Capturing Depth in iPhone Photography

Session 507

Brad Ford, Purveyor of Deep Thoughts
Depth and disparity on iPhone 7 Plus
Streaming depth data from the camera
Capturing photos with depth data
Dual photo capture
Depth and disparity on iPhone 7 Plus

Streaming depth data from the camera

Capturing photos with depth data

Dual photo capture
Depth and disparity on iPhone 7 Plus
Streaming depth data from the camera
Capturing photos with depth data
Dual photo capture
Depth and disparity on iPhone 7 Plus
Streaming depth data from the camera
Capturing photos with depth data
Dual photo capture
Depth and disparity on iPhone 7 Plus
Streaming depth data from the camera
Capturing photos with depth data
Dual photo capture
Dual Camera Zoom

Switches between wide and tele automatically
Matches exposure, focus, and frame rate
Compensates for parallax shift to smooth the transition
Portrait Mode
Deep Learning
Depth Map

Pinhole Camera

Image Plane (Sensor)

Observed Objects
Depth Map

Pinhole Camera

Image Plane
(Sensor)
Depth Map

Pinhole Camera

Focal Point

Image Plane (Sensor)

Focal Length
Depth Map

Time of Flight Camera
Depth Map

Time of Flight Camera
Disparity
Disparity Map

Camera 1

Camera 2
Disparity Map

Camera 1

Camera 2

Focal Length
Disparity Map

Camera 1
Optical Center

Camera 2
Optical Center

Focal Length
Disparity Map

- Observed Point
- Baseline
- "Z" (Depth)
- Focal Length
- Camera 1
  - Optical Center
- Camera 2
  - Optical Center
Disparity Map

Camera 1
Optical Center

Camera 2
Optical Center

Baseline

Observed Point

"Z" (Depth)

Focal Length
Disparity Map

- Observed Point
- Camera 1 (Optical Center)
- Camera 2 (Optical Center)
- Baseline
- "Z" (Depth)
- Focal Length
Disparity Map

Camera 1
Optical Center

Baseline

Camera 2
Optical Center

Observed Point

"Z" (Depth)

Focal Length
Disparity Map

Camera 1
Optical Center
Baseline

Camera 2
Optical Center
Baseline

Observed Point

"Z" (Depth)

Focal Length
Disparity Map

Camera 1

Camera 2
Disparity Map

Camera 1

Camera 2

Disparity
Removing Despair from Disparity

Observed Point

Camera 1

Camera 2
Removing Despair from Disparity

Observed Point

Camera 1

Camera 2
Removing Despair from Disparity

Observed Point

Camera 1

Camera 2
Removing Despair from Disparity
Removing Despair from Disparity

- Focal length (pixels)
- Disparity (pixels)
- Baseline (meters)
- \( Z \) (meters)
Removing Despair from Disparity

- Disparity (pixels)
- Focal length (pixels)
- Baseline (meters)
- Z (meters)
Removing Despair from Disparity

\[ \frac{b}{z} = \frac{d}{fl} \]
Removing Despair from Disparity

\[ \frac{b}{z} = \frac{d}{fl} \]
Removing Despair from Disparity

\[
\frac{b}{Z} = \frac{d}{fl}
\]
Removing Despair from Disparity

\[ \frac{b}{z} \frac{b}{d} = \frac{d}{fl} \frac{b}{b} \]
Removing Despair from Disparity

\[
\frac{b}{Z} = \frac{d}{f \cdot b}
\]
Removing Despair from Disparity

\[ \frac{1}{z} = \frac{d}{flb} \]
Normalized Disparity

\[
\frac{1}{z} = \frac{d}{f l b}
\]
Disparity vs. Depth

iPhone 7 Plus Dual Camera system is disparity based
Disparity vs. Depth

iPhone 7 Plus Dual Camera system is disparity based

Disparity is a proxy for depth
Disparity vs. Depth

iPhone 7 Plus Dual Camera system is disparity based

Disparity is a proxy for depth

Normalized disparity is the inverse of depth
A Deep Thought
New Term: Depth Data
Introducing AVDepthData

