Understanding ARKit Tracking and Detection

Session 610

Marion Maerz, ARKit Engineer
Michele Stoppa, ARKit Engineer
Orientation Tracking
Orientation Tracking

World Tracking
Orientation Tracking

World Tracking

Plane Detection
Saving and Loading Maps
• Saving and Loading Maps
• Image Tracking
Saving and Loading Maps

Image Tracking

Object Detection
Basics of ARKit

Recap

ARSesson
Basics of ARKit

Recap

ARConfiguration

ARSession
Basics of ARKit

Recap

```
run(_ configuration)
```

```
ARConfiguration  ── ARSession
```
Basics of ARKit

Recap

run(_ configuration)

ARConfiguration → ARSession

AVCaptureSession, CMMotionManager
Basics of ARKit

Recap

- run(_ configuration)
- currentFrame

ARConfiguration → ARSession

AVCaptureSession
CMMotionManager

ARFrame
Orientation Tracking
Orientation Tracking

Tracks orientation only (3 DoF)
Orientation Tracking

Tracks orientation only (3 DoF)

Spherical virtual environment
Orientation Tracking

Tracks orientation only (3 DoF)

Spherical virtual environment

Augmentation of far objects
Orientation Tracking

- Tracks orientation only (3 DoF)
- Spherical virtual environment
- Augmentation of far objects
- Not suited for physical world augmentation from different views
Behind the Scenes
Rotation data

