NAMRATA BANDEKAR

SO YOU WANT TO BUILD AN ARKIT APP

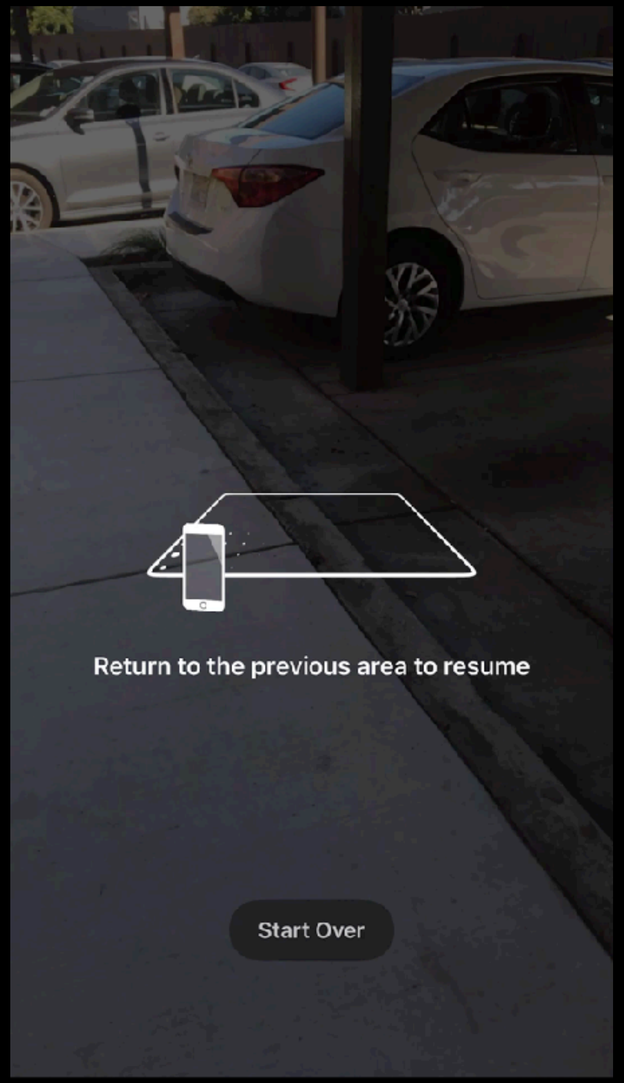
ARKIT BY TUTORIALS

AGENDA

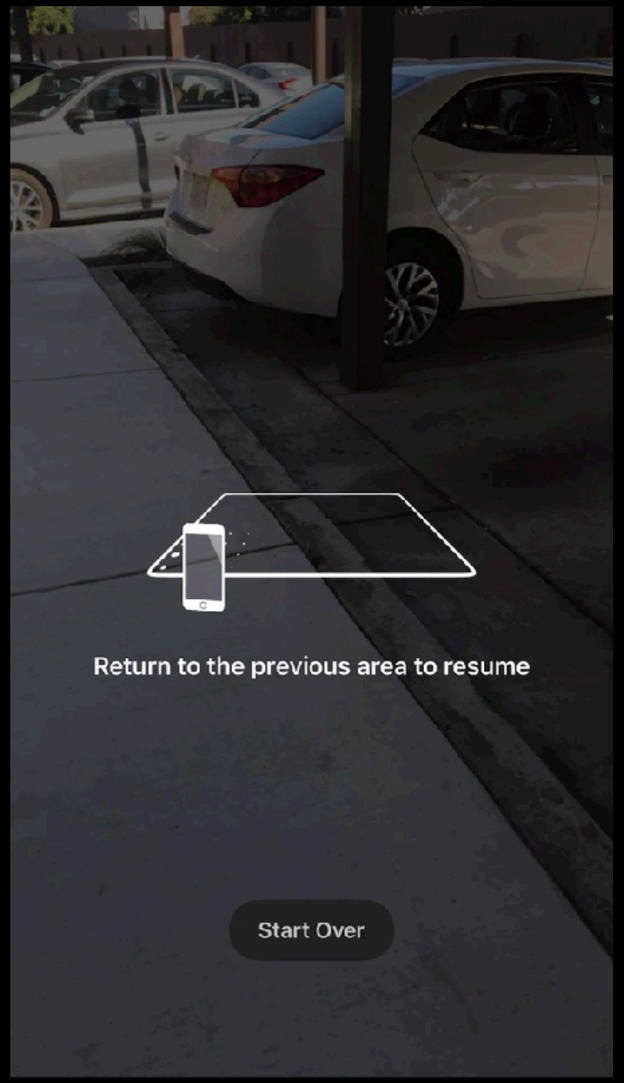
- Getting Started
- Diving Deeper
- Do’s and Don’ts