The canonical representation of depth on iOS, macOS, and tvOS

Class in the AVFoundation framework

Represents depth or disparity maps

Converts between depth/disparity formats
Introducing AVDepthData

```swift
public var kCVPixelFormatType_DisparityFloat16: OSTYPE { get } /* 'hdis' */

public var kCVPixelFormatType_DisparityFloat32: OSTYPE { get } /* 'hdis' */

public var kCVPixelFormatType_DepthFloat16: OSTYPE { get } /* 'hdep' */

public var kCVPixelFormatType_DepthFloat32: OSTYPE { get } /* 'fdep' */
```
Introducing AVDepthData

```swift
public var kCVPixelFormatType_DisparityFloat16: OSTYPE { get } /* 'hdis' */

public var kCVPixelFormatType_DisparityFloat32: OSTYPE { get } /* 'hdis' */

public var kCVPixelFormatType_DepthFloat16: OSTYPE { get } /* 'hdep' */

public var kCVPixelFormatType_DepthFloat32: OSTYPE { get } /* 'fdep' */
```
Introducing AVDepthData

```swift
public var kCVPixelFormatType_DisparityFloat16: OSTYPE { get } /* 'hdis' */

public var kCVPixelFormatType_DisparityFloat32: OSTYPE { get } /* 'hdis' */

public var kCVPixelFormatType_DepthFloat16: OSTYPE { get } /* 'hdep' */

public var kCVPixelFormatType_DepthFloat32: OSTYPE { get } /* 'fdep' */
```
Introducing AVDepthData

@available(iOS 11.0, *)
open class AVDepthData: NSObject {

    open var depthDataType: OSType { get }

    open var depthDataMap: CVPixelBuffer { get }

    open var isDepthDataFiltered: Bool { get }

    open var depthDataAccuracy: AVDepthDataAccuracy { get }

}
Introducing AVDepthData

@available(iOS 11.0, *)
open class AVDepthData: NSObject {

    open var depthDataType: OSType { get }

    open var depthDataMap: CVPixelBuffer { get }

    open var isDepthDataFiltered: Bool { get }

    open var depthDataAccuracy: AVDepthDataAccuracy { get }

}
Introducing AVDepthData

@available(iOS 11.0, *)
open class AVDepthData: NSObject {

    open var depthDataType: OSType { get }

    open var depthDataMap: CVPixelBuffer { get }

    open var isDepthDataFiltered: Bool { get }

    open var depthDataAccuracy: AVDepthDataAccuracy { get }

}
Introducing AVDepthData

@available(iOS 11.0, *)
open class AVDepthData: NSObject {

    open var depthDataType: OSType { get }

    open var depthDataMap: CVPixelBuffer { get }

    open var isDepthDataFiltered: Bool { get }

    open var depthDataAccuracy: AVDepthDataAccuracy { get }

}
Holes

Camera 1

Camera 2
Holes

Observed Point

Camera 1

Camera 2
Holes

Observed Point

Camera 1

Camera 2
Holes

Observed Point

Camera 1

Camera 2

??

??
Hole-Reporting in AVDepthData

\[ \text{NaN} = \text{Invalid depthDataMap value} \]

```swift
open var isDepthDataFiltered: Bool { get }
```
Calibration Errors

Camera 1

Camera 2
Calibration Errors

Camera 1

Camera 2
Calibration Errors

Camera 1

Baseline

Camera 2
Calibration Errors

Camera 1

Baseline

Camera 2
Calibration Errors

Camera 1

Baseline

Camera 2
Calibration Errors

Optical Image Stabilization

Gravity

Focus Coil
Depth Data Accuracy

```swift
extension AVDepthData {
    public enum Accuracy: Int {
        case relative
        case absolute
    }
}
```
\[ \hat{d} = d + e \]

- Relative disparity
- Absolute disparity
- Fixed error
\[ \hat{d}_1 - \hat{d}_2 = (d_1 + e) - (d_2 + e) \]
\( \hat{d}_1 - \hat{d}_2 = (d_1 + e) - (d_2 + e) \)

