

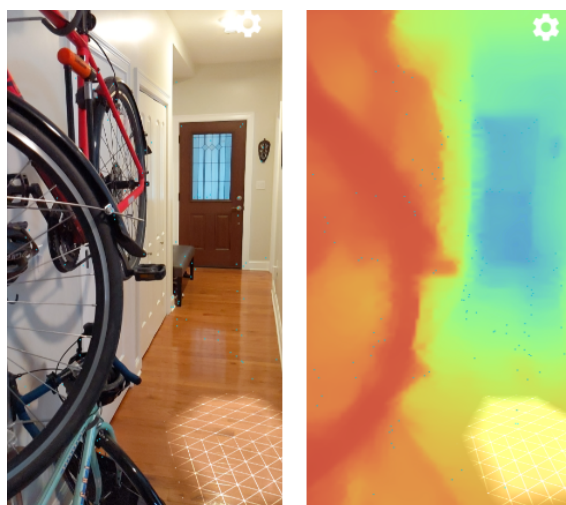
# 1. Depth understanding

## 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 1.1) 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 principale 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.



Fonte:

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

Figura 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 [google2022depth]. (Codice 1.1)

```
1
2  val config = session.config
3
4  // Check whether the user's device supports the Depth API.
5  val isDepthSupported = session.isDepthModeSupported(Config.DepthMode.AUTOMATIC)
6  if (isDepthSupported) {
7      config.depthMode = Config.DepthMode.AUTOMATIC
8  }
9  session.configure(config)
```

Listing 1.1: Controllo supporto depth API

Per ottenere l'immagine di profondità relativa al frame corrente viene invocato il metodo `acquireDepthImage16Bits()`. (Codice 1.2)

```
1  val frame = arFragment.arSceneView.frame
2
3  // Retrieve the depth image for the current frame, if available
4  try {
5      frame.acquireDepthImage16Bits().use { depthImage ->
6          // Use the depth image here
7      }
8  } catch (e: NotYetAvailableException) {
9      // This means that depth data is not available yet.
10     // Depth data will not be available if there are no tracked
11     // feature points. This can happen when there is no motion, or when the
12     // camera loses its ability to track objects in the surrounding
13     // environment.
14 }
```

Listing 1.2: Estrazione di un'immagine profonda

```

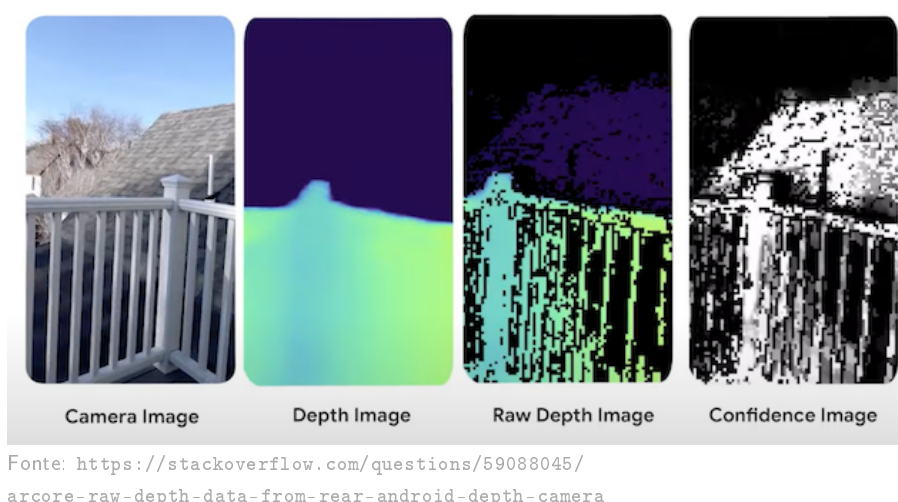
1  /** Obtain the depth in millimeters for [depthImage] at coordinates ([x], [y]). */
2  fun getMillimetersDepth(depthImage: Image, x: Int, y: Int): Int {
3
4      // The depth image has a single plane, which stores depth for each
5      // pixel as 16-bit unsigned integers.
6      val plane = depthImage.planes[0]
7
8      //The distance between adjacent pixel samples, in bytes
9      val pixelStride= x * plane.pixelStride
10
11     //The row stride for this color plane, in bytes.
12     val rowStride= y * plane.rowStride
13
14     val byteIndex = pixelStride + rowStride
15
16     //Retrieves this buffer's byte order.
17     val buffer = plane.buffer.order(ByteOrder.nativeOrder())
18
19     //Reads two bytes at the given index, composing them into a short value
20     // according to the current byte order.
21     val depthSample = buffer.getShort(byteIndex)
22
23     return depthSample.toInt()
24 }

