# 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

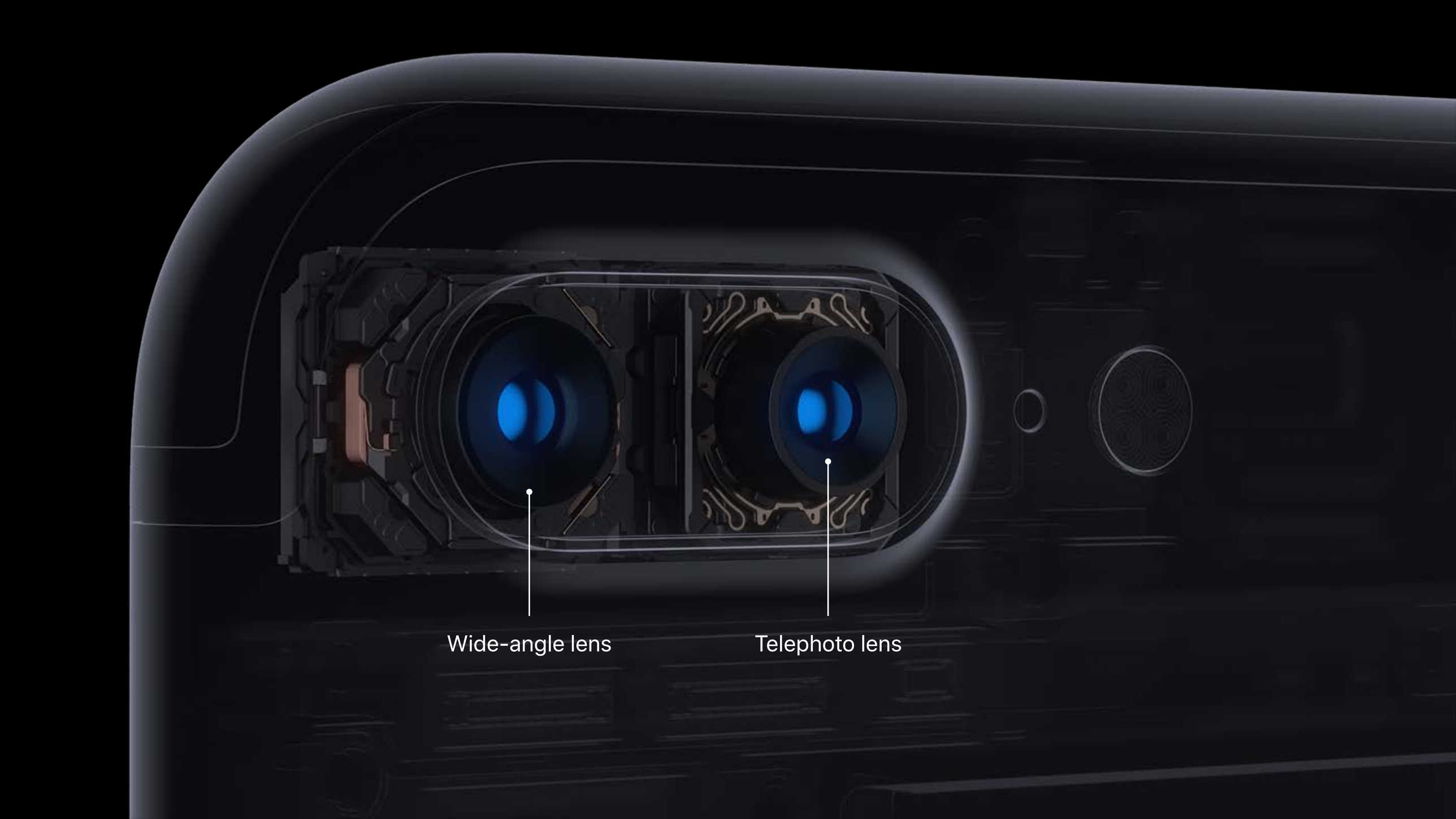
Dual photo capture

Depth and disparity on iPhone 7 Plus

Streaming depth data from the camera

Capturing photos with depth data

Dual photo capture


Wide-angle lens