\( \hat{d}_1 - \hat{d}_2 = d_1 - d_2 + e - e \)
\hat{d}_1 - \hat{d}_2 = d_1 - d_2 + e - e
\( \hat{d}_1 - \hat{d}_2 = d_1 - d_2 + \epsilon - \epsilon \)
\[ \hat{d}_1 - \hat{d}_2 = d_1 - d_2 \]
\hat{d}_1 - \hat{d}_2 = d_1 - d_2
Streaming Depth Data
Demo
AVCamPhotoFilter
Introducing AVCaptureDepthDataOutput
Introducing AVCaptureDepthDataOutput
Introducing AVCaptureDepthDataOutput
Introducing AVCaptureDepthDataOutput

AVCaptureDevice (DualCamera)

AVCaptureDeviceInput

AVCaptureSession

AVCaptureDepthDataOutput
Introducing AVCaptureDepthDataOutput
AVCaptureDepthDataOutput

Only supported on Dual Camera
AVCaptureDepthDataOutput

Only supported on Dual Camera

Dual Camera auto-zooms to 2x (zoom is disabled)
AVCaptureDepthDataOutput

Only supported on Dual Camera

Dual Camera auto-zooms to 2x (zoom is disabled)

Dual Camera video formats have a `supportedDepthDataFormats` property
AVCaptureDepthDataOutput

Only supported on Dual Camera

Dual Camera auto-zooms to 2x (zoom is disabled)

Dual Camera video formats have a `supportedDepthDataFormats` property

AVCaptureDevice has new `activeDepthDataFormat` property
## Supported Depth Resolutions for Streaming

<table>
<thead>
<tr>
<th>Preset or Active Format</th>
<th>AVCaptureVideoDataOutput</th>
<th>AVCaptureDepthDataOutput</th>
</tr>
</thead>
<tbody>
<tr>
<td>.photo</td>
<td>1440x1080 @ 30</td>
<td>320x240 @ 24 fps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>160x120 @ 24 fps</td>
</tr>
<tr>
<td>1280x720@3-30 FPS</td>
<td>1280x720 @ 30</td>
<td>320x180 @ 24 fps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>160x90 @ 24 fps</td>
</tr>
<tr>
<td>.vga640x480</td>
<td>640x480 @ 30</td>
<td>320x240 @ 24 fps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>160x120 @ 24 fps</td>
</tr>
</tbody>
</table>
## Supported Depth Resolutions for Streaming

<table>
<thead>
<tr>
<th>Preset or Active Format</th>
<th>AVCaptureVideoDataOutput</th>
<th>AVCaptureDepthDataOutput</th>
</tr>
</thead>
<tbody>
<tr>
<td>.photo</td>
<td>1440x1080 @ 30</td>
<td>320x240 @ 24 fps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>160x120 @ 24 fps</td>
</tr>
<tr>
<td>1280x720@3-30 FPS</td>
<td>1280x720 @ 30</td>
<td>320x180 @ 24 fps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>160x90 @ 24 fps</td>
</tr>
<tr>
<td>.vga640x480</td>
<td>640x480 @ 30</td>
<td>320x240 @ 24 fps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>160x120 @ 24 fps</td>
</tr>
</tbody>
</table>
## Supported Depth Resolutions for Streaming

<table>
<thead>
<tr>
<th>Preset or Active Format</th>
<th>AVCaptureVideoDataOutput</th>
<th>AVCaptureDepthDataOutput</th>
</tr>
</thead>
<tbody>
<tr>
<td>.photo</td>
<td>1440x1080 @ 30</td>
<td>320x240 @ 24 fps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>160x120 @ 24 fps</td>
</tr>
<tr>
<td>1280x720@3-30 FPS</td>
<td>1280x720 @ 30</td>
<td>320x180 @ 24 fps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>160x90 @ 24 fps</td>
</tr>
<tr>
<td>.vga640x480</td>
<td>640x480 @ 30</td>
<td>320x240 @ 24 fps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>160x120 @ 24 fps</td>
</tr>
</tbody>
</table>
## Supported Depth Resolutions for Streaming

