WWDC 2017

Prepared for CocoaHeads by
Jeff Meador
June 29, 2017

WWDC 2017

Overview

01 ARKit History

02 ARKit Demo Breakdown

03 CoreML

04 Vision

05 NLP

06 Demo

ARKit Origin Story

Pre-ARKit AR

2014 Launch of BMW i8

- Need to do 3D recognition on a car
- Overlay animated 3D models on car
- Must run on Google Glass

2014 Launch of BMW i8

Metaio

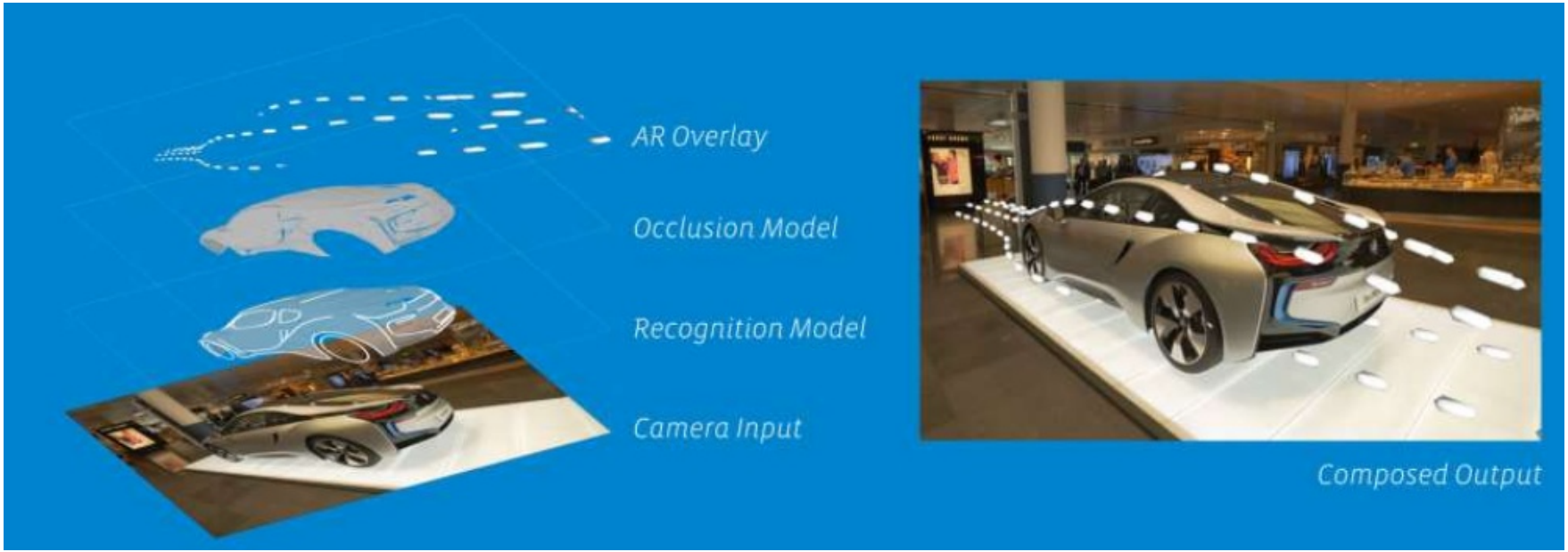
Pioneers in Augmented Reality
and Computer Vision

www.metaio.com

Metaio SDK Features

- SLAM + 3D Tracking
- **3D Markerless Tracking based on CAD data**
- 2D Tracking
- Face Tracking
- Works on Mobile, **Wearables**, Desktop
- **Has its own renderer**
- Cloud recognition for image search
- **Runs offline**
- Unity Plugin and **Native Support**

