ARCore Image Tracking

Sensor Based Mobile Applications

Patrick Ausderau, Ulla Sederlöf, Jarkko Vuori
Original author: Kari Salo

Helsinki Metropolia University of Applied Science

2022

Metropolia

Outline

Augmented Reality Recap

Image Tracking

AR Tracking App

Lab

For more info:

`https://developers.google.com/ar/develop/java/augmented-images/`

`https://developers.google.com/ar/develop/java/augmented-images/arcoreimg`

Augmented Reality Recap

- Check this article in order to understand ARcore terminology and workflow: `https://blog.novoda.com/getting-started-with-google-arcore-on-android/`

Images as Trackables

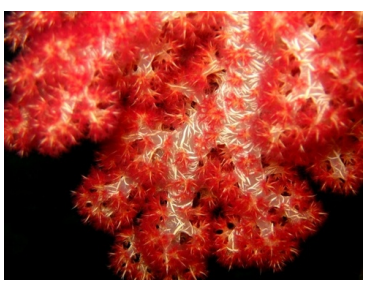
- ARCore is capable of detecting 2D images in the camera view
- Reference images will be stored into image database
- database can store information up to 1,000 reference images
- ARCore can track up to 20 images simultaneously
- Use arcoreimg tool[1] to find suitable images[2]
  - Run arcoreimg eval-img to get a quality score (0 ... 100) for image - score should be at least 75.
  - `arcoreimg eval-img --input_image_path=my_img.jpg`

[1] `https://github.com/google-ar/arcore-android-sdk/tree/master/tools/arcoreimg`

[2] `https://developers.google.com/ar/develop/java/augmented-images#best_practices`

Example

```
$ cd Downloads/arcore-android-sdk-1.33.0/tools/arcoreimg/linux/
$ ./arcoreimg  eval-img --input_image_path=corals.jpg
100
$ ./arcoreimg  eval-img --input_image_path=sofa.jpg
30
$ ./arcoreimg  eval-img --input_image_path=sunflower.jpg
60
$ ./arcoreimg  eval-img --input_image_path=scenery.jpg
45
```

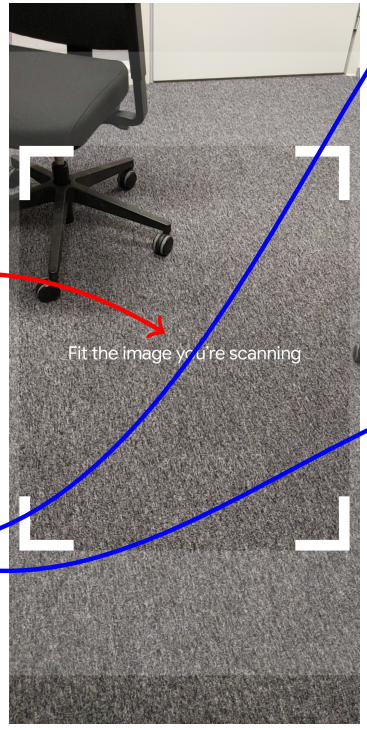


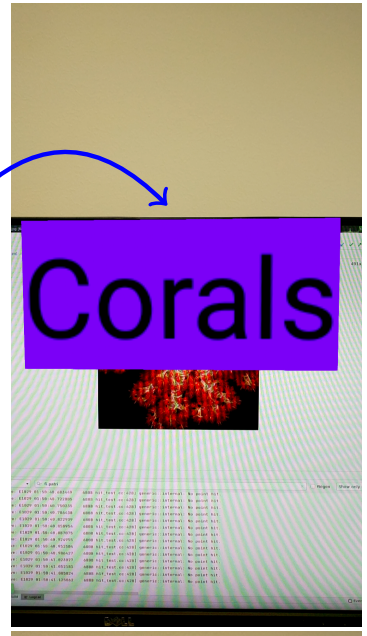

ArFragment subclass

- Create your own ArFragment class
  - Disable plane renderer and turn off instructions controller by overriding `onCreateView` method
  - Create image database and set it as a part of session configuration by overriding `onCreateSessionConfig` – this enables ARCore session to start tracking images

AR Tracking App

- Let's create an app, which detects two different images inside camera view (utilising ArFragment). When an image is tracked, then a respective info will be displayed over the image with a TextView.