PORTAL

PORTAL

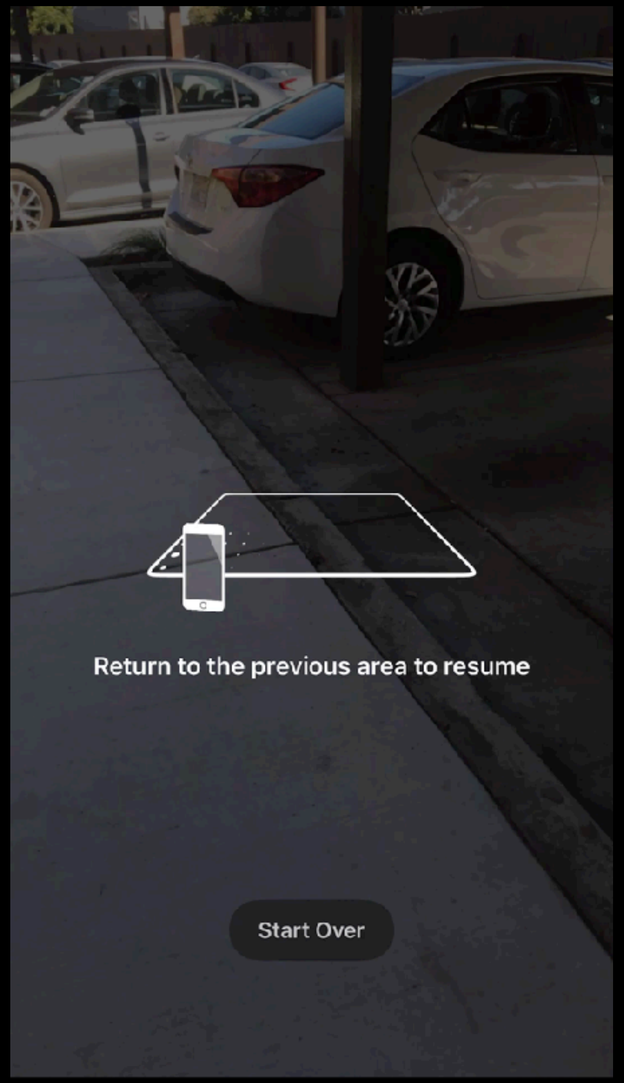


PORTAL

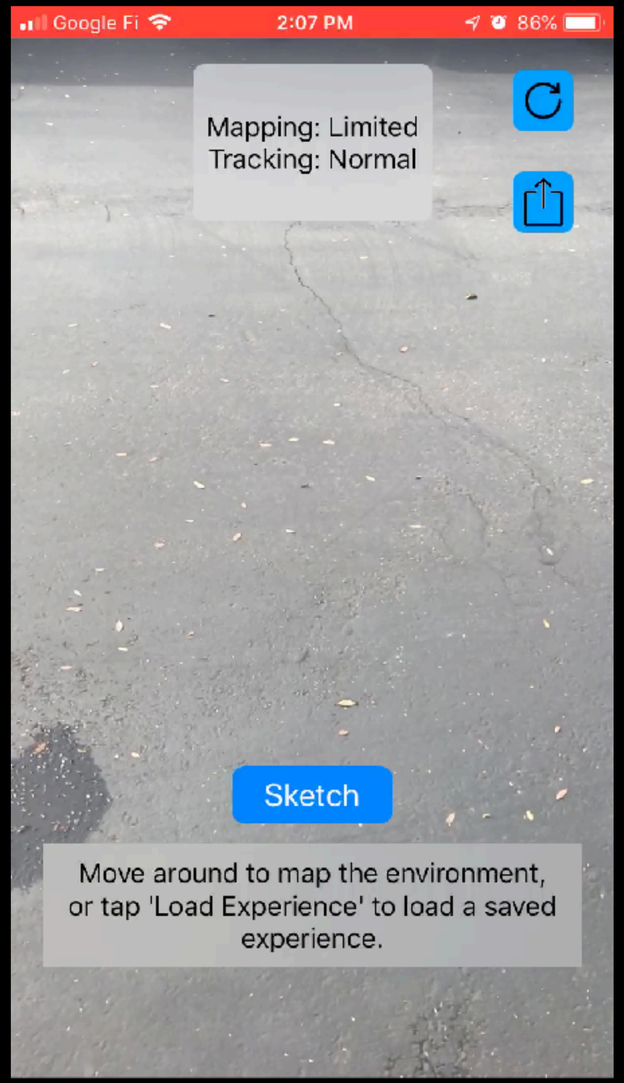


ARSKETCH

PORTAL



ARSKETCH



AUGMENTED REALITY EXPERIENCE

AUGMENTED REALITY EXPERIENCE

- Detect and understand environment geometry

AUGMENTED REALITY EXPERIENCE

- Detect and understand environment geometry
- Render virtual 3D content

AUGMENTED REALITY EXPERIENCE

- Detect and understand environment geometry
- Render virtual 3D content
- User guidance

GETTING STARTED

GETTING STARTED

- Device Compatibility

GETTING STARTED

- Device Compatibility
  - A9 processor (iPhone 6) or later

GETTING STARTED

- Device Compatibility
  - A9 processor (iPhone 6) or later
  - iOS 11 and up

GETTING STARTED

- Device Compatibility
  - A9 processor (iPhone 6) or later
  - iOS 11 and up
- Handle unsupported devices and iOS versions

GETTING STARTED

- Device Compatibility
  - A9 processor (iPhone 6) or later
  - iOS 11 and up
- Handle unsupported devices and iOS versions

```
<key>UILaunchStoryboardName</key>
<string>LaunchScreen</string>
<key>UIMainStoryboardFile</key>
<string>Main</string>
<key>UIRequiredDeviceCapabilities</key>
<array>
  <string>arkit</string>
</array>
```

AR Required

GETTING STARTED

- Device Compatibility
  - A9 processor (iPhone 6) or later
  - iOS 11 and up
- Handle unsupported devices and iOS versions

GETTING STARTED

- Device Compatibility
  - A9 processor (iPhone 6) or later
  - iOS 11 and up
- Handle unsupported devices and iOS versions

```
guard ARFaceTrackingConfiguration.isSupported else {
  updateMessage(text: "Face Tracking Not Supported.")
  return
}
```