Using Metaio



2015 – Sale to Apple

TECHNOLOGY NEWS | Fri May 29, 2015 | 1:47pm EDT

Apple buys German augmented-reality software maker Metaio

Vuforia

Because Metaio Doesn't Exist Anymore

www.vuforia.com

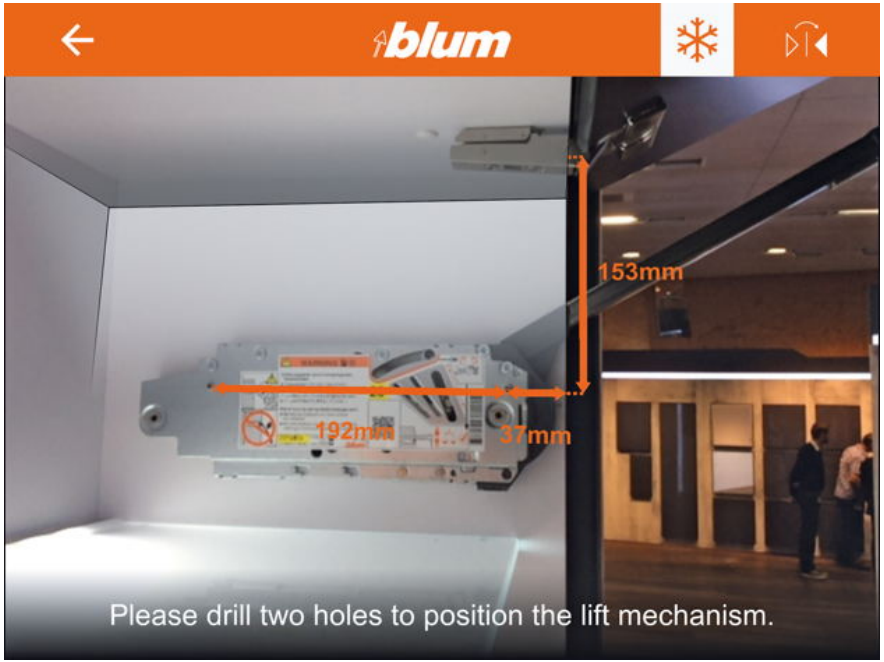
Vuforia Differences

- No 3D Markerless Tracking based on CAD data (in beta now)
- Limited 3D Markerless tracking
  - Scan small objects
  - Boxes
  - Cylindars
- No Renderer

Vuforia Object Scanner

Vuforia


blum
Please use an electric screwdriver to adjust the lift mechanism. Front should be in balance.


blum
153mm
192mm
37mm
Please drill two holes to position the lift mechanism.

ARKit

Metaio is Back!

ARKit Features Compared to Metaio

- SLAM ~~+ 3D Tracking~~
- ~~3D Markerless Tracking based on CAD data~~
- ~~2D Tracking~~
- ~~Face Tracking~~
- Works on ~~Mobile~~ (iOS only), ~~Wearables, Desktop~~
- Has its own renderer
- ~~Cloud recognition for image search~~
- Runs offline
- Unity Plugin and Native Support (and Unreal)

ARKit Potential Features

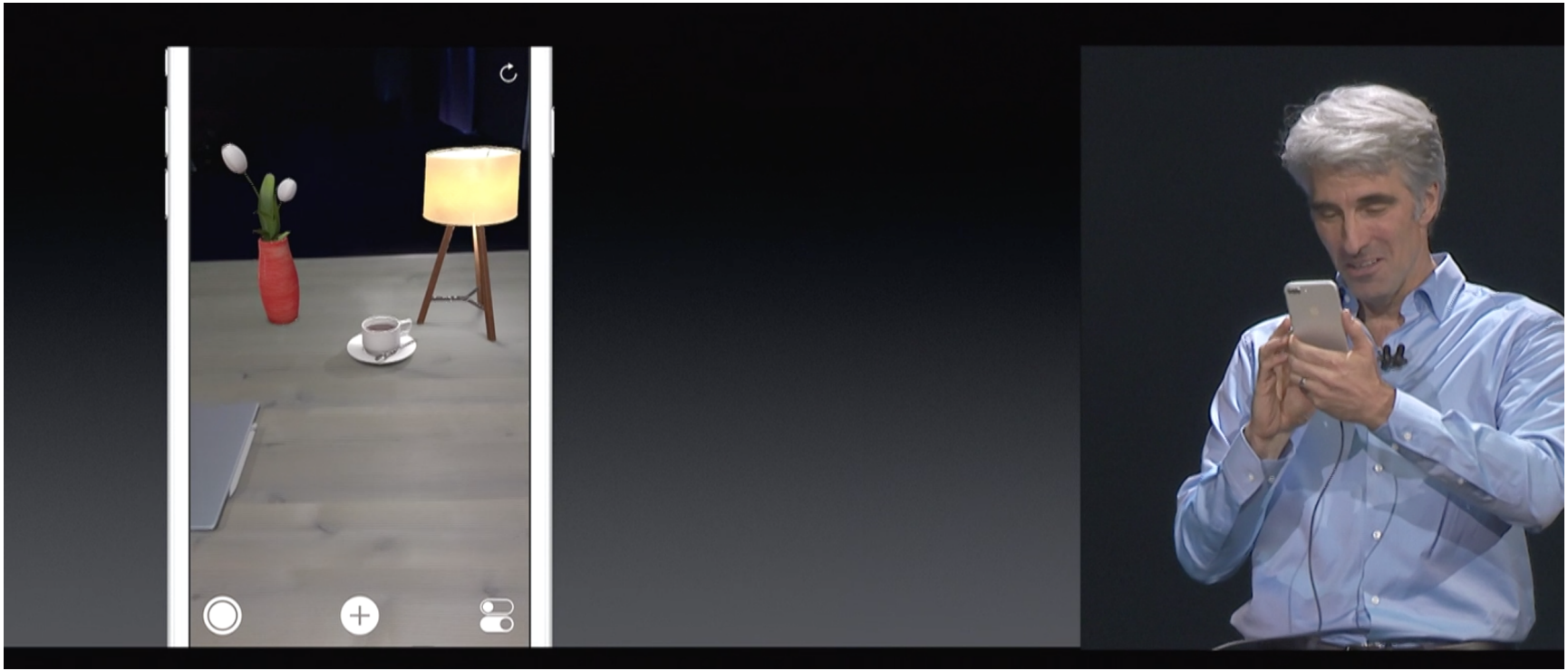
Either Custom Implementation, or Future iOS API from Apple