Dependencies

- In manifest, add permission, feature and metadata:

```
1 <uses-permission android:name="android.permission.CAMERA" />
2 <uses-feature android:name="android.hardware.camera.ar"
      android:required="true" />
3 <application>
4   <!-- ... -->
5   <meta-data android:name="com.google.ar.core"
        android:value="required" />
6 </application>
```

Dependencies (Continued...)

- In module build gradle:

```
//...
android {
  defaultConfig {
    //...
    minSdkVersion 24
  }
  //...
  compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
  }
}
//...
dependencies {
  implementation "com.google.ar:core:1.33.0"
  implementation "com.gorisse.thomas.sceneform:sceneform:1.21.0"
  //...
}
```

Activity Layout

```
1  <FrameLayout><!-- xmlns,... -->
2  <!-- FrameLayout: display elements on top of each other -->
3    <fragment
4      android:id="@+id/fragArImg"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:name="fi.patricka.kotlinartracking.TrackImgFrag" />
8      <!-- note: own implementation that override ArFragment  -->
9
10   <ImageView
11     android:id="@+id/fitToScanImg"
12     android:layout_width="match_parent"
13     android:layout_height="match_parent"
14     android:layout_gravity="center"
15     android:scaleType="fitCenter"
16     android:src="@drawable/fit_to_scan"
17     android:contentDescription="@string/fit_image_to_scan" />
18 </FrameLayout>
```

https://developers.google.com/ar/images/FitToScan.png

ViewRenderable Layout

- ViewRenderable needs it's own layout

```
1  <LinearLayout><!-- xmlns,... -->
2
3    <TextView
4      android:id="@+id/txtImgTrack"
5      android:layout_width="wrap_content"
6      android:layout_height="wrap_content"
7      android:background="@color/purple_200"
8      android:textColor="@color/black"
9      android:textSize="24sp" />
10 </LinearLayout>
```

Activity – Attributes and onCreate(), onResume()

```
1  private lateinit var arFrag: ArFragment
2  private var viewRenderable: ViewRenderable? = null
3
4  //onCreate(), super, layout...
5    arFrag = supportFragmentManager.findFragmentById(R.id.fragArImg)
         as ArFragment
6
7    ViewRenderable.builder()
8        .setView(this, R.layout.rend_text)
9        .build()
10       .thenAccept { viewRenderable = it }
11
12 //onResume(), super,...
13   arFrag.arSceneView.scene.addOnUpdateListener { frameUpdate() }
```

Activity – `frameUpdate()`

```
1  private fun frameUpdate() {
2    val arFrame = arFrag.arSceneView.arFrame
3    if (arFrame == null || arFrame.camera.trackingState !=
         TrackingState.TRACKING) return
4
5    val updatedAugmentedImages =
         arFrame.getUpdatedTrackables(AugmentedImage::class.java)
6    updatedAugmentedImages.forEach {
7      when (it.trackingState) {
8        null -> return@forEach
9        //...
10     }
11   }
12 }
```

Activity – `frameUpdate()` (Continued...)

```
1  //frameUpdate continued... when (it.trackingState) { ...
2    TrackingState.PAUSED -> {
3      // Image initially detected, but not enough data available to
           estimate its location in 3D space.
4      // Do not use the image's pose and size estimates until the
           image's tracking state is tracking
5      val text = getString(R.string.detected_img_need_more_info,
           it.name)
6      Toast.makeText(this, text, Toast.LENGTH_SHORT).show()
7    }
8    TrackingState.STOPPED -> {
9      val text = getString(R.string.track_stop, it.name)
10     Toast.makeText(this, text, Toast.LENGTH_SHORT).show()
11   }
12   //...
```