<table>
<thead>
<tr>
<th>Preset or Active Format</th>
<th>AVCaptureVideoDataOutput</th>
<th>AVCaptureDepthDataOutput</th>
</tr>
</thead>
<tbody>
<tr>
<td>.photo</td>
<td>1440x1080 @ 30</td>
<td>320x240 @ 24 fps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>160x120 @ 24 fps</td>
</tr>
<tr>
<td>1280x720@3-30 FPS</td>
<td>1280x720 @ 30</td>
<td>320x180 @ 24 fps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>160x90 @ 24 fps</td>
</tr>
<tr>
<td>.vga640x480</td>
<td>640x480 @ 30</td>
<td>320x240 @ 24 fps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>160x120 @ 24 fps</td>
</tr>
</tbody>
</table>
# Depth Frame Rate Examples

<table>
<thead>
<tr>
<th>VideoDataOutput</th>
<th>DepthDataOutput</th>
</tr>
</thead>
<tbody>
<tr>
<td>1440x1080 @ 24</td>
<td>320x240 @ 24 fps</td>
</tr>
<tr>
<td>1440x1080 @ 30</td>
<td>320x240 @ 15 fps</td>
</tr>
</tbody>
</table>
# Depth Frame Rate Examples

<table>
<thead>
<tr>
<th>VideoDataOutput</th>
<th>DepthDataOutput</th>
</tr>
</thead>
<tbody>
<tr>
<td>1440x1080 @ 24</td>
<td>320x240 @ 24 fps</td>
</tr>
<tr>
<td>1440x1080 @ 30</td>
<td>320x240 @ 15 fps</td>
</tr>
</tbody>
</table>
## Depth Frame Rate Examples

<table>
<thead>
<tr>
<th>VideoDataOutput</th>
<th>DepthDataOutput</th>
</tr>
</thead>
<tbody>
<tr>
<td>1440x1080 @ 24</td>
<td>320x240 @ 24 fps</td>
</tr>
<tr>
<td>1440x1080 @ 30</td>
<td>320x240 @ 15 fps</td>
</tr>
</tbody>
</table>
Depth Filtering with AVCaptureDepthDataOutput

open var isFilteringEnabled: Bool
Unsynchronized Data Output

AVCaptureDevice (DualCamera) -> AVCaptureDeviceInput

AVCaptureDevice (Microphone) -> AVCaptureDeviceInput

AVCaptureSession

AVCaptureVideoDataOutput

video sbufs
Unsynchronized Data Output

AVCaptureDeviceInput (DualCamera) -> AVCaptureVideoDataOutput

AVCaptureDeviceInput (Microphone) -> AVCaptureAudioDataOutput

AVCaptureSession

video sbufs

audio sbufs
Unsynchronized Data Output

AVCaptureDeviceInput

- AVCaptureVideoDataOutput
  - video sbufs
- AVCaptureMetadataOutput
  - faces or codes

AVCaptureDeviceInput

- AVCaptureAudioDataOutput
  - audio sbufs

AVCaptureDevice

- AVCaptureDevice (DualCamera)
- AVCaptureDevice (Microphone)

AVCaptureDeviceInput

AVCaptureSession
Unsynchronized Data Output

AVCaptureDevice (DualCamera)
  AVCaptureDeviceInput
  AVCaptureVideoDataOutput
  AVCaptureMetadataOutput
    video sbufs
  AVCaptureDepthDataOutput
    faces or codes
  AVCaptureAudioDataOutput
    audio sbufs

AVCaptureDevice (Microphone)
  AVCaptureDeviceInput
  AVCaptureAudioDataOutput
    audio sbufs
Synchronized Data Output

AVCaptureDevice (DualCamera)
- AVCaptureDeviceInput
- AVCaptureVideoDataOutput
- AVCaptureDepthDataOutput
- AVCaptureMetadataOutput

AVCaptureDevice (Microphone)
- AVCaptureDeviceInput
- AVCaptureAudioDataOutput

AVCaptureSession
- AVCaptureDataOutputSynchronizer

AVCaptureSynchronizedDataCollection
func dataOutputSynchronizer(_ synchronizer: AVCaptureDataOutputSynchronizer, didOutput synchronizedDataCollection: AVCaptureSynchronizedDataCollection) {