- SLAM + 3D Tracking
- 3D Markerless Tracking based on CAD data
- 2D Tracking
- Face Tracking
- Works on ~~Mobile~~ (iOS only), Wearables~~, Desktop~~
- Has its own renderer
- Cloud recognition for image search
- Runs offline
- Unity Plugin and Native Support (and Unreal)

ARKit Demo Breakdown

The Code Behind

ARKit Demo Breakdown



https://developer.apple.com/sample-code/wwdc/2017/PlacingObjects.zip

ARSCNView & ARSession

ARSCNView (3D, SceneKit) or ARSKView (2D, SpriteKit). Both are UIView subclasses.

ARSession

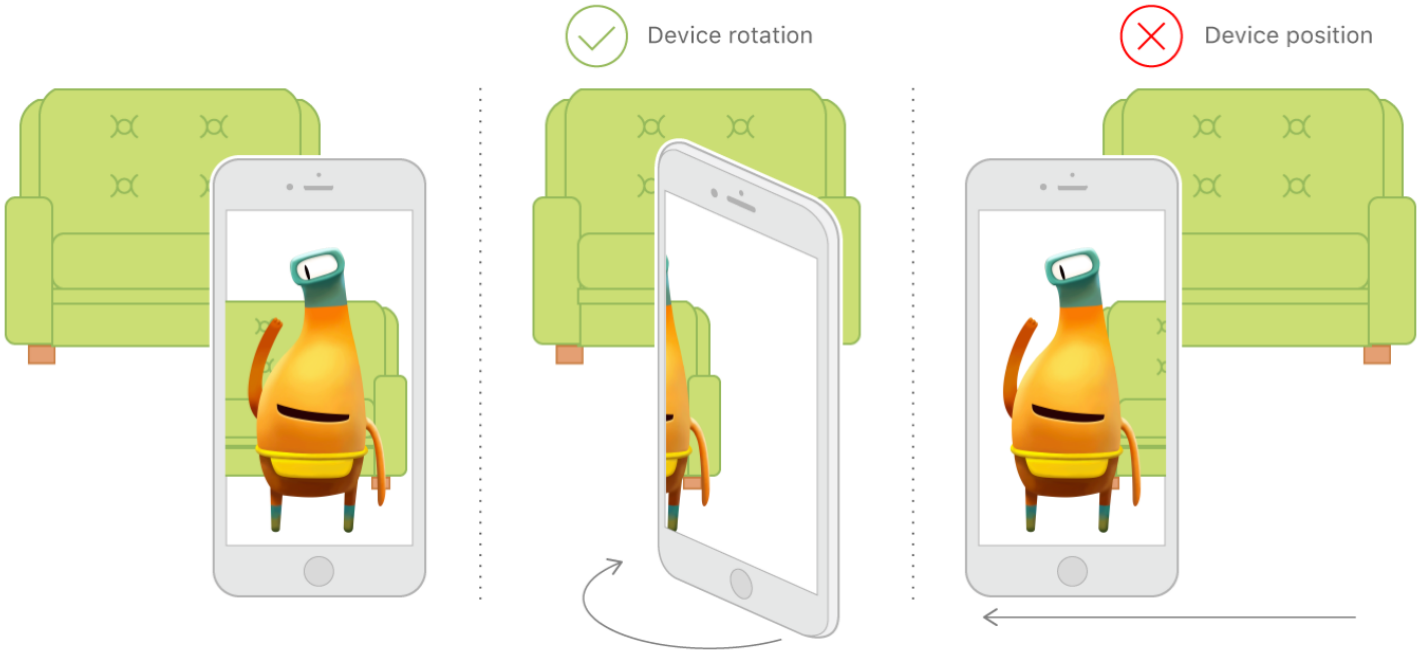
A shared object that manages the device camera and motion processing needed for augmented reality experiences.

