **misurazione** (*PHORIA ARConnect App*), **ricostruzione 3d** (*3d Live Scanner App*), **rilevamento delle forme** (*Jam3*), mostrati nella figura ??.



Fonte: <https://www.youtube.com/watch?v=13WugTMOdSs>

Figure 1.3: Applicazioni Raw Depth API

1.2.1 Sessione ARCore con Raw depth API

Inizialmente si deve effettuare il controllo sulla compatibilità del dispositivo come riportato nel codice ???. Per acquisire un'immagine di confidenza viene invocato il metodo `acquireRawDepthConfidenceImage()`, dalla quale si può ricavare l'accuratezza di ogni pixel di profondità non elaborato [?]. Si veda il listing ???.

```
[caption=Estrazione di un'immagine di confidenza., label=lst: confidence-
image, language=Kotlin] try // All'inizio, recupera un raw depth image.
// Un Raw Depth image è un uint16, con GPU aspect ratio e orientamento
nativo. frame.acquireRawDepthImage16Bits().use rawDepth -;
// Il Confidence image è un uint8, in corrispondenza alla dimensione di
depth image. frame.acquireRawDepthConfidenceImage().use rawDepth-
Confidence -;
// Compara i timestamp per determinare se la distanza è basata su nuovi //
depth data, o se è una proiezione basata sul movimento del dispositivo. val
thisFrameHasNewDepthData = frame.timestamp == rawDepth.timestamp
if (thisFrameHasNewDepthData) val depthData =
rawDepth.planes[0].buffer val confidenceData = rawDepthConfidence.
planes[0].buffer val width = rawDepth.width val height =
rawDepth.height someReconstructionPipeline.integrateNewImage( depth-
Data, confidenceData, width = width, height = height ) catch (e:
NotYetAvailableException) // Depth image non è ancora disponibile.
```