```

Listing 1.3: Estrazione di informazioni da un'immagine profonda

## 1.2 Raw Depth API

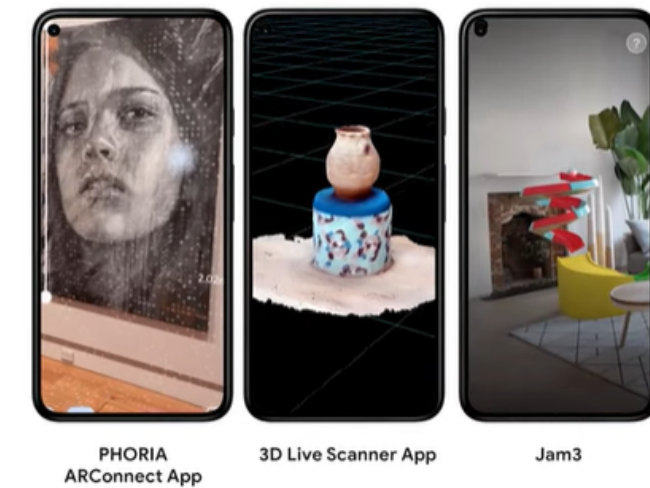
Le ARCore Raw Depth API forniscono informazioni più precise sulla profondità di alcuni pixel di un'immagine e permettono di rappresentare la geometria della scena generando delle **immagini di profondità grezze**. A queste immagini vengono associate delle **immagini di affidabilità** che stabiliscono il grado di confidenza di ciascun pixel (associato ad un valore di profondità) [mobilear2021rawdepth]. 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.



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

Figura 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).



Fonte: <https://www.youtube.com/watch?v=13WugTM0dSs>

Figura 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 ?? a pagina ??.

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 [google2022rawdepth]. (Codice riportato 1.4)

```

1  try {
2
3      //First of all, retrieve raw depth image
4      // Raw Depth image is in uint16, at GPU aspect ratio, in native orientation.
5      frame.acquireRawDepthImage16Bits().use { rawDepth ->
6
7          // Confidence image is in uint8, matching the depth image size.
8          frame.acquireRawDepthConfidenceImage().use { rawDepthConfidence ->
9
10
11         // Compare timestamps to determine whether depth is based on new
12         // depth data, or is a reprojection based on device movement.
13         val thisFrameHasNewDepthData = frame.timestamp == rawDepth.timestamp
14
15         if (thisFrameHasNewDepthData) {
16
17             val depthData = rawDepth.planes[0].buffer
18             val confidenceData = rawDepthConfidence.planes[0].buffer
19             val width = rawDepth.width
20             val height = rawDepth.height
21             someReconstructionPipeline.integrateNewImage(
22                 depthData,
23                 confidenceData,
24                 width = width,
25                 height = height
26             )
27         }
28     }
29 } catch (e: NotYetAvailableException) {
30     // Depth image is not (yet) available.
31 }

```

Listing 1.4: Estrazione di un'immagine di confidenza

### 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 [google2022depth]. (Codice in 1.5)

```

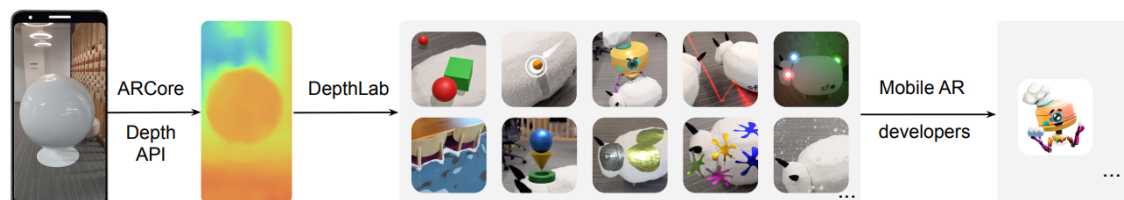
1      val frame = arFragment.arSceneView.frame
2
3      val hitResultList: List<HitResult> = frame.hitTest(tap)
4
5      for (hit in hitResultList) {
6          val trackable: Trackable = hit.trackable
7
8          if (trackable.isPlane || trackable.isPoint || trackable.isDepthPoint) {
9              val anchor = hit.createAnchor()
10             // Use anchor here
11         }
12     }

```

Listing 1.5: Depth Hit Test

### 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 **rilievo** della geometria e **rendering** nell'interazione AR. (Figura 1.4)



Fonte: [https://augmentedperception.github.io/depthlab/assets/Du\\_DepthLab-Real-Time3DInteractionWithDepthMapsForMobileAugmentedReality\\_UIST2020.pdf](https://augmentedperception.github.io/depthlab/assets/Du_DepthLab-Real-Time3DInteractionWithDepthMapsForMobileAugmentedReality_UIST2020.pdf)