Activity – frameUpdate() (Continued...)

```
1  //frameUpdate continued... when (it.trackingState) { ...
2    TrackingState.TRACKING -> {
3      val anchors = it.anchors
4      if (anchors.isEmpty()) {
5        findViewById<ImageView>(R.id.fitToScanImg).visibility =
             View.GONE
6        // Create anchor and anchor node in the center of the image.
7        val pose = it.centerPose
8        val anchor = it.createAnchor(pose)
9        val anchorNode = AnchorNode(anchor)
10       //Attach anchor node in the scene
11       anchorNode.parent = arFrag.arSceneView.scene
12       // Create a node as a child node of anchor node, and define
             node's renderable according to augmented image
13       val imgNode = TransformableNode(arFrag.transformationSystem)
14       imgNode.parent = anchorNode
15       viewRenderable?.view?.findViewById<TextView>(
             R.id.txtImgTrack)?.text = it.name
16       imgNode.localRotation = Quaternion.axisAngle(Vector3(1f, 0f,
             0f), -90f)
17       imgNode.renderable = viewRenderable
18 }}}
```

ArFragment

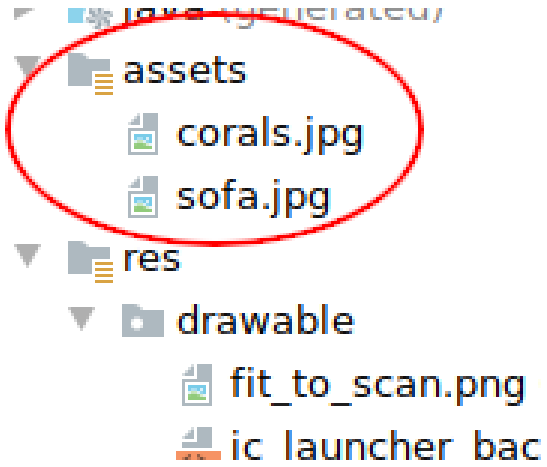
```
1  // extends ArFragment in order to override defaults
2  class TrackImgFrag: ArFragment() {
3
4    override fun onCreateView(inflater: LayoutInflater, container:
         ViewGroup?, savedInstanceState: Bundle?): View? {
5      val view =  super.onCreateView(inflater, container,
           savedInstanceState)
6      // Disable plane renderer and turn off instruction controller
7      arSceneView.planeRenderer.isVisible = false
8      arSceneView.planeRenderer.isEnabled = false
9      instructionsController.isEnabled = false
10     instructionsController.isVisible = false
11     return view
12   }
13   //...
14 }
```

ArFragment (Continued...)

```
1  override fun onCreateSessionConfig(session: Session?): Config {
2    val config = super.onCreateSessionConfig(session)
3    // Create image database and set it as a part of session
         configuration
4    setupAugmentedImageDatabase(config, session)
5    return config
6  }
```

ArFragment (Continued...)

```
1  private fun setupAugmentedImageDatabase(config: Config, session:
       Session?) {
2    val augmentedImageDb = AugmentedImageDatabase(session)
3    val assetManager = requireContext().assets
4    listOf("sofa", "corals").forEach {
5      val inputStream = assetManager.open("$it.jpg")
6      val augmentedImageBitmap =
           BitmapFactory.decodeStream(inputStream)
7      augmentedImageDb.addImage(it, augmentedImageBitmap, 0.1f)
8    }
9    config.augmentedImageDatabase = augmentedImageDb
10 }
```



Lab_w4_d5 AR Tracking

- Create Image Recognition application, which detects more than one image. Detecting image will result displaying either 2D widget or 3D model in front of the image/above the image.
- When designing this app you need to think potential users – who would use your app and why.