AR Optional

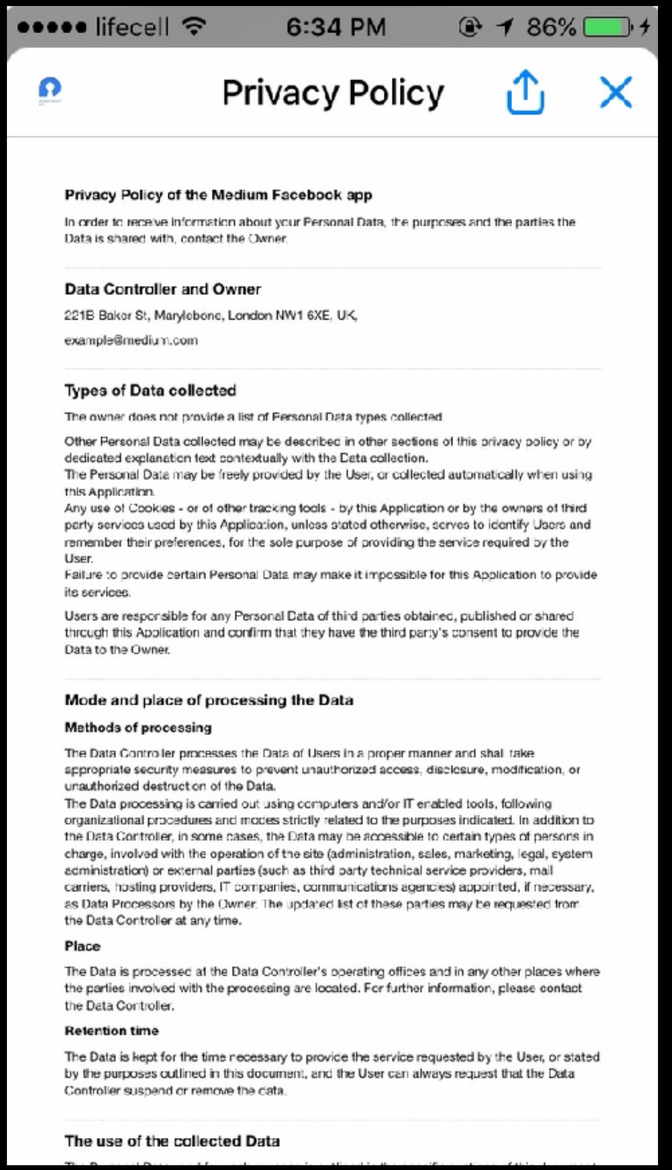
GETTING STARTED

- Device permissions

```
<key>LSRequiresIPhoneOS</key>
<true/>
<key>NSCameraUsageDescription</key>
<string>Portal needs to use your camera.</string>
<key>UILaunchStoryboardName</key>
<string>LaunchScreen</string>
```

GETTING STARTED

- Privacy and data handling



DIVING DEEPER

TRACKING

TRACKING

- ARAnchors fix position in real world ⚓

TRACKING

- ARAnchors fix position in real world ⚓
- ARAnchors adjust for drift

TRACKING

- ARAnchors fix position in real world ⚓
- ARAnchors adjust for drift
- Don’t use feature points to place content

TRACKING

- ARAnchors fix position in real world ⚓
- ARAnchors adjust for drift
- Don’t use feature points to place content