ARFrame

Rotation

Image

...
Orientation Tracking API

AROrientationTrackingConfiguration
Orientation Tracking API

AROrientationTrackingConfiguration

open class ARCamera : NSObject, NSCopying {

    open var transform: simd_float4x4 { get }

    open var eulerAngles: simd_float3 { get }
    ...
}

Orientation Tracking API

AROrientationTrackingConfiguration

open class ARCamera : NSObject, NSCopying {

    open var transform: simd_float4x4 { get }

    open var eulerAngles: simd_float3 { get }
    ...
}


Orientation Tracking API

AROrientationTrackingConfiguration

open class ARCamera : NSObject, NSCopying {

    open var transform: simd_float4x4 { get }

    open var eulerAngles: simd_float3 { get }

    ...
}

Behind the Scenes—Inertial Odometry

Motion sensor

ARFrame

+ 

Motion

Image

Time
Behind the Scenes—Inertial Odometry

Motion sensor

ARFrame + Motion + Image

ARFrame + Motion + Image

ARFrame + Motion + Image

ARFrame + Motion + Image

Time
Behind the Scenes—Inertial Odometry

Motion sensor

ARFrame + Motion + Image

ARFrame + Motion + Image

ARFrame + Motion + Image

ARFrame + Motion + Image

ARFrame + Motion + Image

Time

1 m
Behind the Scenes—Visual Odometry

Computer vision

Image

ARFrame

CV

Time
Behind the Scenes—Visual Odometry

Computer vision

ARFrame → CV → Image

ARFrame → CV → Image

ARFrame → CV → Image

ARFrame → CV → Image

ARFrame → CV → Image

Time
Behind the Scenes—Visual Inertial Odometry

Motion data and computer vision

ARFrame + Image

CV

Motion

Image

ARFrame + Image

CV

Motion

Image

ARFrame + Image

CV

Motion

Image

ARFrame + Image

CV

Motion

Image

1 m
Behind the Scenes—Visual Inertial Odometry

Extracting robust features
Behind the Scenes—Visual Inertial Odometry

Matching features
Behind the Scenes—Visual Inertial Odometry

Triangulation creates World Map
Behind the Scenes—Visual Inertial Odometry

Triangulation creates World Map
Behind the Scenes—Visual Inertial Odometry

Initialization
Behind the Scenes—Visual Inertial Odometry

Initialization
Behind the Scenes—Visual Inertial Odometry

Initialization
Behind the Scenes
Augmentation
Behind the Scenes
Augmentation
Behind the Scenes
Augmentation
Behind the Scenes
Augmentation
Behind the Scenes
Augmentation
Behind the Scenes
Augmentation
Behind the Scenes
Augmentation
Behind the Scenes—Visual Inertial Odometry Tracking
Behind the Scenes—Visual Inertial Odometry Tracking
Behind the Scenes—Visual Inertial Odometry Tracking
Behind the Scenes—Visual Inertial Odometry Tracking
Behind the Scenes—Visual Inertial Odometry Tracking
Behind the Scenes—Visual Inertial Odometry Tracking
Behind the Scenes—Visual Inertial Odometry Tracking
Behind the Scenes—Visual Inertial Odometry
Optimization for robust tracking
Behind the Scenes—Visual Inertial Odometry
Optimization for robust tracking
Behind the Scenes—Visual Inertial Odometry
Optimization for robust tracking
Behind the Scenes—Visual Inertial Odometry
Optimization for robust tracking
Behind the Scenes—Visual Inertial SLAM
Optimization for robust tracking
Behind the Scenes—Visual Inertial SLAM
Optimization for robust tracking
Behind the Scenes—Visual Inertial SLAM
Optimization for robust tracking
Behind the Scenes—Visual Inertial SLAM
Optimization for robust tracking
World Tracking API

ARWorldTrackingConfiguration
World Tracking API

ARWorldTrackingConfiguration

open class ARCamera : NSObject, NSCopying {

    open var transform: simd_float4x4 { get }

    open var trackingState: ARTrackingState { get }
    open var trackingStateReason: ARTrackingStateReason { get }

    ...
}

World Tracking API

```swift
open class ARCamera : NSObject, NSCopying {

open var transform: simd_float4x4 { get }

open var trackingState: ARTrackingState { get }
open var trackingStateReason: ARTrackingStateReason { get }

...
}
```
World Tracking API

open class ARCamera : NSObject, NSCopying {

    open var transform: simd_float4x4 { get }

    open var trackingState: ARTrackingState { get }
    open var trackingStateReason: ARTrackingStateReason { get }

    ...
}

ARWorldTrackingConfiguration
World Tracking Quality

Uninterrupted sensor data
World Tracking Quality

Uninterrupted sensor data

Textured environments
World Tracking Quality

Uninterrupted sensor data

Textured environments
World Tracking Quality

Uninterrupted sensor data
Textured environments
Static scenes
World Tracking Quality

Uninterrupted sensor data

Textured environments

Static scenes
World Tracking Quality—Behind the Scenes
“Health Monitoring”
World Tracking Quality—Behind the Scenes

“Health Monitoring”
Tracking State

Not Available  Limited  <Initializing>  Normal  Limited  <Insufficient Features>  Normal
// Called when tracking state changed
func session(_ session: ARSession, cameraDidChangeTrackingState camera: ARCamera) {
    if case .limited(let reason) = camera.trackingState {
        // Guide user to improve tracking state
        ...
    }
}
Tracking State

// Called when tracking state changed
func session(_ session: ARSession, cameraDidChangeTrackingState camera: ARCamera) {
    if case .limited(let reason) = camera.trackingState {
        // Guide user to improve tracking state
        ...
    }
}
World Tracking

Take-aways

Tracks orientation and position (6 DoF)
World Tracking

Take-aways

Tracks orientation and position (6 DoF)
World Tracking
Take-aways

Tracks orientation and position (6 DoF)
Augmentation into your physical world
World Tracking

Take-aways

Tracks orientation and position (6 DoF)

Augmentation into your physical world

World Map
World Tracking

Take-aways

Tracks orientation and position (6 DoF)

Augmentation into your physical world

World Map

Guide user to achieve best tracking quality
World Tracking
Take-aways

- Tracks orientation and position (6 DoF)
- Augmentation into your physical world
- World Map
- Guide user to achieve best tracking quality
- Runs on your device only
World Tracking

Take-aways

Tracks orientation and position (6 DoF)
Augmentation into your physical world

World Map
Guide user to achieve best tracking quality

Runs on your device only

“Building Your First AR Experience”
Plane Detection
Behind the Scenes

Frame

3D points
aggregate

Frame

3D points
aggregate

Frame

3D points
aggregate

Frame

3D points
aggregate

Planes

merge

Time
// Create a world tracking configuration
let configuration = ARWorldTrackingConfiguration()

// Enable plane detection
configuration.planeDetection = [.horizontal, .vertical]

// Run the session
session.run(configuration)
// Create a world tracking configuration
let configuration = ARWorldTrackingConfiguration()

// Enable plane detection
configuration.planeDetection = [.horizontal, .vertical]

// Run the session
session.run(configuration)
// Create a world tracking configuration
let configuration = ARWorldTrackingConfiguration()

// Enable plane detection
configuration.planeDetection = [.horizontal, .vertical]

// Run the session
session.run(configuration)
open class ARPlaneAnchor : ARAnchor {

   open var transform: simd_float4x4 { get }

   open var center: simd_float3 { get }
   open var extent: simd_float3 { get }

   open var geometry: ARPlaneGeometry { get }

   ...