Session Configuration

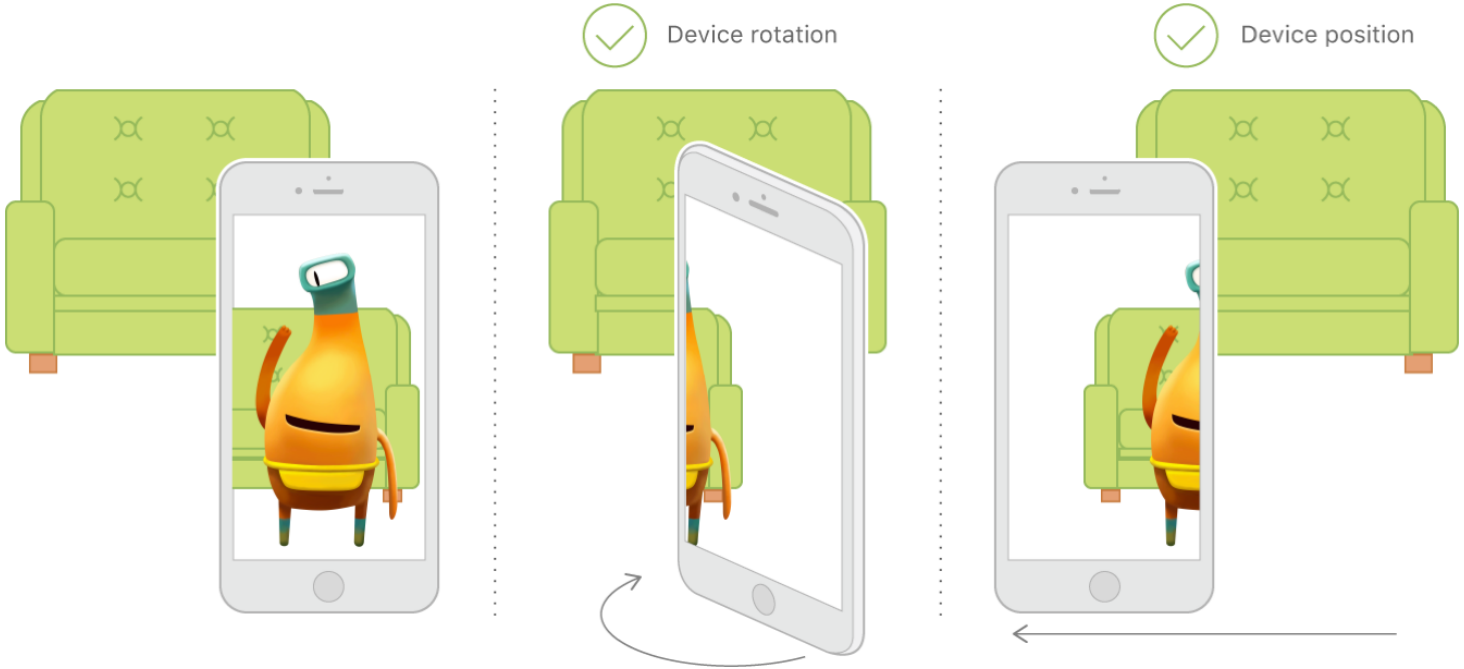
ARSessionConfiguration

ARWorldTrackingSessionConfiguration
(inherits from ARSessionConfiguration)

ARSessionConfiguration



ARWorldTrackingSessionConfiguration



Session Configuration

Supported	RAM	Processor	iOS devices			Highest supported iOS
			iPhone	iPod touch	iPad	
Yes	4GB	A9X			Pro 12.9" 1G	10.3.2 (Current)
	3GB	A10 Fusion	7 Plus			
			7			
		A9X			Pro 9.7"	
	2GB	A9	6s (Plus) / SE		5	
		A8X			Air 2	
		A8			mini 4	
			6 (Plus)	6		
		A7	5s		Air / mini 2, 3	
No	1GB	A6X			4	10.3.2
		A6	5 / 5c			
		A5X			3	9.3.5
	512MB	A5	4S	5	2 / mini 1G	
		A4	4			7.1.2
	256MB		3GS			6.1.6
		A4		4		
					1G	5.1.1
				3		
	128MB		3G	2		4.2.1
			Original or 2G	1		3.1.3