TRACKING QUALITY

TRACKING QUALITY

- Ambient light

TRACKING QUALITY

- Ambient light

TRACKING QUALITY

- Ambient light

TRACKING QUALITY

- Ambient light

TRACKING QUALITY

- Ambient light

TRACKING QUALITY

- Ambient light
- Static environments

TRACKING QUALITY

- Ambient light
- Static environments

TRACKING QUALITY

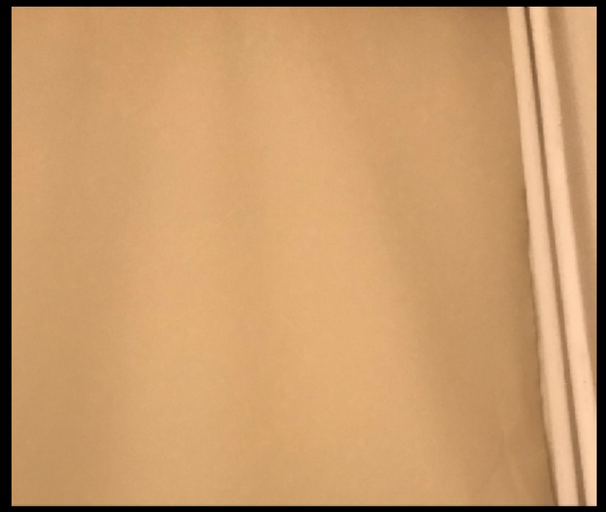
- Ambient light
- Static environments

TRACKING QUALITY

- Ambient light
- Static environments
- Surface texture

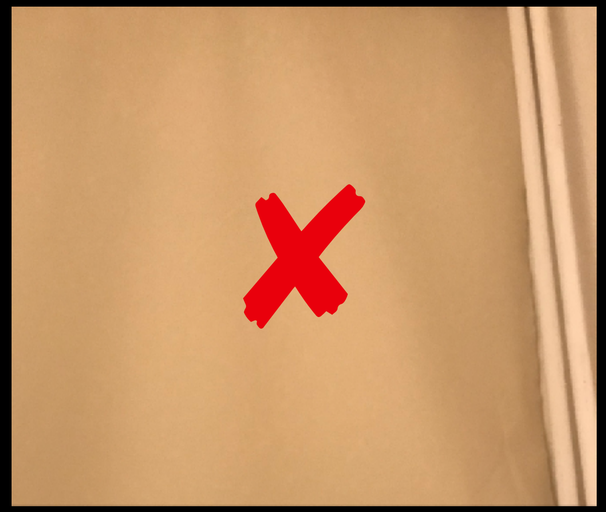
TRACKING QUALITY

- Ambient light
- Static environments
- Surface texture



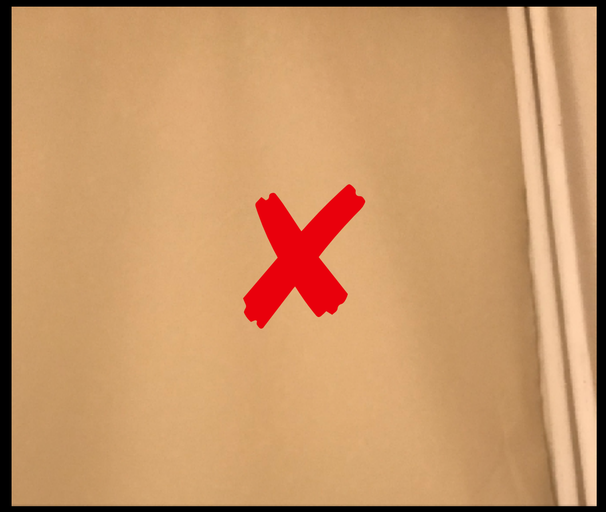
TRACKING QUALITY

- Ambient light
- Static environments
- Surface texture



TRACKING QUALITY

- Ambient light
- Static environments
- Surface texture
- Sensor data quality



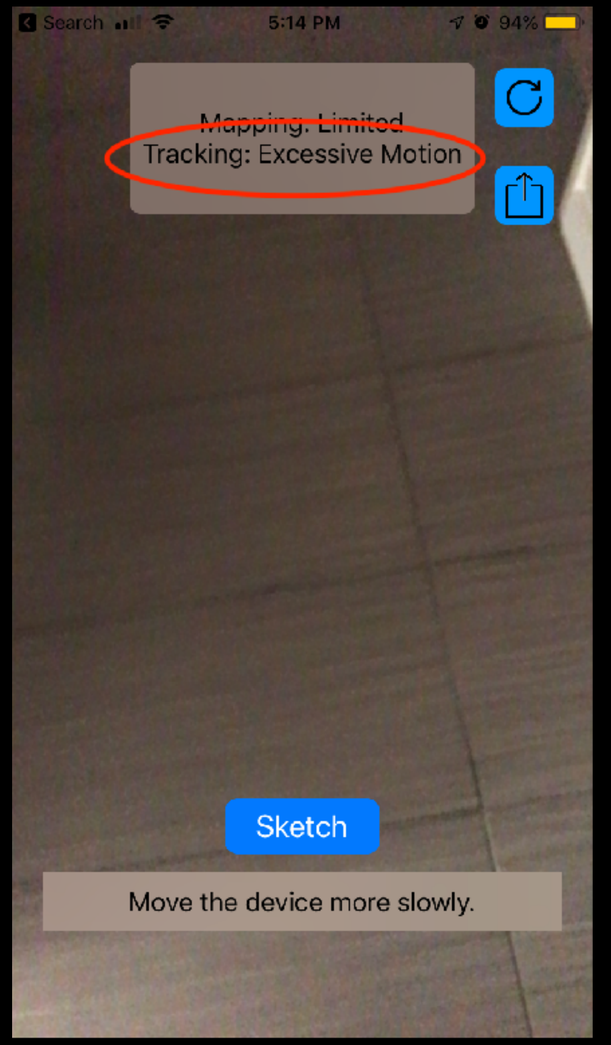
TRACKING QUALITY

TRACKING QUALITY

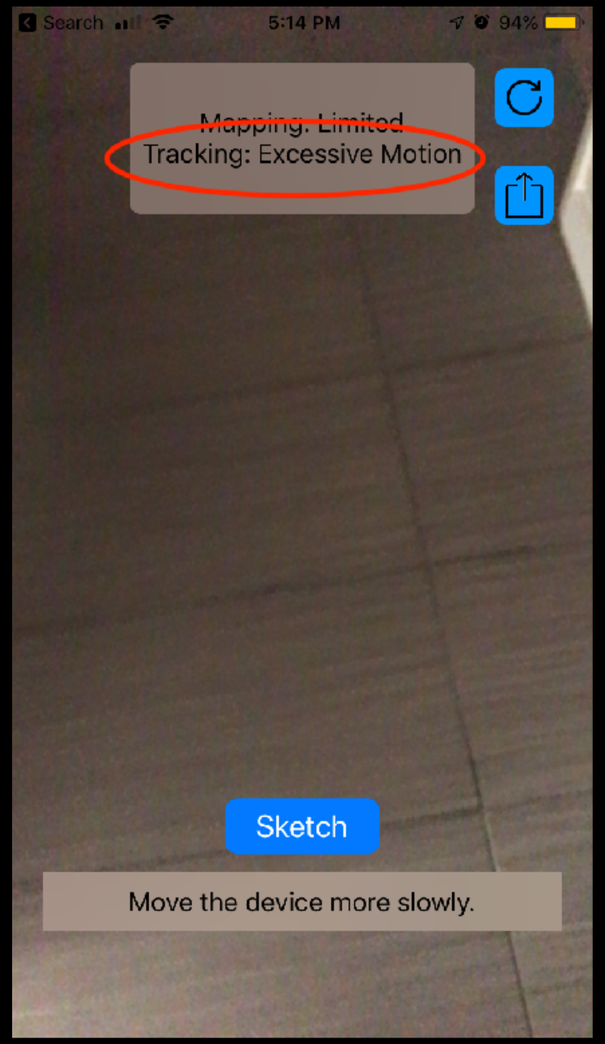
```
ARWorldTrackingConfiguration

open class ARCamera : NSObject, NSCopying {

    open var transform: simd_float4x4 { get }

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

    …
}
```

TRACKING QUALITY

```
ARWorldTrackingConfiguration

open class ARCamera : NSObject, NSCopying {

    open var transform: simd_float4x4 { get }

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

    …
}
```