}
open class ARPlaneAnchor : ARAnchor {

open var transform: simd_float4x4 { get }

open var center: simd_float3 { get }
open var extent: simd_float3 { get }

open var geometry: ARPlaneGeometry { get }
...
open class ARPlaneAnchor : ARAnchor {

    open var transform: simd_float4x4 { get }

    open var center: simd_float3 { get }
    open var extent: simd_float3 { get }

    open var geometry: ARPlaneGeometry { get }
    ...
}

open class ARPlaneAnchor : ARAnchor {

    open var transform: simd_float4x4 { get }

    open var center: simd_float3 { get }
    open var extent: simd_float3 { get }

    open var geometry: ARPlaneGeometry { get }

    ...
}

Saving and Loading Maps

Michele Stoppa, ARKit Engineering
Saving and Loading World Map
Saving and Loading World Map

Acquire a good World Map
Saving and Loading World Map

Acquire a good World Map

Share the World Map
Saving and Loading World Map

Acquire a good World Map
Share the World Map
Relocalize to World Map
World Map Data

Internal tracking data
  • Map of 3D feature points
  • Local appearance
World Map Data

Internal tracking data
  • Map of 3D feature points
  • Local appearance
List of named anchors
World Map Data

Internal tracking data
• Map of 3D feature points
• Local appearance
List of named anchors
Serializable
Acquiring a Good World Map
Acquiring a Good World Map
Acquiring a Good World Map
Acquiring a Good World Map

Dense feature points on map
Acquiring a Good World Map

Dense feature points on map

Static environment
Acquiring a Good World Map

Dense feature points on map
Static environment
Multiple points of view
Acquiring a Good World Map

Dense feature points on map
Static environment
Multiple points of view
World mapping status
Acquiring a Good World Map
Guide the user
Acquiring a Good World Map

Guide the user

Indicate mapping status
Acquiring a Good World Map

Guide the user

Indicate mapping status
Acquiring a Good World Map

Guide the user

Indicate mapping status

Warn about limited tracking state

Tracking was limited for a long time...
You may want to restart
Acquiring a Good World Map

Guide the user

Indicate mapping status

Warn about limited tracking state

Guide relocalization

Find this view to start!
// Retrieve world map from ARSession
session.getCurrentWorldMap { worldMap, error in
    guard let worldMap = worldMap else {
        showAlert(error)
        return
    }
    // Serialize
    let data = try NSKeyedArchiver.archivedData(withRootObject: worldMap,
                                          requiringSecureCoding: true)
}
Share the World Map

```
// Retrieve world map from ARSession
session.getCurrentWorldMap { worldMap, error in
  guard let worldMap = worldMap else {
    showAlert(error)
    return
  }

  // Serialize
  let data = try NSKeyedArchiver.archivedData(withRootObject: worldMap,
    requiringSecureCoding: true)
}```
Share the World Map

// Retrieve world map from ARSession
session.getCurrentWorldMap { worldMap, error in
    guard let worldMap = worldMap else {
        showAlert(error)
        return
    }
}

// Serialize
let data = try NSKeyedArchiver.archivedData(withRootObject: worldMap,
    requiringSecureCoding: true)
Share the World Map

// Retrieve world map from ARSession
session.getCurrentWorldMap { worldMap, error in
    guard let worldMap = worldMap else {
        showAlert(error)
        return
    }
    // Serialize
    let data = try NSKeyedArchiver.archivedData(withRootObject: worldMap,
                                                requiringSecureCoding: true)
}

For shared experiences—MultipeerConnectivity framework
See sample “Creating a Multiuser AR Experience”
Relocalize to World Map

Setup configuration

```swift
// Receive / Load world map
let worldMap : ARWorldMap = ...