https://en.wikipedia.org/wiki/List_of_iOS_devices

ARWorldTrackingSessionConfiguration

ARSessionConfiguration

ARSessionConfiguration

Options

```
open class ARSessionConfiguration : NSObject, NSCopying {


    /**
     Determines whether this device supports the ARSessionConfiguration.
     */
    open class var isSupported: Bool { get }


    /**
     Determines how the session's coordinate system should be aligned with the world.
     @discussion The default is ARWorldAlignmentGravity.
     */
    open var worldAlignment: ARSessionConfiguration.WorldAlignment


    /**
     Enable or disable light estimation.
     @discussion Enabled by default.
     */
    open var isLightEstimationEnabled: Bool
}
```

ARSessionConfiguration

Options

```
extension ARSessionConfiguration {


    /**
     Enum constants for indicating the world alignment.
     */
    @available(iOS 11.0, *)
    public enum WorldAlignment : Int {


        /** Aligns the world with gravity that is defined by vector (0, -1, 0). */
        case gravity


        /** Aligns the world with gravity that is defined by the vector (0, -1, 0)
         and heading (w.r.t. True North) that is given by the vector (0, 0, -1). */
        case gravityAndHeading


        /** Aligns the world with the camera's orientation. */
        case camera
    }
}
```

ARWorldTrackingSessionConfiguration

Options

```
/**
 A session configuration for world tracking.

 @discussion World tracking provides 6 degrees of freedom tracking of the device.
 By finding feature points in the scene, world tracking enables performing hit-tests against the frame.
 Tracking can no longer be resumed once the session is paused.
 */
@available(iOS 11.0, *)
open class ARWorldTrackingSessionConfiguration : ARSessionConfiguration {


    /**
     Type of planes to detect in the scene.
     @discussion If set, new planes will continue to be detected and updated over time. Detected planes will be added to the session as
     ARPlaneAnchor objects. In the event that two planes are merged, the newer plane will be removed. Defaults to ARPlaneDetectionNone.
     */
    open var planeDetection: ARWorldTrackingSessionConfiguration.PlaneDetection
}
```

ARWorldTrackingSessionConfiguration

Options

```
extension ARWorldTrackingSessionConfiguration {


    /**
     Option set indicating the type of planes to detect.
     */
    @available(iOS 11.0, *)
    public struct PlaneDetection : OptionSet {

        public init(rawValue: UInt)


        /** Plane detection determines horizontal planes in the scene. */
        public static var horizontal: ARWorldTrackingSessionConfiguration.PlaneDetection { get }
    }
}
```

ARAnchor & SCNNode

Detecting Planes

ARAnchor

A real-world position and orientation that can be used for placing objects in an AR scene.

SCNNode

A structural element of a scene graph, representing a position and transform in a 3D coordinate space, to which you can attach geometry, lights, cameras, or other displayable content.

Delegate

Detecting Planes

```
// MARK: - ARSessionDelegate

func session(_ session: ARSession, didAdd anchors: [ARAnchor]) {
    for anchor in anchors {
        if let _ = anchor as? ARPlaneAnchor {
            print("found plane")
        }
    }
}

// MARK: - ARSCNViewDelegate

func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor) {
    DispatchQueue.main.async {
        if let planeAnchor = anchor as? ARPlaneAnchor {
            self.addPlane(node: node, anchor: planeAnchor)
            self.checkIfObjectShouldMoveOntoPlane(anchor: planeAnchor)
        }
    }
}
```

(lldb) po node

<SCNNode: 0x1c43eb700 pos(0.224395 -0.952981 -0.928924) rot(0.000000 1.000001 0.000000 0.562208) scale(1.000000 1.000000 1.000000) | no child>

(lldb) po anchor

<ARPlaneAnchor: 0x1034ee340 identifier="11D4B7B0-0100-0000-EE2B-484FFEFFFFFF" transform=<translation=(0.224395 -0.952981 -0.928924) rotation=(-0.00° 32.21° 0.00°)> alignment=horizontal center=(0.025461 0.000000 -0.148811) extent=(0.709298 0.000000 1.088733)>