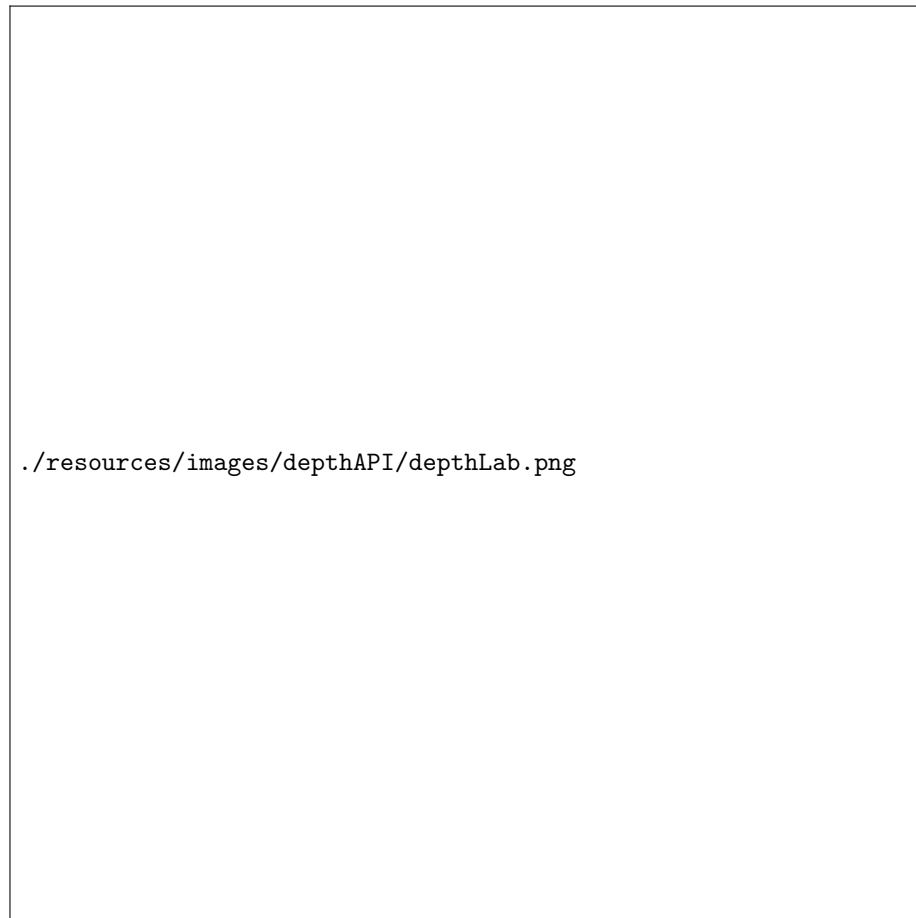
1.3 Depth Hit Test

Integrando la profondità sono stati ottenuti degli hit test più precisi grazie ai quali è possibile posizionare contenuti virtuali anche su superfici non piane in aree con bassa texture [?]. Si veda il listing ??.

```
[caption=Depth Hit Test, label=lst: Depth Hit Test, language=Kotlin] val
frame = arFragment.arSceneView.frame
val hitResultList: List<HitResult> = frame.hitTest(tap)
for(hit in hitResultList) val trackable: Trackable=hit.trackable
if(trackable is Plane —— trackable is Point —— trackable is DepthPoint)
val anchor = hit.createAnchor() // Uso di anchor...
```

1.4 Depth Lab

Depth Lab è una libreria software che incapsula un insieme ristretto di depth API grazie alle quali è possibile recuperare risorse utili per **rilevamento** della geometria e **rendering** nell'interazione AR. Si veda la figura ??.



`./resources/images/depthAPI/depthLab.png`

Fonte: https://augmentedperception.github.io/depthlab/assets/DuDepthLab-Real-Time3DInteractionWithDepthMapsForMobileAugmentedReality_UIST2020.pdf

Figure 1.4: Panoramica ad alto livello di Depth Lab

Depth Lab è costituito da quattro componenti [?]:

- **Tracking and Input:** vengono utilizzate immagini di profondità (ottenute da Depth API), posa del dispositivo, altri parametri della fotocamera per stabilire la mappatura del mondo fisico e degli oggetti virtuali nella scena.
- **Data structure:** i valori di profondità associati a ciascun pixel sono memorizzati all'interno di un'immagine prospettica (a bassa risoluzione). Esistono 3 strutture dati differenti:

- *array di profondità*: memorizza la profondità in un array 2D di un'immagine orizzontale con numeri interi a 16 bit. È possibile ricavare il valore della profondità di un qualsiasi punto dello schermo.
 - *mesh di profondità*: fornisce una ricostruzione 3D in tempo reale dell'ambiente circostante per ciascuna mappa di profondità. In figura (a) di ?? è rappresentata un'immagine di profondità che specifica il valore di profondità di ciascun pixel in base al colore (i pixel più luminosi indicano regioni più lontane). Ognuno di questi valori verrà proiettato nel template mesh tassellato generando una mesh di profondità in tempo reale (c) costituita dall'interconnessione di superfici triangolari.
 - *texture della profondità* è una texture che rappresenta la profondità della scena inquadrata dalla fotocamera.
- **Conversion Utilities and Algorithm** : l'utilizzo di algoritmi di conversione è diverso a seconda della funzionalità che viene offerta. Per gestire la fisica dell'ambiente, ombre, occlusione, mappatura di texture e molto altro sono necessari diversi approcci per l'elaborazione della profondità.
 - **DepthLab Algorithms**: sulla base delle strutture dati gli algoritmi si dividono in 3 categorie:
 - *profondità localizzata*: adatto per calcolare misurazioni fisiche, stimare vettori normali, muovere avatar virtuali in giochi AR (creare dei percorsi per evitare che l'avatar collida con qualche oggetto come nella figura ??). Utilizza l'array di profondità per operare su un numero ridotto di punti.
 - *profondità superficiale*: aggiorna mesh di profondità per gestire collisioni, texture decal (texture che vengono applicate su una superficie di un oggetto con una specifica proiezione), fisica dell'ambiente, riconoscimento geometrico delle ombre. (Esempi in figura ??)
 - *profondità densa*: utilizzato sul rendering di effetti sensibili alla profondità, reilluminazione, messa a fuoco, occlusione. (Esempi in figura ??)



Fonte:
<https://augmentedperception.github.io/depthlab/assets>

Figure 1.5: Generazione di una depth mesh.



Fonte: <https://augmentedperception.github.io/depthlab/assets/>

Figure 1.6: Pianificazione di un percorso per un avatar virtuale.



`./resources/images/depthAPI/surfacesdepth.png`

Fonte: <https://augmentedperception.github.io/depthlab/assets/>

Figure 1.7: Esempi di profondità superficiale.



Fonte: <https://augmentedperception.github.io/depthlab/assets/>

Figure 1.8: Esempi di profondità densa.