// Create a session configuration
let configuration = ARWorldTrackingConfiguration()
configuration.initialWorldMap = worldMap

// Run the session
session.run(configuration)
```
Relocalize to World Map

Setup configuration

```swift
// Receive / Load world map
let worldMap: ARWorldMap = ...

// Create a session configuration
let configuration = ARWorldTrackingConfiguration()
configuration.initialWorldMap = worldMap

// Run the session
session.run(configuration)
```
Relocalize to World Map

Setup configuration

```swift
// Receive / Load world map
let worldMap : ARWorldMap = ...

// Create a session configuration
let configuration = ARWorldTrackingConfiguration()
configuration.initialWorldMap = worldMap

// Run the session
session.run(configuration)
```
Relocalize to World Map

Setup configuration

```swift
// Receive / Load world map
let worldMap : ARWorldMap = ...

// Create a session configuration
let configuration = ARWorldTrackingConfiguration()
configuration.initialWorldMap = worldMap

// Run the session
session.run(configuration)
```
Relocalize to World Map

Before Relocalization
Relocalize to World Map

Before Relocalization

Tracking state is Limited with reason Relocalizing
Relocalize to World Map
Before Relocalization

Tracking state is Limited
with reason Relocalizing

World origin is first camera
Relocalize to World Map

After Relocalization
Relocalize to World Map
After Relocalization

Tracking state is Normal

World origin is initial world map
Re localize to World Map

After Relocalization

Tracking state is Normal

World origin is initial world map

Only minor changes in environment allowed
Image Tracking
Behind the Scenes
Localize the image
Behind the Scenes

Localize the image
Behind the Scenes
Localize the image
Behind the Scenes

Localize the image
Behind the Scenes

Localize the image
Behind the Scenes
Localize the image
Behind the Scenes

Dense tracking
Behind the Scenes

Dense tracking
Behind the Scenes

Dense tracking
Behind the Scenes
Dense tracking
Behind the Scenes

Dense tracking
Behind the Scenes
Dense tracking
Image Tracking API

Reference Images

ARRReferenceImage
Image Tracking API

Reference Images
- ARReferenceImage

Configurations
- ARWorldTrackingConfiguration
- ARImageTrackingConfiguration

Getting Results
- ARSession
- ARFrame
- ARImageAnchor
Adding Images as Assets

Create AR Resource Group
Adding Images as Assets

Create AR Resource Group

Drag images to be detected
Adding Images as Assets

Create AR Resource Group

Drag images to be detected

Set physical dimension for images
Setting Physical Dimension for Images

Physical image size must be known
Setting Physical Dimension for Images

Physical image size must be known

Allows content to be in physical dimension
Setting Physical Dimension for Images

Physical image size must be known

Allows content to be in physical dimension

Consistent with world tracking data
Good Images to Track

1. Vincent van Gogh's painting of a field of wheat and cypresses.
2. Yosemite National Park's El Capitan rock formation with a sunset sky.
Good Images to Track

- High texture
- High local contrast
- Well distributed histogram
- No repetitive structures
Good Images to Track

- High texture
- High local contrast
- Well distributed histogram
- No repetitive structures
Good Images to Track

High texture
High local contrast
Well distributed histogram
No repetitive structures

![Image with high texture and local contrast](image1.png)
![Image with repetitive structures](image2.png)
Reference Image Quality in Xcode
Each image is analyzed
Reference Image Quality in Xcode

Each image is analyzed

- **AR reference image “ZMagazinePage”:** The histogram of the image is narrow or not well distributed. Recognition works better on images with wider, flatter histograms.
- **AR reference image “ZMagazinePage”:** The image has repetitive structures. Recognition works better on images without repetitive structures.
Reference Image Quality in Xcode

Similar images

AR Resource Group

AR Reference Image