    // Iterate through an AVCaptureSynchronizedDataCollection using fast enumeration

    for syncedData in synchronizedDataCollection {
        if let syncedDepthData = syncedData as? AVCaptureSynchronizedDepthData {
            // ...
        }
    }
}
func dataOutputSynchronizer(_ synchronizer: AVCaptureDataOutputSynchronizer, didOutput synchronizedDataCollection: AVCaptureSynchronizedDataCollection) {

    // Iterate through an AVCaptureSynchronizedDataCollection using fast enumeration
    
    for syncedData in synchronizedDataCollection {
        if let syncedDepthData = syncedData as? AVCaptureSynchronizedDepthData {
            // ...
        }
    }
}
func dataOutputSynchronizer(_ synchronizer: AVCaptureDataOutputSynchronizer, didOutput synchronizedDataCollection: AVCaptureSynchronizedDataCollection) {

    // Iterate through an AVCaptureSynchronizedDataCollection using fast enumeration
    for syncedData in synchronizedDataCollection {
        if let syncedDepthData = syncedData as? AVCaptureSynchronizedDepthData {
            // ...
        }
    }
}
func dataOutputSynchronizer(_ synchronizer: AVCaptureDataOutputSynchronizer, didOutput synchronizedDataCollection: AVCaptureSynchronizedDataCollection) {

    // Use dictionary-esque subscripting to find a particular data

    if let synDepth = synchronizedDataCollection[self.ddo] as? AVCaptureSynchronizedDepthData {
        // ...
    }
}
func dataOutputSynchronizer(_ synchronizer: AVCaptureDataOutputSynchronizer, didOutput synchronizedDataCollection: AVCaptureSynchronizedDataCollection) {

    // Use dictionary-esque subscripting to find a particular data
    if let synDepth = synchronizedDataCollection[self.ddo] as? AVCaptureSynchronizedDepthData {
        // ...
    }
}
AVCamPhotoFilter
Sample code
Streaming Camera Intrinsics

Optical Center

Focal Length
open class AVCaptureConnection : NSObject {
    open var isCameraIntrinsicMatrixDeliveryEnabled: Bool
}

Streaming Camera Intrinsics
Streaming Camera Intrinsics

\[ K = \begin{bmatrix}
    f_x & s & x_0 \\
    0 & f_y & y_0 \\
    0 & 0 & 1
\end{bmatrix} \]
Streaming Camera Intrinsics

\[ K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \]
Streaming Camera Intrinsics

\[ K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \]
Streaming Camera Intrinsics

\[ K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \]

kCMSampleBufferAttachmentKey_CameraIntrinsicMatrix
Capturing Photos with Depth
Photos with Depth

- Flash
- Auto still image stabilization
- Auto exposure brackets
- Live photos
Capturing Photos with Depth

AVCaptureDevice (Camera)

AVCaptureDeviceInput

AVCaptureSession

AVCapturePhotoOutput
Capturing Photos with Depth

AVCaptureDevice (Camera)

AVCaptureDeviceInput

AVCaptureSession

AVCapturePhotoOutput
Capturing Photos with Depth

AVCaptureDevice
(Camera)

AVCaptureDeviceInput

AVCaptureSession

AVCapturePhotoOutput
Capturing Photos with Depth

AVCaptureDeviceInput

AVCaptureSession

AVCapturePhotoOutput
Capturing Photos with Depth

- AVCaptureDevice (Camera)
- AVCaptureDeviceInput
- AVCaptureSession
- AVCapturePhotoOutput
func photoOutput(_ output: AVCapturePhotoOutput, 
didFinishProcessingPhoto photo: AVCapturePhoto, 
error: Error?)
Requesting Depth with Photos
Requesting Depth with Photos

Before starting the capture session, opt in!

```javascript
photoOutput.isDepthDataDeliveryEnabled = true
```
Requesting Depth with Photos

Before starting the capture session, opt in!
photoOutput.isDepthDataDeliveryEnabled = true

Request depth on a per-photo basis
photoSettings.isDepthDataDeliveryEnabled = true
Requesting Depth with Photos