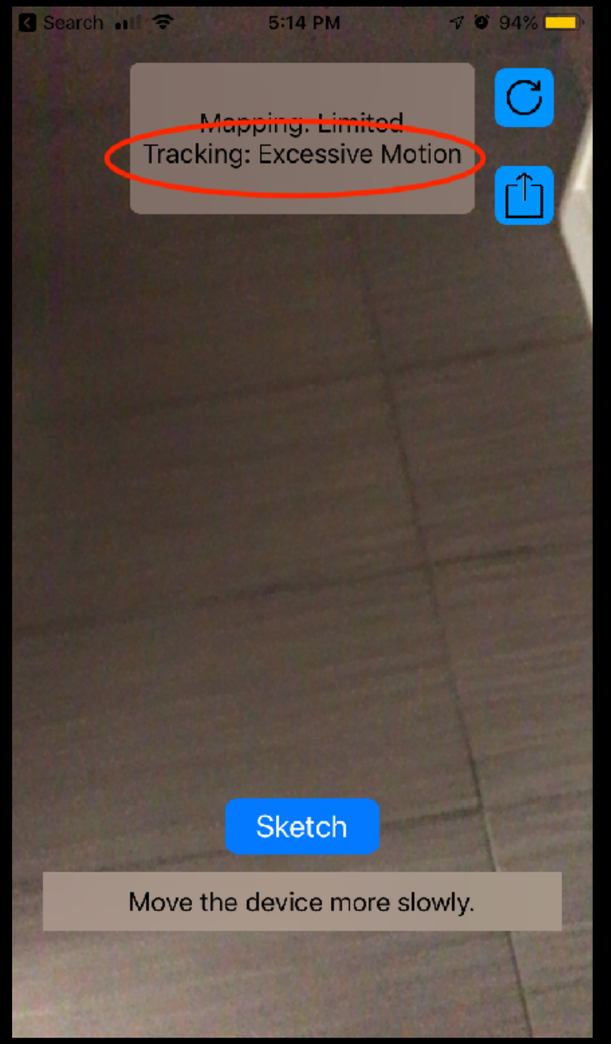


TRACKING QUALITY



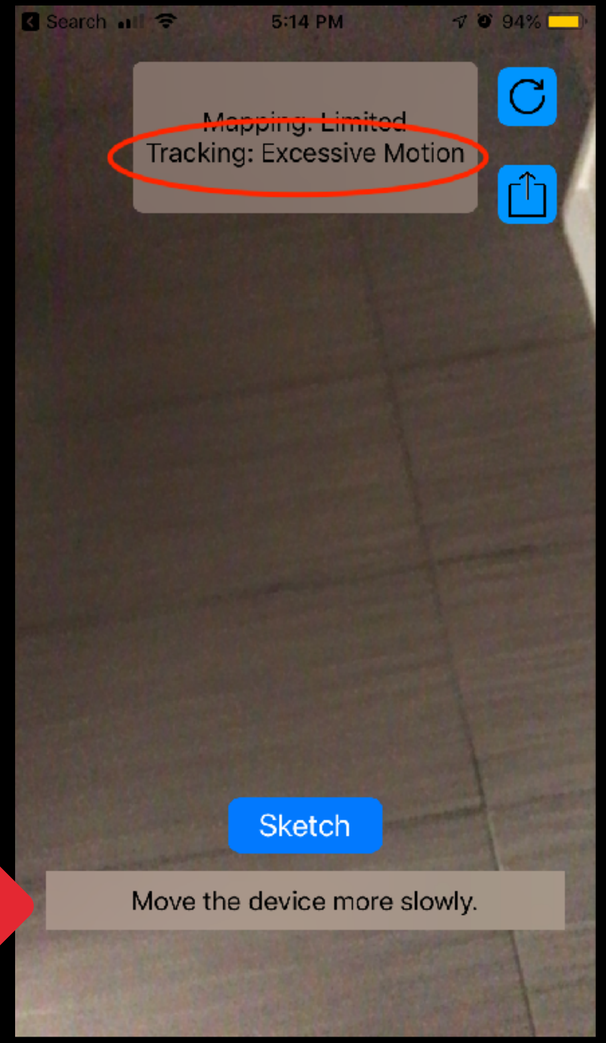
TRACKING QUALITY

```
extension ARCamera.TrackingState {
  var localizedFeedback: String {
    switch self {
    case .normal:
      return "All good!"
    case .notAvailable:
      return "Tracking unavailable."
    case .limited(.excessiveMotion):
      return "Move the device more slowly."
    case .limited(.insufficientFeatures):
      return "Point the device at an area with visible surface detail."
    case .limited(.relocalizing):
      return "Resuming session — move to where you were when the session
    case .limited(.initializing):
      return "Initializing AR session."
    }
  }
}
```



TRACKING QUALITY

```
extension ARCamera.TrackingState {
  var localizedFeedback: String {
    switch self {
    case .normal:
      return "All good!"
    case .notAvailable:
      return "Tracking unavailable."
    case .limited(.excessiveMotion):
      return "Move the device more slowly."
    case .limited(.insufficientFeatures):
      return "Point the device at an area with visible surface detail."
    case .limited(.relocalizing):
      return "Resuming session — move to where you were when the session
    case .limited(.initializing):
      return "Initializing AR session."
    }
  }
}
```



PERSISTING VIRTUAL CONTENT

PERSISTING VIRTUAL CONTENT

- ARWorldMap - serializable

PERSISTING VIRTUAL CONTENT

- ARWorldMap - serializable
- Use NSSecureCoding protocol

PERSISTING VIRTUAL CONTENT

- ARWorldMap - serializable
- Use NSSecureCoding protocol
- ARAnchors store object state and position