Telephoto lens

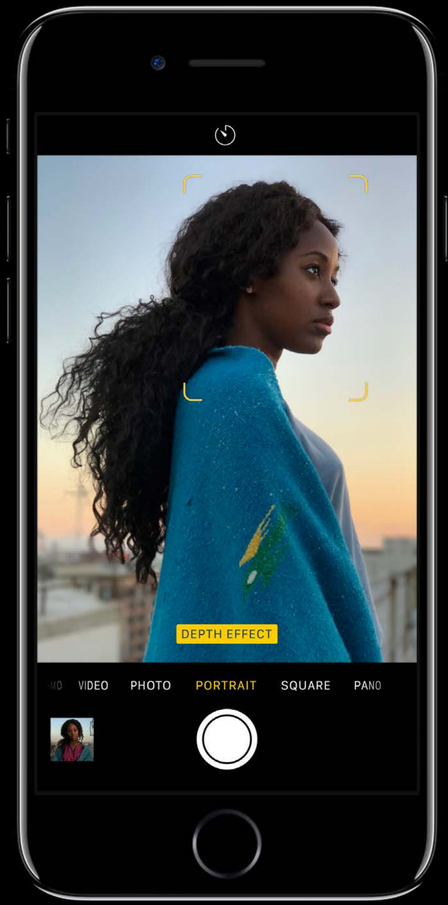
# Dual Camera Zoom

Switches between wide and tele automatically

Matches exposure, focus, and frame rate

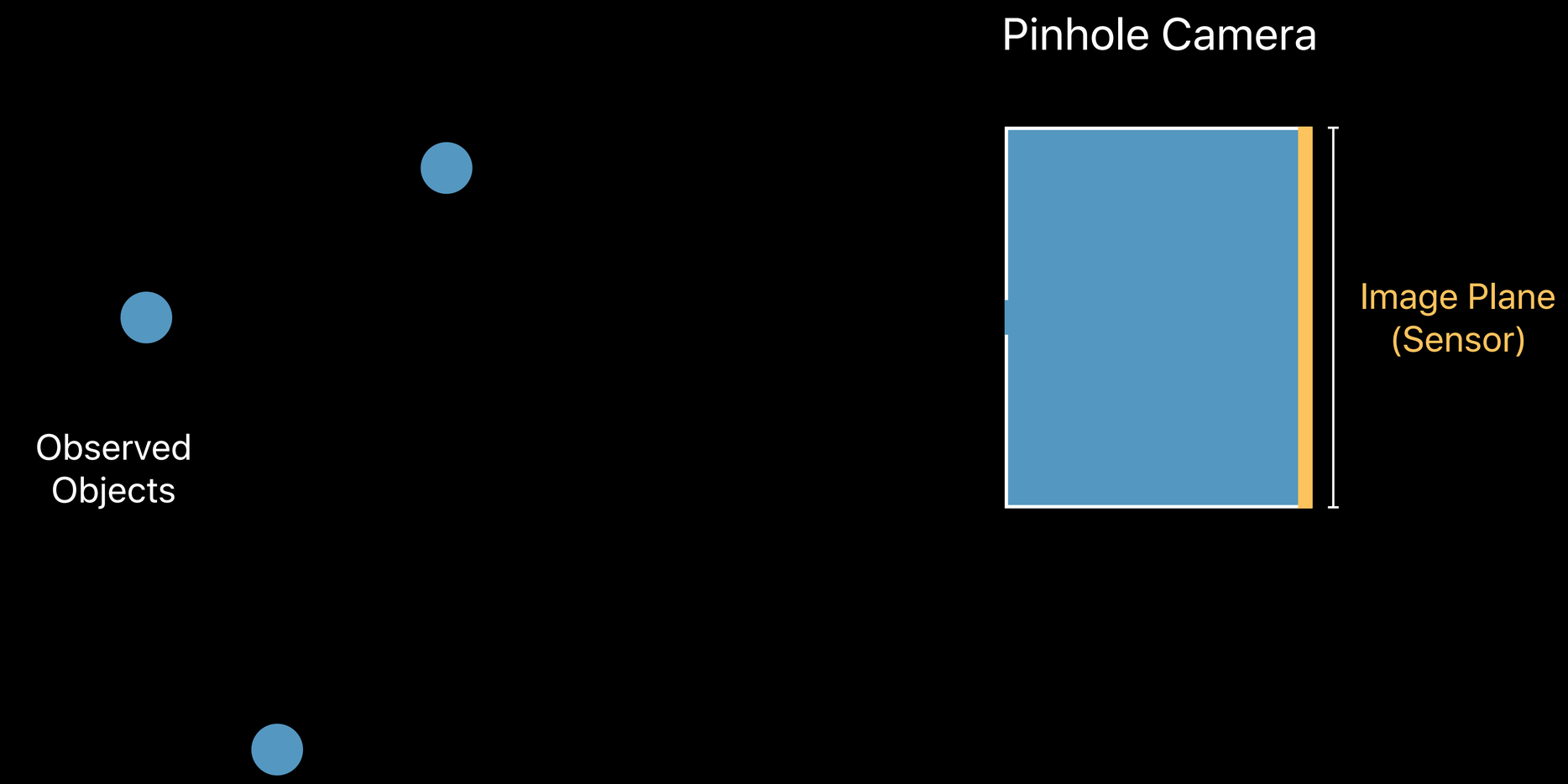
Compensates for parallax shift to smooth the transition