Before starting the capture session, opt in!

```swift
photoOutput.isDepthDataDeliveryEnabled = true
```

Request depth on a per-photo basis

```swift
photoSettings.isDepthDataDeliveryEnabled = true
```

Work with the resulting `AVDepthData` in your `AVCapturePhoto`

```swift
open class AVCapturePhoto: NSObject {
    open var depthData: AVDepthData? { get }
}
```
# High Res Photo Depth Maps

<table>
<thead>
<tr>
<th>Preset or Active Format</th>
<th>Streaming Video Resolution</th>
<th>High Res Photo Resolution</th>
<th>Streaming Depth Resolution</th>
<th>Photo Depth Map Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>.photo</td>
<td>1440x1080</td>
<td>4032x3024</td>
<td>320x240</td>
<td>768x576</td>
</tr>
<tr>
<td>1280x720@3-30 FPS</td>
<td>1280x720</td>
<td>4096x2304</td>
<td>320x180</td>
<td>768x432</td>
</tr>
<tr>
<td>.vga640x480</td>
<td>640x480</td>
<td>4032x3024</td>
<td>320x240</td>
<td>768x576</td>
</tr>
</tbody>
</table>
## High Res Photo Depth Maps

<table>
<thead>
<tr>
<th>Preset or Active Format</th>
<th>Streaming Video Resolution</th>
<th>High Res Photo Resolution</th>
<th>Streaming Depth Resolution</th>
<th>Photo Depth Map Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>.photo</td>
<td>1440x1080</td>
<td>4032x3024</td>
<td>320x240 160x120</td>
<td>768x576</td>
</tr>
<tr>
<td>1280x720@3-30 FPS</td>
<td>1280x720</td>
<td>4096x2304</td>
<td>320x180 160x90</td>
<td>768x432</td>
</tr>
<tr>
<td>.vga640x480</td>
<td>640x480</td>
<td>4032x3024</td>
<td>320x240 160x120</td>
<td>768x576</td>
</tr>
</tbody>
</table>
# High Res Photo Depth Maps

<table>
<thead>
<tr>
<th>Preset or Active Format</th>
<th>Streaming Video Resolution</th>
<th>High Res Photo Resolution</th>
<th>Streaming Depth Resolution</th>
<th>Photo Depth Map Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>.photo</td>
<td>1440x1080</td>
<td>4032x3024</td>
<td>320x240</td>
<td>768x576</td>
</tr>
<tr>
<td>1280x720@3-30 FPS</td>
<td>1280x720</td>
<td>4096x2304</td>
<td>320x180</td>
<td>768x432</td>
</tr>
<tr>
<td>.vga640x480</td>
<td>640x480</td>
<td>4032x3024</td>
<td>320x240</td>
<td>768x576</td>
</tr>
</tbody>
</table>
Rectilinear vs. Lens Distorted Images
Rectilinear vs. Lens Distorted Images
Rectilinear vs. Lens Distorted Images
Depth Map Distortions

Tele

Wide
Depth Map Distortions

Tele

Wide
Depth Map Distortions

Tele

Wide
Depth Map Distortions

Rectilinear Disparity Map
Depth Map Distortions

Rectilinear Disparity Map → Tele Warped Disparity Map
Depth Map Distortions

Depth data maps work well for applying effects to the images

Don't work well for 3D-scene reconstruction

Maps and images can be made rectilinear
Depth in Image Files

HEIF HEVC (HEIC)
• Map is an auxiliary image
• Encoded as monochrome HEVC
• Metadata is XMP in the auxiliary image
Depth in Image Files

JPEG
• Map is 8-bit lossy JPEG (filtered) or 16-bit lossless (unfiltered) MPO
• Metadata is XMP in the second image
Dual Photo Capture
Dual Photo Capture

AVCapturePhotoOutput

12 MP Wide

12 MP Tele

NEW
Requesting Dual Photo Delivery
Requesting Dual Photo Delivery

Before starting the capture session, opt in!

```javascript
photoOutput.isDualCameraDualPhotoDeliveryEnabled = true
```
Requesting Dual Photo Delivery