Name: SierraC
Size: 1 (Width: 0.49159, Height: 0.49159)
Units: Meters

AR reference image "SierraC" is too similar to "SierraA".
Reference Image Quality—Tips

Crop image to its core content
Reference Image Quality—Tips

Crop image to its core content
Reference Image Quality—Tips

Use multiple AR Resource Groups

Allow many more images to be detected
Reference Image Quality—Tips

Use multiple AR Resource Groups

Allow many more images to be detected

Max 25 images per group recommended
Reference Image Quality—Tips

Use multiple AR Resource Groups

Allow many more images to be detected

Max 25 images per group recommended

Switch between groups programmatically
Image Tracking Configurations

ARImageTrackingConfiguration

ARWorldTrackingConfiguration
Image Tracking Configurations

ARImageTrackingConfiguration

ARWorldTrackingConfiguration

Standalone image tracking
Image Tracking Configurations

ARIImageTrackingConfiguration

ARWorldTrackingConfiguration

Standalone image tracking

No world origin
Image Tracking Configurations

- ARImageTrackingConfiguration
  - Standalone image tracking
  - No world origin

- ARWorldTrackingConfiguration
  - Image and World tracking
Image Tracking Configurations

ARImageTrackingConfiguration
Standalone image tracking
No world origin

ARWorldTrackingConfiguration
Image and World tracking
Scene understanding available
Image Tracking Configurations

ARIImageTrackingConfiguration  ARWorldTrackingConfiguration
Image Tracking Configurations

ARIImageTrackingConfiguration

ARWorldTrackingConfiguration
Image Tracking Configurations

ARIImageTrackingConfiguration  ARWorldTrackingConfiguration
Image Tracking Configurations

ARImageTrackingConfiguration

ARWorldTrackingConfiguration
// Load Images from Assets
let imageSet = ARReferenceImage.referenceImages(inGroupNamed: "Room1", bundle: Bundle.main)
guard let imageSet = imageSet else {
    print("Error loading images")
    return
}

// Create a session configuration
let configuration = ARI mageTrackingConfiguration()
configuration.trackingImages = imageSet

// Run the session
let session = ARSession()
session.run(configuration)
// Load Images from Assets
let imageSet = ARReferenceImage.referenceImages(inGroupNamed: "Room1", bundle: Bundle.main)
guard let imageSet = imageSet else {
    print("Error loading images")
    return
}

// Create a session configuration
let configuration = ARI mageTrackingConfiguration()
configuration.trackingImages = imageSet

// Run the session
let session = ARSession()
session.run(configuration)
// Load Images from Assets
let imageSet = ARReferenceImage.referenceImages(inGroupNamed: "Room1", bundle: Bundle.main)
guard let imageSet = imageSet else {
    print("Error loading images")
    return
}

// Create a session configuration
let configuration = ARI mageTrackingConfiguration()
configuration.trackingImages = imageSet

// Run the session
let session = ARSession()
session.run(configuration)
// Load Images from Assets
let imageSet = ARReferenceImage.referenceImages(inGroupNamed: "Room1", bundle: Bundle.main)
guard let imageSet = imageSet else {
    print("Error loading images")
    return
}

// Create a session configuration
let configuration = ARImageTrackingConfiguration()
configuration.trackingImages = imageSet

// Run the session
let session = ARSession()
session.run(configuration)
func session(_ session: ARSession, didUpdate anchors: [ARAnchor]) {
    for anchor in anchors {
        if let imageAnchor = anchor as? ARImageAnchor {...}
    }
}
func session(_ session: ARSession, didUpdate anchors: [ARAnchor]) {
    for anchor in anchors {
        if let imageAnchor = anchor as? ARImageAnchor {...}
    }
}
func session(_ session: ARSession, didUpdate anchors: [ARAnchor]) {
    for anchor in anchors {
        if let imageAnchor = anchor as? ARImageAnchor {...}
    }
}
Getting Results

```swift
func session(_ session: ARSession, didUpdate anchors: [ARAnchor]) {
    for anchor in anchors {
        if let imageAnchor = anchor as? ARImageAnchor {...}
    }
}