CAPTURING THE ARWORLDMAP

CAPTURING THE ARWORLDMAP

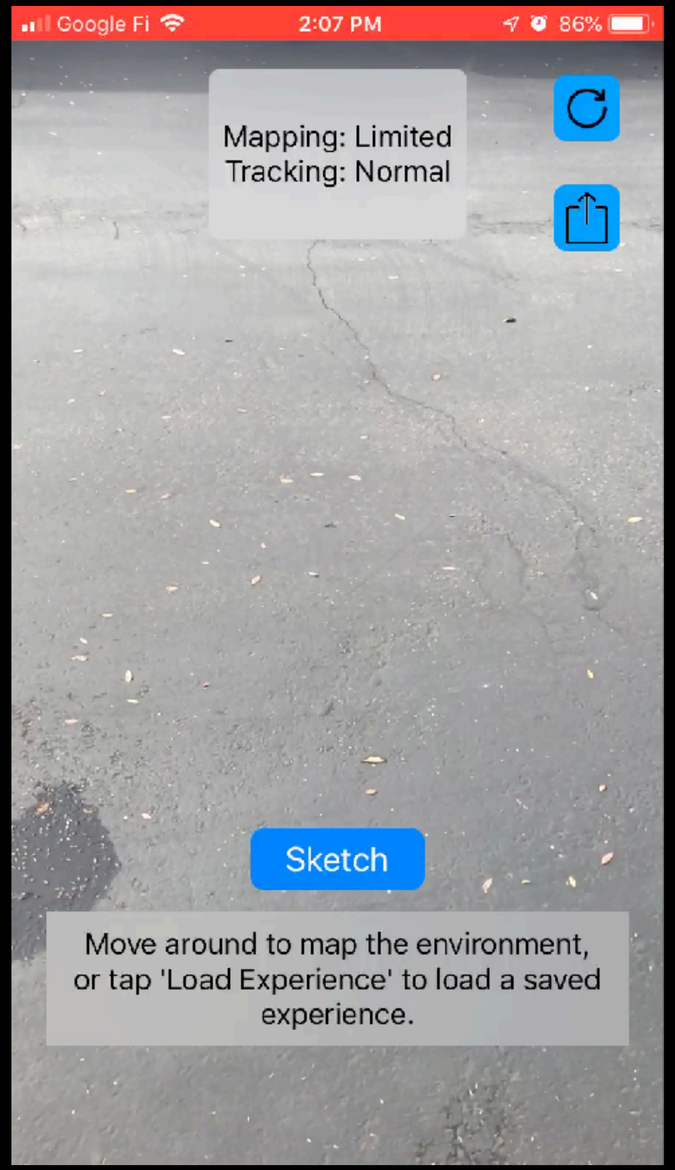
```
extension ARFrame.WorldMappingStatus: CustomStringConvertible {
  public var description: String {
    switch self {
    case .notAvailable:
      return "Not Available"
    case .limited:
      return "Limited"
    case .extending:
      return "Extending"
    case .mapped:
      return "Mapped"
    }
  }
}
```

CAPTURING THE ARWORLDMAP

- Use WorldMappingStatus

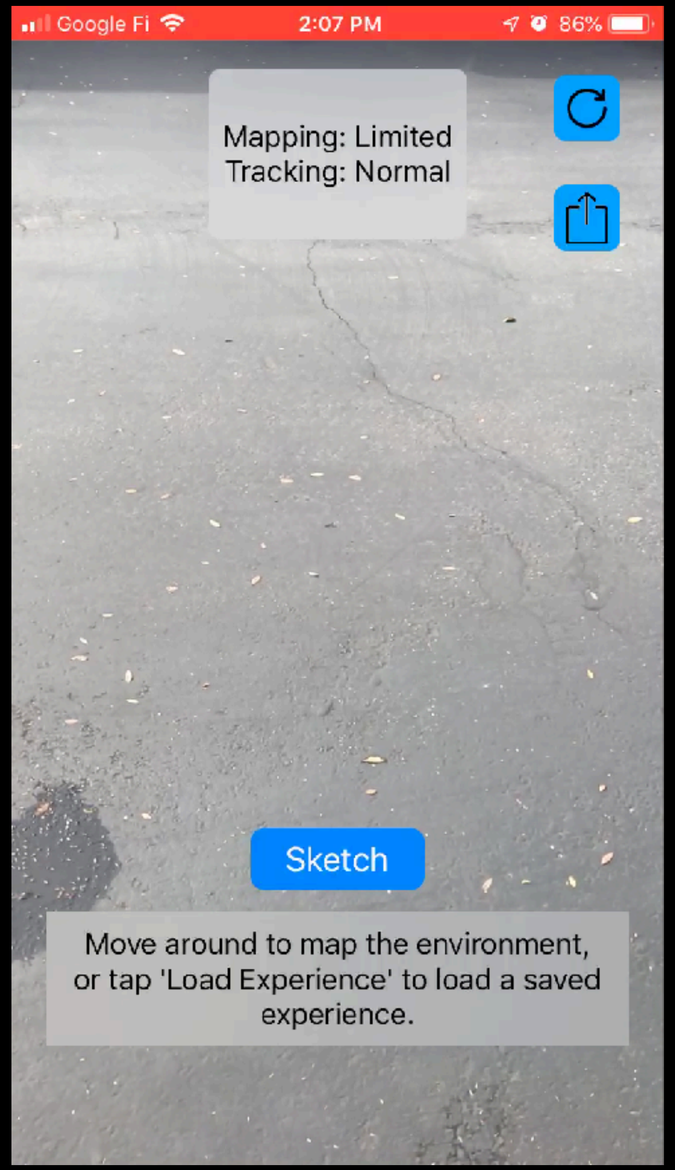
```
extension ARFrame.WorldMappingStatus: CustomStringConvertible {
  public var description: String {
    switch self {
    case .notAvailable:
      return "Not Available"
    case .limited:
      return "Limited"
    case .extending:
      return "Extending"
    case .mapped:
      return "Mapped"
    }
  }
}
```