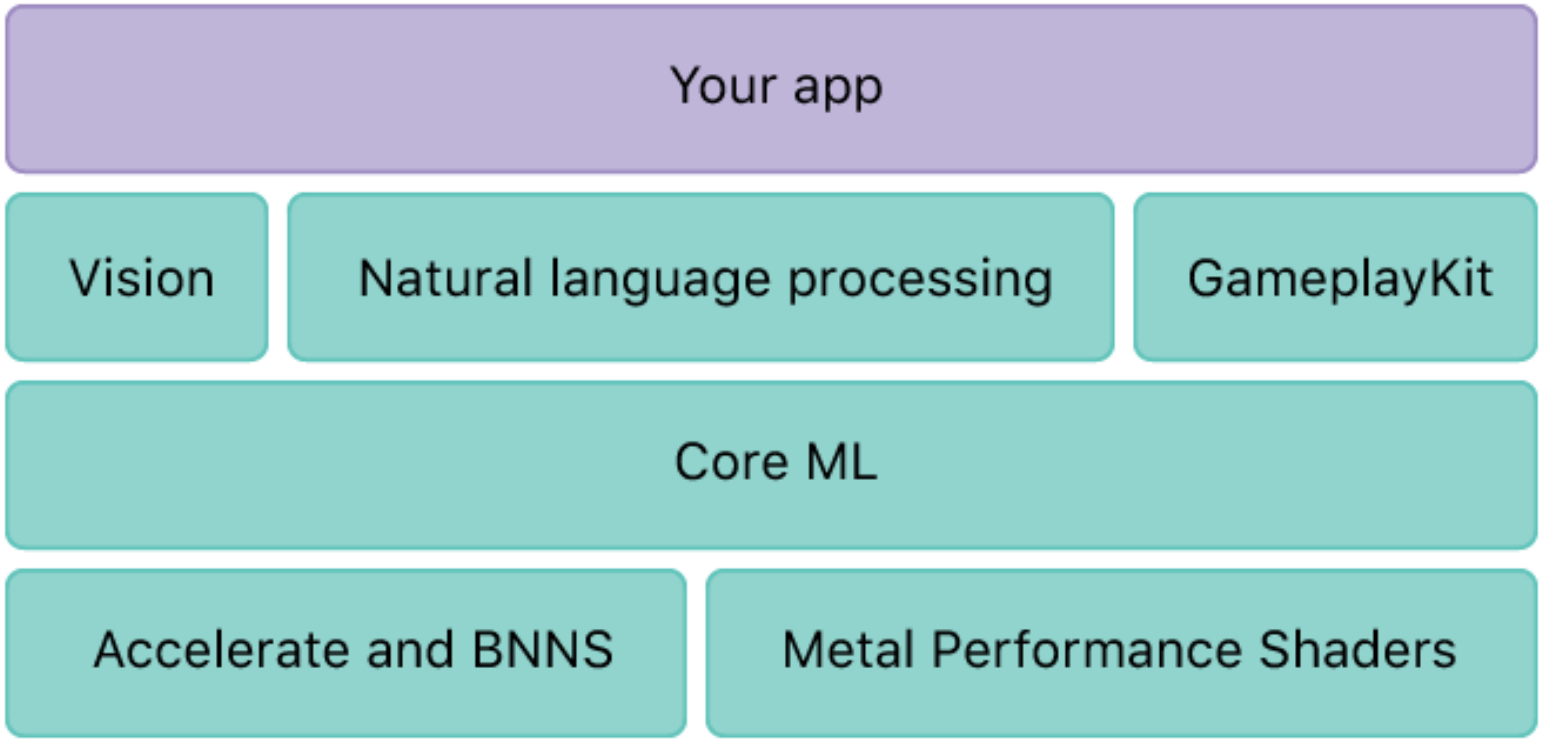
Adding Objects

1. Create SCNode for 3D Model
2. Perform Hit Test on ARSCNView (function provided). Will find planes.
3. Set position on SCNode using result from hit test
4. Add SCNNode to ARSCNView
5. ???
6. Profit

CoreML

What are you?

API


Your app
Vision
Natural language processing
GameplayKit
Core ML
Accelerate and BNNS
Metal Performance Shaders

API

Where's the API….?

Core ML API

Use the Core ML API directly to support custom workflows and advanced use cases.

Framework
Core ML

On This Page
Overview
Topics

Overview

In most cases, you interact only with your model's dynamically generated interface, which is created by Xcode automatically when you add a model to your Xcode project. You can use Core ML APIs directly in cases where you need to support custom workflows or advanced use cases. As an example, if you need to make predictions while asynchronously collecting input data into a custom structure, you can use that structure to provide input features to your model by adopting the `MLFeatureProvider` protocol.