Before starting the capture session, opt in!

```javascript
photoOutput.isDualCameraDualPhotoDeliveryEnabled = true
```

Request Dual Photo capture on a per-request basis

```javascript
photoSettings.isDualCameraDualPhotoDeliveryEnabled = true
```
Requesting Dual Photo Delivery

Before starting the capture session, opt in!

```swift
photoOutput.isDualCameraDualPhotoDeliveryEnabled = true
```

Request Dual Photo capture on a per-request basis

```swift
photoSettings.isDualCameraDualPhotoDeliveryEnabled = true
```

Number of AVCapturePhoto callbacks doubles

```swift
let sourceDeviceType = photo.sourceDeviceType
```
Dual Photo Capture

Supports

• Flash captures
• Auto still image stabilization
• Auto exposure brackets
• Depth delivery
Dual Photo Capture + Zoom

Area outside of the preview field of view is blackened

Clean aperture rectangle defines the area with valid pixel data

12 MP Wide

12 MP Tele
Dual Photo Capture

Dual Photos can be delivered with camera calibration data

Make your own depth maps!

Augment reality!
Introducing AVCameraCalibrationData

Property of AVDepthData
Introducing AVCameraCalibrationData

Property of AVDepthData

Property of AVCapturePhoto (if you opt in):

```cpp
photoSettings.isCameraCalibrationDeliveryEnabled = true
```
open var intrinsicMatrix: matrix_float3x3 { get }

\[
K = \begin{bmatrix}
 f_x & s & x_0 \\
 0 & f_y & y_0 \\
 0 & 0 & 1 \\
\end{bmatrix}
\]
open var intrinsicMatrix: matrix_float3x3 { get }

\[
K = \begin{bmatrix}
f_x & s & x_0 \\
0 & f_y & y_0 \\
0 & 0 & 1
\end{bmatrix}
\]
AVCameraCalibrationData

open var intrinsicMatrix: matrix_float3x3 { get }

$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Optical Center**: $(x_0, y_0)$
- **Focal Length**: $f_x, f_y$
AVCameraCalibrationData

open var intrinsicMatrix: matrix_float3x3 { get }

\[
K = \begin{bmatrix}
  f_x & s & x_0 \\
  0 & f_y & y_0 \\
  0 & 0 & 1
\end{bmatrix}
\]

open var intrinsicMatrixReferenceDimensions: CGSize { get }

Focal Length
Optical Center
Focal Length
AVCameraCalibrationData

open var extrinsicMatrix: matrix_float4x3 { get }

open var extrinsicMatrix: matrix_float4x3 { get }

$$[R \mid t] = \begin{bmatrix}
  r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\
  r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\
  r_{3,1} & r_{3,2} & r_{3,3} & t_3
\end{bmatrix}$$
AVCameraCalibrationData

open var extrinsicMatrix: matrix_float4x3 { get }

\[
[R \mid t] = \begin{bmatrix}
  r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\
  r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\
  r_{3,1} & r_{3,2} & r_{3,3} & t_3 \\
\end{bmatrix}
\]

Rotation
open var extrinsicMatrix: matrix_float4x3 { get }
AVCameraCalibrationData

open var extrinsicMatrix: matrix_float4x3 { get }

\[
[R \mid t] = \begin{bmatrix}
r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\
r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\
r_{3,1} & r_{3,2} & r_{3,3} & t_3 
\end{bmatrix}
\]

Tele camera is the world origin
AVCameraCalibrationData
AVCameraCalibrationData

open var lensDistortionCenter: CGPoint { get }
AVCameraCalibrationData

open var lensDistortionCenter: CGPoint { get }

open var lensDistortionLookupTable: Data { get }
AVCameraCalibrationData

open var lensDistortionCenter: CGPoint { get }

open var lensDistortionLookupTable: Data { get }
open var inverseLensDistortionLookupTable: Data { get }
open var inverseLensDistortionLookupTable: Data { get }
Straighten Up
Summary

iPhone 7 Plus Dual Camera is a **disparity** system
Summary

iPhone 7 Plus Dual Camera is a **disparity** system.