WORLD MAPPING



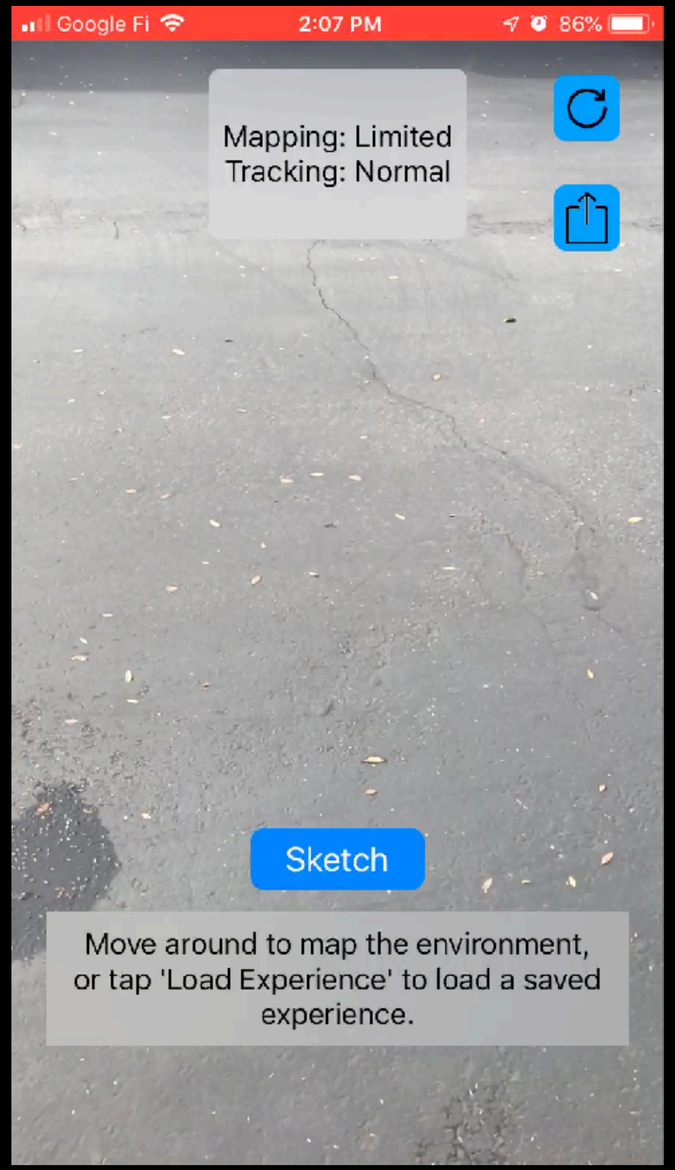
WORLD MAPPING

- Don’t display UI until ready



WORLD MAPPING

- Don’t display UI until ready



DO'S ✅
AND DON'TS ❌

USE THE ENTIRE SCREEN ✅

COMFORTABLE AND SAFE TECHNOLOGY

KISS - Keep it short and safe

ENABLE ONE-HANDED APP USE

ENABLE ONE-HANDED APP USE

INITIALIZATION TAKES TIME ⏳

Do ✅	Don’t ❌
Try moving around, turning on more lights and making sure your environment has sufficiently textured surfaces	Surface detection is taking too long
Move your phone slower	Excessive motion detected
Tap a location to place the [name of object to be placed]	Tap a plane to anchor an object

GHOST EFFECT 👻

GHOST EFFECT 👻

DON'T ASSUME IDEAL ENVIRONMENT

MAPPING FROM A DISTANCE


Google Fi
2:09 PM
86%
Mapping: Extending
Tracking: Initializing
Sketch
Initializing AR session.


Google Fi
2:09 PM
86%
Mapping: Extending
Tracking: Initializing
Sketch
Initializing AR session.

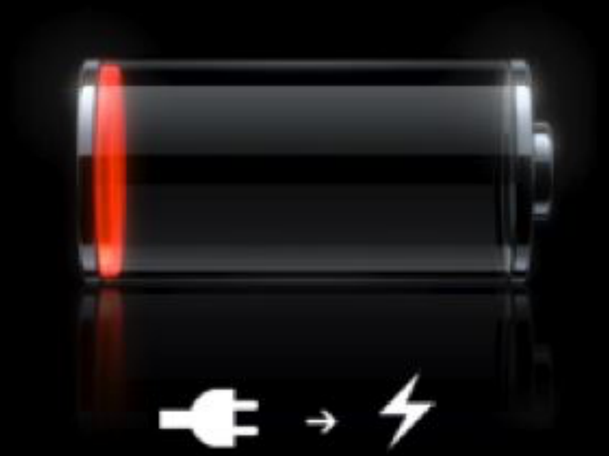
SCENEKIT DELEGATES

```
extension ViewController : ARSCNViewDelegate {
  func renderer(_ renderer: SCNSceneRenderer, updateAtTime time: TimeInterval) {
    DispatchQueue.main.async {
      self.statusLabel.text = self.trackingStatus
    }
  }

  func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor) {
    guard let planeAnchor = anchor as? ARPlaneAnchor else { return }
    DispatchQueue.main.async {
      let planeNode = self.createARPlaneNode(planeAnchor: planeAnchor,
                                              color: UIColor.yellow.withAlphaComponent(0.5))
      node.addChildNode(planeNode)
    }
  }

  func renderer(_ renderer: SCNSceneRenderer, didUpdate node: SCNNode, for anchor: ARAnchor) {
    guard let planeAnchor = anchor as? ARPlaneAnchor else { return }
    DispatchQueue.main.async {
      self.updateARPlaneNode(planeNode: node.childNodes[0], planeAchor: planeAnchor)
    }
  }
}
```