# Portrait Mode

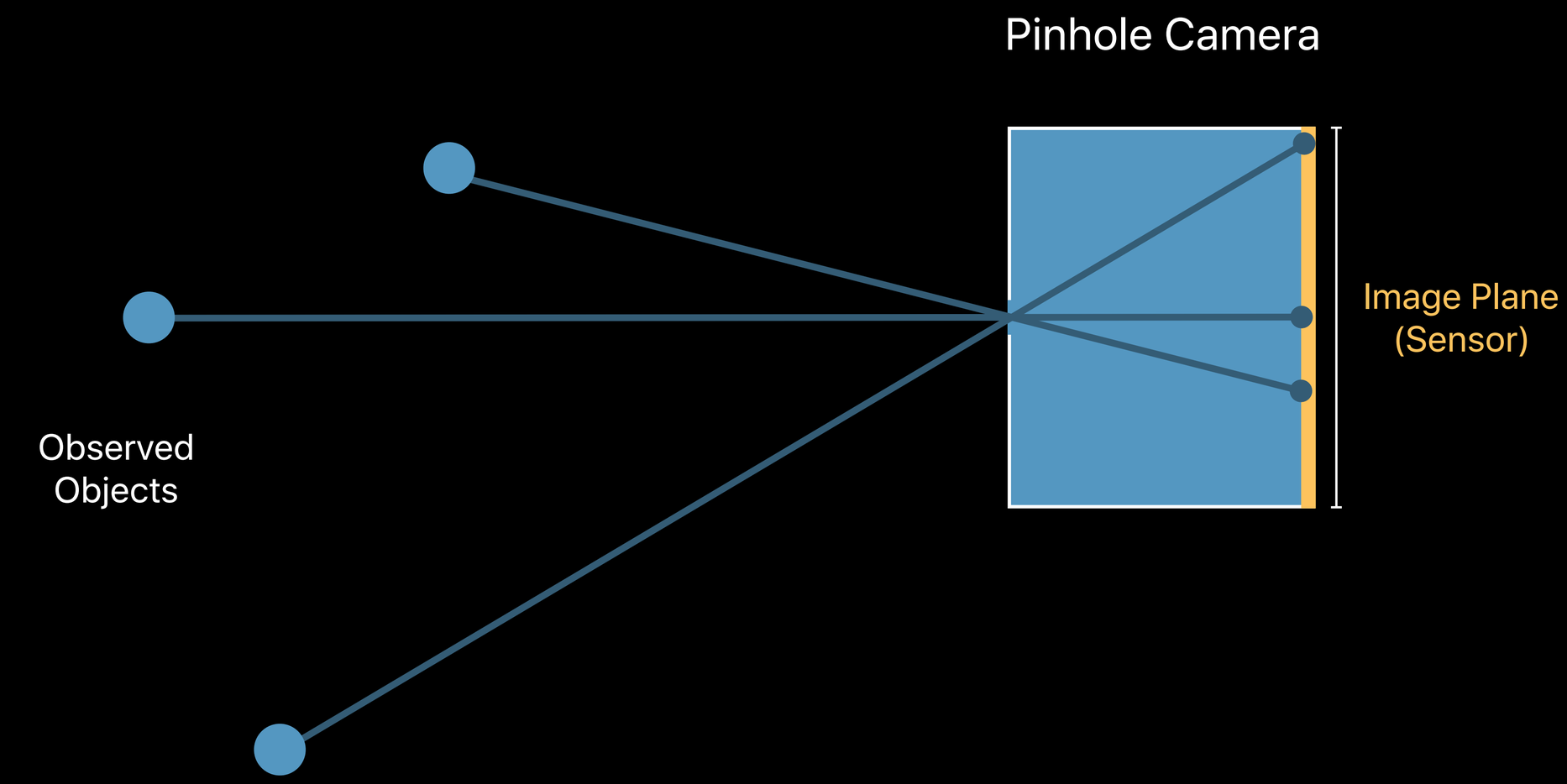


# Deep Learning

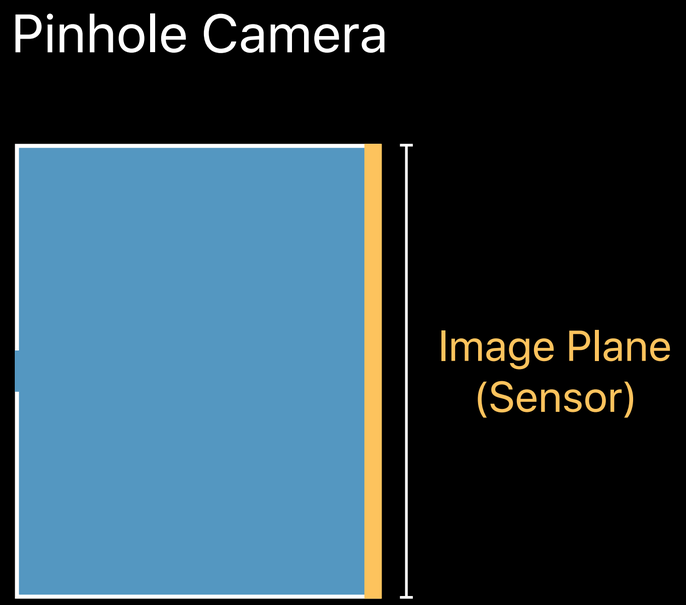
# Depth Map



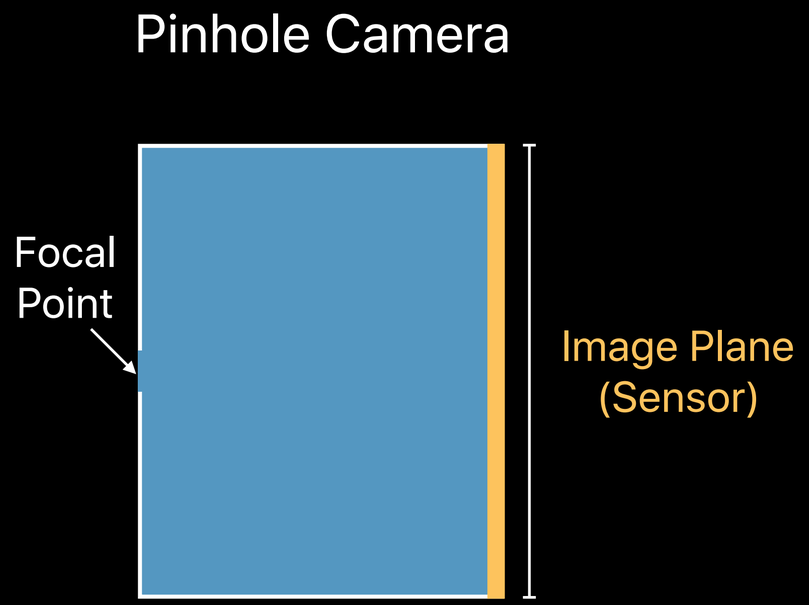