open class ARImageAnchor : ARAnchor, ARTrackable {
    open var transform: simd_float4x4 { get }
    open var referenceImage: ARReferenceImage { get }
    public var isTracked: Bool { get }
}
```
Absolute Coordinate Space for Shared Experiences
Absolute Coordinate Space at Precise Location
Trail Head
Object Detection
Object Detection

Scanning

Detection
How to Acquire an Object?

Similar representation as a world map
How to Acquire an Object?

Similar representation as a world map

Use “Scanning and Detecting 3D Objects”
How to Acquire an Object?

Similar representation as a world map

Use “Scanning and Detecting 3D Objects”

Detection Quality affected by Scanning Quality
Tap 'Next' to create an approximate bounding box around the object which you want to scan.
Scan the object
Scan the object from all sides that you are interested in. Do not move the object while scanning! When done scanning, tap 'Stop'.
Adjust origin

Touch one of the spheres and pan to move the origin along the selected axis, or rotate and move with two fingers. When satisfied, tap 'Test'.

You can load a *usd: 3D model overlay by tapping the 'Load Model' button.
Adjust origin

Touch one of the spheres and pan to move the origin along the selected axis, or rotate and move with two fingers. When satisfied, tap 'Test'.

You can load a *.usd: 3D model overlay by tapping the 'Load Model' button.
Scanning & detecting might not work.
ARKit is initializing

Point at the scanned object
Attempting to detect the scanned object.
Consider moving the object to different environments and testing there.
Point at the scanned object
Attempting to detect the scanned object.
Consider moving the object to different environments and testing there.
Point at the scanned object
Attempting to detect the scanned object.
Consider moving the object to different environments and testing there.
Point at the scanned object
Attempting to detect the scanned object.
Consider moving the object to different environments and testing there.
Object Scanning
Share ARReferenceObject
Good Objects to Track

Rigid objects
Texture rich
No Reflective
No Transparent
// Load Objects from Assets
let objects = ARReferenceObjects.referenceObjects(inGroupNamed: "Object", bundle: Bundle.main)
guard let objects = objects else {
    print("Error loading objects")
    return
}

// Create a session configuration
let configuration = ARWorldTrackingConfiguration()
configuration.detectionObjects = objects

// Run the session
let session = ARSession()
session.run(configuration)
// Load Objects from Assets
let objects = ARReferenceObjects.referenceObjects(inGroupNamed: "Object", bundle: Bundle.main)
guard let objects = objects else {
    print("Error loading objects")
    return
}

// Create a session configuration
let configuration = ARWorldTrackingConfiguration()
configuration.detectionObjects = objects

// Run the session
let session = ARSession()
session.run(configuration)
// Load Objects from Assets
let objects = ARReferenceObjects.referenceObjects(inGroupNamed: "Object", bundle: Bundle.main)
guard let objects = objects else {
    print("Error loading objects")
    return
}

// Create a session configuration
let configuration = ARWorldTrackingConfiguration()
configuration.detectionObjects = objects

// Run the session
let session = ARSession()
session.run(configuration)
Object Detection API

// Load Objects from Assets
let objects = ARReferenceObjects.referenceObjects(inGroupNamed: "Object", bundle: Bundle.main)
if let objects = objects {
    print("Error loading objects")
    return
}

// Create a session configuration
let configuration = ARWorldTrackingConfiguration()
configuration.detectionObjects = objects

// Run the session
let session = ARSession()
session.run(configuration)
func session(_ session: ARSession, didUpdate anchors: [ARAnchor]) {
    for anchor in anchors {
        if let objectAnchor = anchor as? ARObjectAnchor {
            let objectName = objectAnchor.referenceObject.name ?? ""
            print("Object found: \(objectName)")
        }
    }
}
func session(_ session: ARSession, didUpdate anchors: [ARAnchor]) {
    for anchor in anchors {
        if let objectAnchor = anchor as? ARObjectAnchor {
            let objectName = objectAnchor.referenceObject.name ?? ""
            print("Object found: \(objectName)")
        }
    }
}
func session(_ session: ARSession, didUpdate anchors: [ARAnchor]) {
    for anchor in anchors {
        if let objectAnchor = anchor as? ARObjectAnchor {
            let objectName = objectAnchor.referenceObject.name ?? ""
            print("Object found: \(objectName)")
        }
    }
}
func session(_ session: ARSession, didUpdate anchors: [ARAnchor]) {
    for anchor in anchors {
        if let objectAnchor = anchor as? ARObjectAnchor {
            let objectName = objectAnchor.referenceObject.name ?? ""
            print("Object found: \(objectName)")
        }
    }
}
Object Detection vs. World Map Relocalization

Object Detection

World Map Relocalization

Object

C

A

W
Object Detection vs. World Map Relocalization

Object Detection

World Map Relocalization
Object Detection vs. World Map Relocalization

Object Detection

Object Map Relocalization

Object2

Object
More Information

https://developer.apple.com/wwdc18/610