Topics

Model

`class MLModel`
An encapsulation of all the details of your machine learning model.
Beta

Model Features

`protocol MLFeatureProvider`
An interface that represents a collection of feature values for a model.
Beta

`class MLDictionaryFeatureProvider`
A convenience wrapper for the given dictionary of data.
Beta

`class MLFeatureValue`
An immutable instance representing a feature's type and value.
Beta

`class MLFeatureDescription`
A description of a feature.
Beta

`class MLMultiArray`
A multidimensional array used as input or output for a model.
Beta

Errors

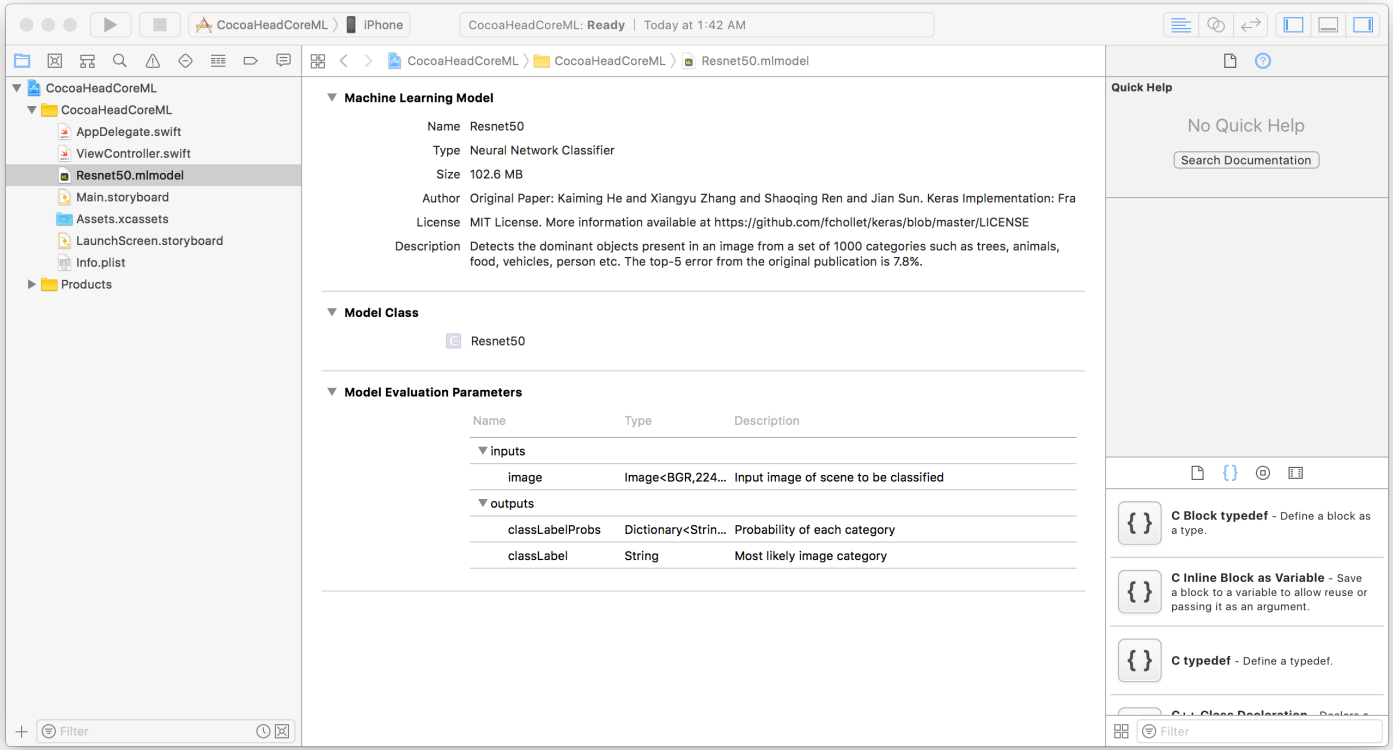
`struct MLModelError`
Error codes for Core ML.
Beta

API

Import these using Apple's tools
or create custom import tool

Model type	Supported models	Supported tools
Neural networks	Feedforward, convolutional, recurrent	Caffe Keras 1.2.2
Tree ensembles	Random forests, boosted trees, decision trees	scikit-learn 0.18 XGBoost 0.6
Support vector machines	Scalar regression, multiclass classification	scikit-learn 0.18 LIBSVM 3.22
Generalized linear models	Linear regression, logistic regression	scikit-learn 0.18
Feature engineering	Sparse vectorization, dense vectorization, categorical processing	scikit-learn 0.18
Pipeline models	Sequentially chained models	scikit-learn 0.18