The canonical representation for depth is `AVDepthData`. 
Summary

iPhone 7 Plus Dual Camera is a **disparity** system.

The canonical representation for depth is **AVDepthData**.

Intrinsic, extrinsic, and lens distortion info are stored in **AVCameraCalibrationData**.
Summary

iPhone 7 Plus Dual Camera is a **disparity** system.

The canonical representation for depth is **AVDepthData**.

Intrinsic, extrinsic, and lens distortion info are stored in **AVCameraCalibrationData**.

Filtered/unfiltered depth is streamed using **AVCaptureDepthDataOutput**.
Summary

iPhone 7 Plus Dual Camera is a **disparity** system

The canonical representation for depth is **AVDepthData**

Intrinsic, extrinsic, and lens distortion info are stored in **AVCameraCalibrationData**

Filtered/unfiltered depth is streamed using **AVCaptureDepthDataOutput**

Depth is captured in photos using **AVCapturePhotoOutput**
Summary

iPhone 7 Plus Dual Camera is a disparity system

The canonical representation for depth is AVDepthData.

Intrinsic, extrinsic, and lens distortion info are stored in AVCameraCalibrationData.

Filtered/unfiltered depth is streamed using AVCaptureDepthDataOutput.

Depth is captured in photos using AVCapturePhotoOutput.

Dual Camera Dual Photo Delivery produces wide and tele plus calibration.
Sample Code

AVCam

AVCamPhotoFilter

Wiggle Me
More Information

## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photography Get Together</td>
<td>Technology Lab J</td>
<td>Wednesday 6:30PM</td>
</tr>
<tr>
<td>Image Editing with Depth</td>
<td>Hall 2</td>
<td>Thursday 11:00AM</td>
</tr>
<tr>
<td>Working with HEIF and HEVC</td>
<td>Hall 2</td>
<td>Friday 11:00AM</td>
</tr>
<tr>
<td>Lab Name</td>
<td>Lab Location</td>
<td>Day</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Photo Extensions &amp; PhotoKit Lab</td>
<td>Technology Lab A</td>
<td>Thur</td>
</tr>
<tr>
<td>AVFoundation Lab</td>
<td>Technology Lab F</td>
<td>Thur</td>
</tr>
<tr>
<td>Photos Editing &amp; Core Image Lab</td>
<td>Technology Lab F</td>
<td>Thur</td>
</tr>
<tr>
<td>Photos Depth &amp; Capture Lab</td>
<td>Technology Lab A</td>
<td>Thur</td>
</tr>
<tr>
<td>HEIF and HEVC Lab</td>
<td>Technology Lab F</td>
<td>Fri</td>
</tr>
<tr>
<td>Vision Lab</td>
<td>Technology Lab A</td>
<td>Fri</td>
</tr>
<tr>
<td>Photos Depth &amp; Capture Lab</td>
<td>Technology Lab F</td>
<td>Fri</td>
</tr>
<tr>
<td>Lab Name</td>
<td>Location</td>
<td>Time</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Photo Extensions &amp; PhotoKit Lab</td>
<td>Technology Lab A</td>
<td>Thur 9:00AM–12:00PM</td>
</tr>
<tr>
<td>AVFoundation Lab</td>
<td>Technology Lab F</td>
<td>Thur 12:00PM–3:00PM</td>
</tr>
<tr>
<td>Photos Editing &amp; Core Image Lab</td>
<td>Technology Lab F</td>
<td>Thur 3:10PM–6:00PM</td>
</tr>
<tr>
<td>Photos Depth &amp; Capture Lab</td>
<td>Technology Lab A</td>
<td>Thur 3:10PM–6:00PM</td>
</tr>
<tr>
<td>HEIF and HEVC Lab</td>
<td>Technology Lab F</td>
<td>Fri 12:00PM–1:50PM</td>
</tr>
<tr>
<td>Vision Lab</td>
<td>Technology Lab A</td>
<td>Fri 1:50PM–4:00PM</td>
</tr>
<tr>
<td>Photos Depth &amp; Capture Lab</td>
<td>Technology Lab F</td>
<td>Fri 1:50PM–4:00PM</td>
</tr>
</tbody>
</table>