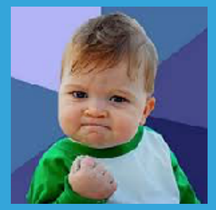
SCENEKIT DELEGATES

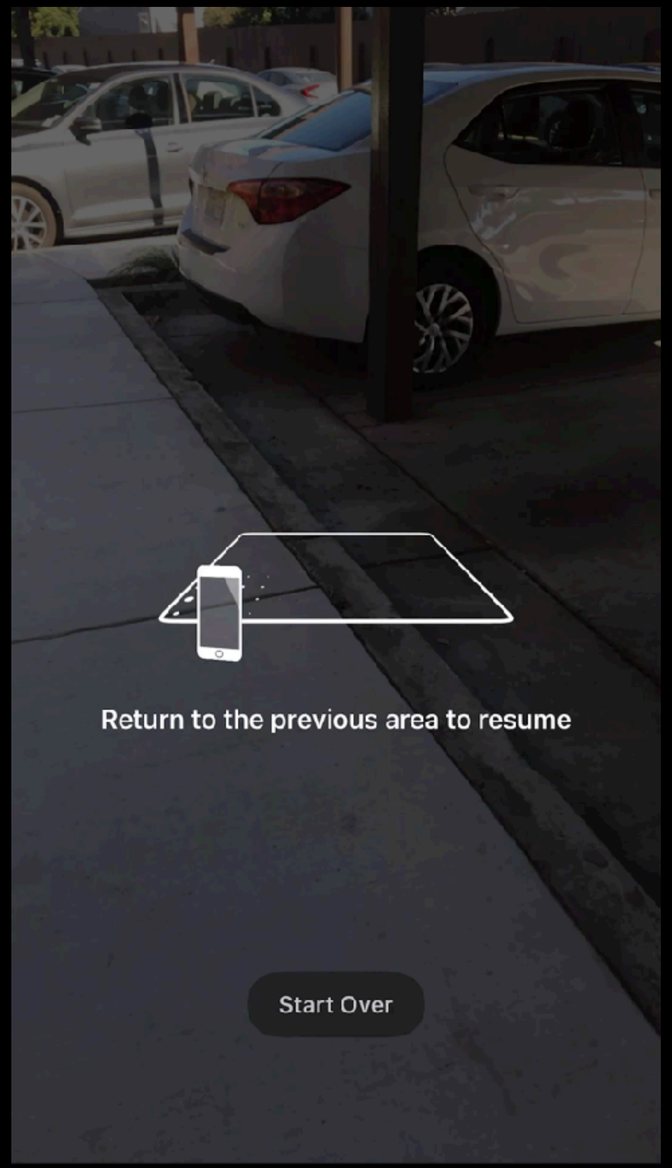
```
extension ViewController : ARSCNViewDelegate {
  func renderer(_ renderer: SCNSceneRenderer, updateAtTime time: TimeInterval) {
    DispatchQueue.main.async {
      self.statusLabel.text = self.trackingStatus
    }
  }

  func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor) {
    guard let planeAnchor = anchor as? ARPlaneAnchor else { return }
    DispatchQueue.main.async {
      let planeNode = self.createARPlaneNode(planeAnchor: planeAnchor,
                                             color: UIColor.yellow.withAlphaComponent(0.5))

      node.addChildNode(planeNode)
    }
  }

  func renderer(_ renderer: SCNSceneRenderer, didUpdate node: SCNNode, for anchor: ARAnchor) {
    guard let planeAnchor = anchor as? ARPlaneAnchor else { return }
    DispatchQueue.main.async {
      self.updateARPlaneNode(planeNode: node.childNodes[0], planeAchor: planeAnchor)
    }
  }
}
```

BATTERY DRAIN! 😖

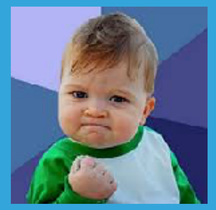


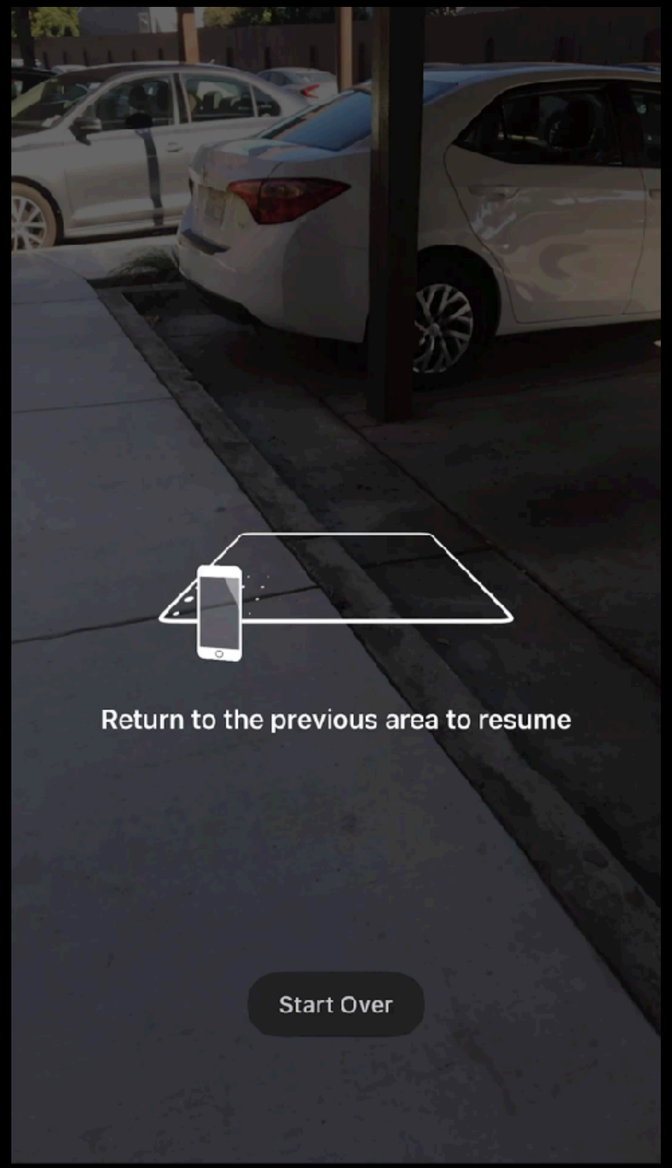
SET YOUR USERS UP FOR SUCCESS





SET YOUR USERS UP FOR SUCCESS





THANK YOU!

 @NAMRATACODES

 NAMRATABANDEKAR

https://tinyurl.com/swiftfest-arkit