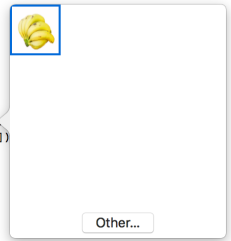
Demo



https://developer.apple.com/machine-learning/

Demo

```
5  class ViewController: UIViewController {
6
7      override func viewDidLoad() {
8          super.viewDidLoad()
9
10         guard let model = try? VNCoreMLModel(for: Resnet50().model) else {
11             return
12         }
13         let request = VNCoreMLRequest(model: model, completionHandler: { request, error in
14             if let results = request.results as? [VNClassificationObservation] {
15                 for result in results {
16                     print(result.identifier, result.confidence)
17                 }
18             }
19         })
20
21
22
23
24         let handler = VNImageRequestHandler(cgImage:
25                                             options: [:])
26
27         do {
28             try handler.perform([request])
29         } catch {
30
31         }
32     }
33
34     override func didReceiveMemoryWarning() {
35         super.didReceiveMemoryWarning()
36         // Dispose of any resources that can be recreated.
37     }
38
39
40 }
```



CocoaHeadCoreML

```
banana 0.992965
zucchini, courgette 0.00301567
orange 0.000891429
lemon 0.000589199
knot 0.00040495
acorn squash 0.00036158
cucumber, cuke 0.00023899
spaghetti squash 0.000161078
Granny Smith 9.96257e-05
coil, spiral, volute, whorl, helix 8.52144e-05
pineapple, ananas 8.06793e-05
custard apple 7.75886e-05
butternut squash 4.9126e-05
grocery store, grocery, food market, market 3.9014e-05
snail 3.76662e-05
meat loaf, meatloaf 3.6082e-05
artichoke, globe artichoke 2.8655e-05
strainer 2.53869e-05
broccoli 2.27566e-05
```

4.0

Vision

ARKit Features

- Face Detection & Recognition
  - Face Rectangle Request
  - Face Landmark Request
- Machine Learning Image Analysis
  - Core ML Request

ARKit Features

- Barcode Detection
  - Detect Barcode Request
- Image Alignment Analysis
  - Translational Image Registration Request
    - Determine affine transform to align content of 2 images
  - Homographic Image Registration Request
    - Determine perspective warp matrix needed to align content of 2 images

ARKit Features

- Text Detection
  - Detect Text Rectangles
- Horizon Detection
  - Detect Horizon Request
- Object Detection and Tracking
  - Detect Rectangles
  - Track Rectangles

5.0

NLP

Class

NSLinguisticTagger

Analyze natural language to tag part of speech and lexical class, identify proper names, perform lemmatization, and determine the language and script (orthography) of text.

Overview

The `NSLinguisticTagger` class provides a uniform interface to a variety of natural language processing functionality with support for many different languages and scripts. You can use `NSLinguisticTagger` to segment natural language text into paragraphs, sentences, or words, and tag information about those tokens, such as part of speech, lexical class, lemma, script, and language.

When you create a linguistic tagger, you specify what kind of information you're interested in by passing one or more `NSLinguisticTagScheme` values. Set the `string` property to the natural language text you want to analyze, and the linguistic tagger processes it according to the specified tag schemes. You can then enumerate over the tags in a specified range, using the methods described in Enumerating Linguistic Tags, to get the information requested for a given scheme and unit.

SDKs

iOS 5.0+

macOS 10.7+

tvOS 9.0+

watchOS 2.0+

Framework

Foundation

On This Page

Demo

ARFrame

ARFrame

A video image and position tracking information captured as part of an AR session.

SDK
iOS 11.0+ Beta

Framework
ARKit

On This Page

Overview

A running AR session continuously captures video frames from the device camera. For each frame, ARKit analyzes the image together with data from the device's motion sensing hardware to estimate the device's real-world position. ARKit delivers this tracking information and imaging parameters in the form of an `ARFrame` object.

Topics

Accessing Captured Video Frames

`capturedImage`
A pixel buffer containing the image captured by the camera.

`timestamp`
The time at which the frame was captured.

Examining Scene Parameters

`camera`
Information about the camera position, orientation, and imaging parameters used to capture the frame.

`lightEstimate`
An estimate of lighting conditions based on the camera image.

– `displayTransformWithViewportSize:orientation:`
Returns an affine transform for converting between normalized image coordinates and a coordinate space appropriate for rendering the camera image onscreen.

Tracking and Finding Objects

`anchors`
The list of anchors representing positions tracked or objects detected in the scene.

– `hitTest:types:`
Searches for real-world objects or AR anchors in the captured camera image.

Debugging Scene Detection

`rawFeaturePoints`
The current intermediate results of the scene analysis ARKit uses to perform world tracking.

`ARPointCloud`
A collection of points in the world coordinate space of the AR session.

Thank You

Vectorform  Invent with us.