# Depth Map



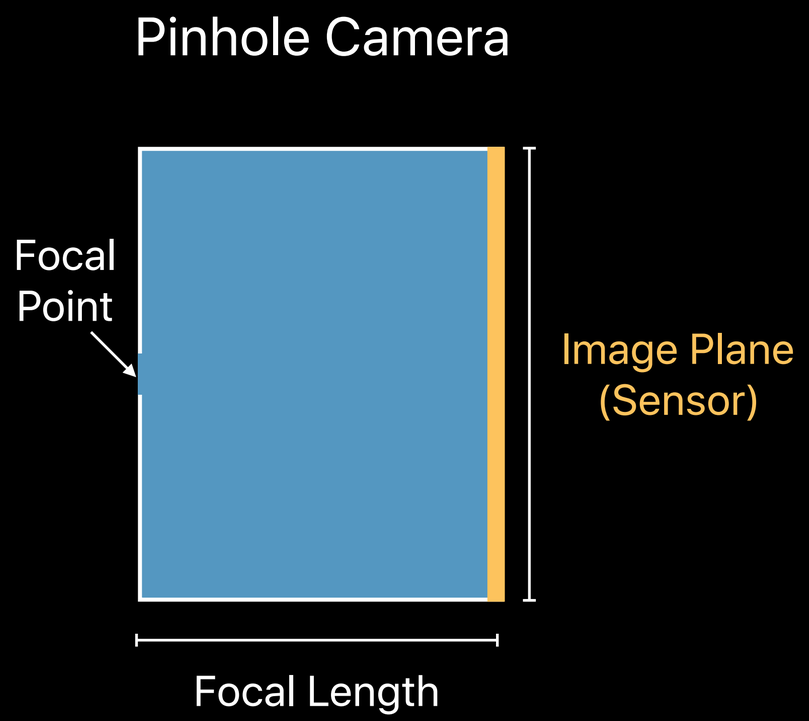
# Depth Map



# Depth Map

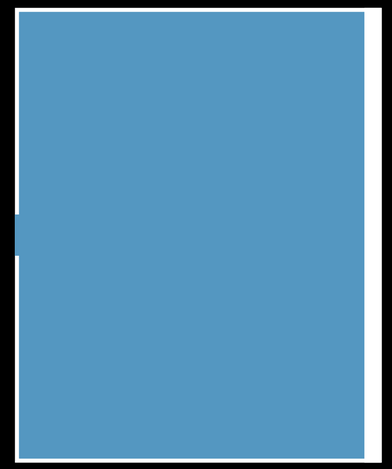