Figura 1.4: Panoramica ad alto livello di Depth Lab

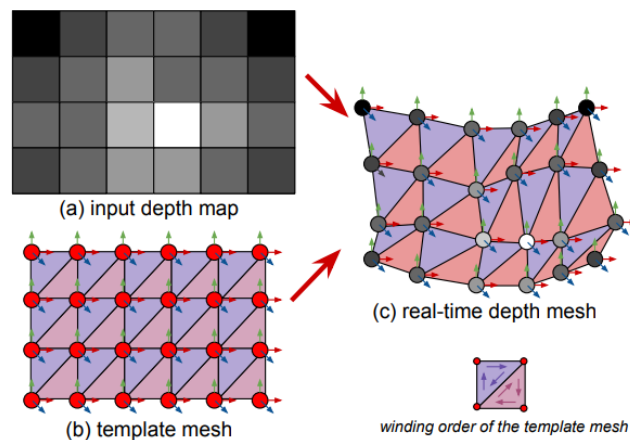
Depth Lab è costituito da quattro componenti [ruofeidu2020depthlab]:

- **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. E' 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) in 1.5 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 1.6 a fronte). 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 1.7 nella pagina successiva)
    - *profondità densa*: utilizzato sul rendering di effetti sensibili alla profondità, riilluminazione, messa a fuoco, occlusione. (Esempi in figura 1.8 a fronte)



Fonte:

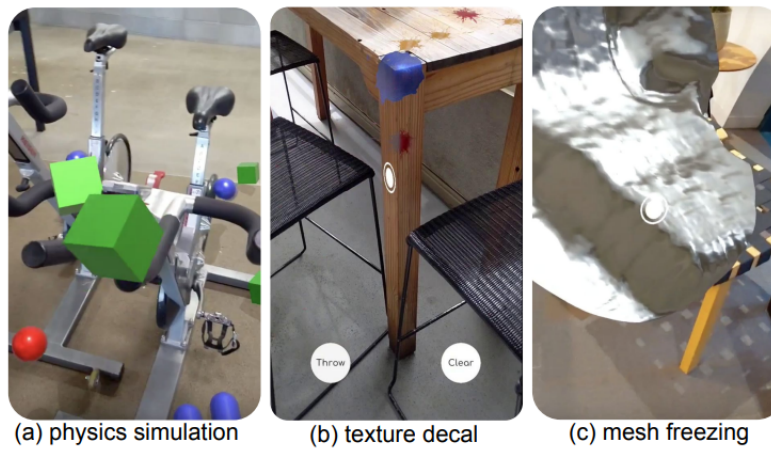
[https://augmentedperception.github.io/depthlab/assets/Du\\_DepthLab-Real-Time3DInteractionWithDepthMapsForMobileAugmentedReality\\_UIST2020.pdf](https://augmentedperception.github.io/depthlab/assets/Du_DepthLab-Real-Time3DInteractionWithDepthMapsForMobileAugmentedReality_UIST2020.pdf)

Figura 1.5: Generazione di una depth mesh



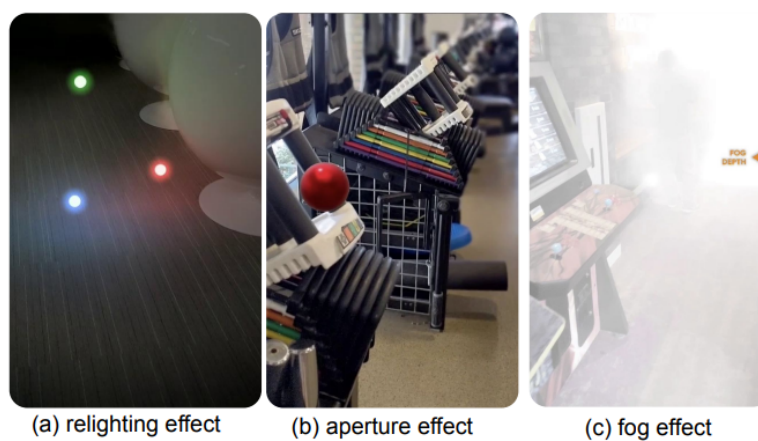
Fonte: [https://augmentedperception.github.io/depthlab/assets/Du\\_DepthLab-Real-Time3DInteractionWithDepthMapsForMobileAugmentedReality\\_UIST2020.pdf](https://augmentedperception.github.io/depthlab/assets/Du_DepthLab-Real-Time3DInteractionWithDepthMapsForMobileAugmentedReality_UIST2020.pdf)

Figura 1.6: Pianificazione di un percorso per un avatar virtuale



Fonte: [https://augmentedperception.github.io/depthlab/assets/Du\\_DepthLab-Real-Time3DInteractionWithDepthMapsForMobileAugmentedReality\\_UIST2020.pdf](https://augmentedperception.github.io/depthlab/assets/Du_DepthLab-Real-Time3DInteractionWithDepthMapsForMobileAugmentedReality_UIST2020.pdf)

Figura 1.7: Esempi di profondità superficiale



Fonte: [https://augmentedperception.github.io/depthlab/assets/Du\\_DepthLab-Real-Time3DInteractionWithDepthMapsForMobileAugmentedReality\\_UIST2020.pdf](https://augmentedperception.github.io/depthlab/assets/Du_DepthLab-Real-Time3DInteractionWithDepthMapsForMobileAugmentedReality_UIST2020.pdf)

Figura 1.8: Esempi di profondità densa