# Depth Map

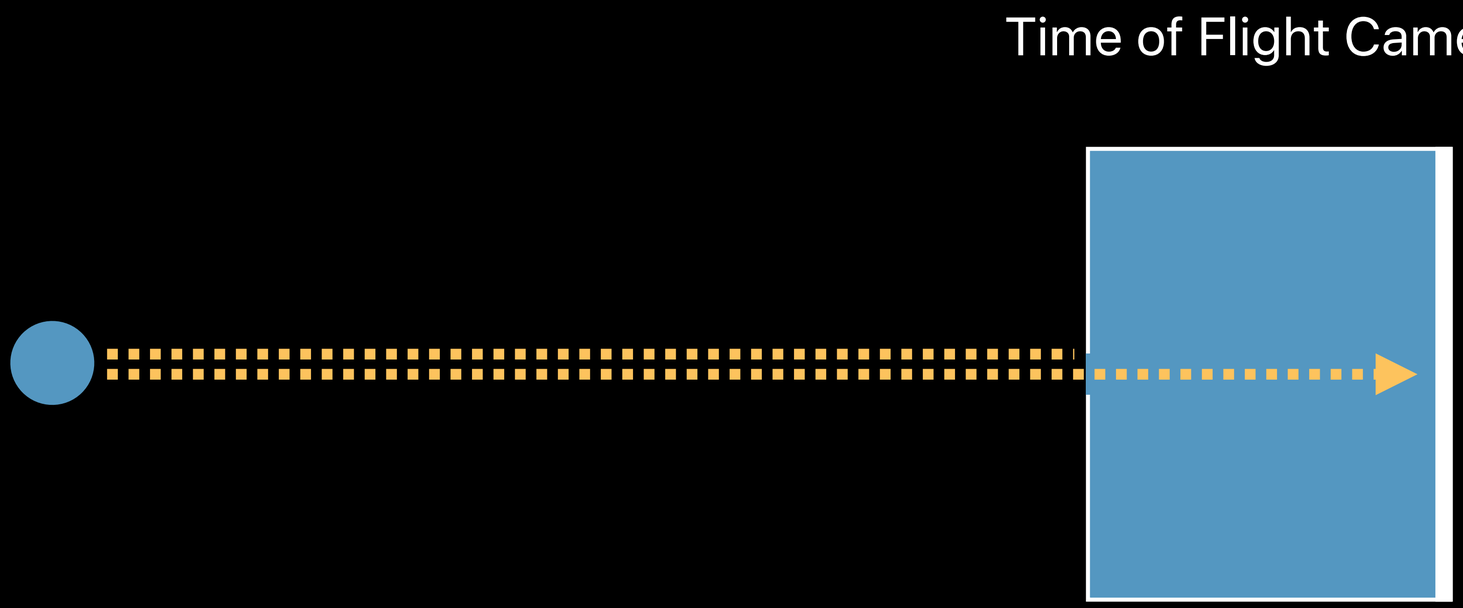

Pinhole Camera
Focal
Point
Image Plane
(Sensor)
Focal Length

# Depth Map

Time of Flight Camera

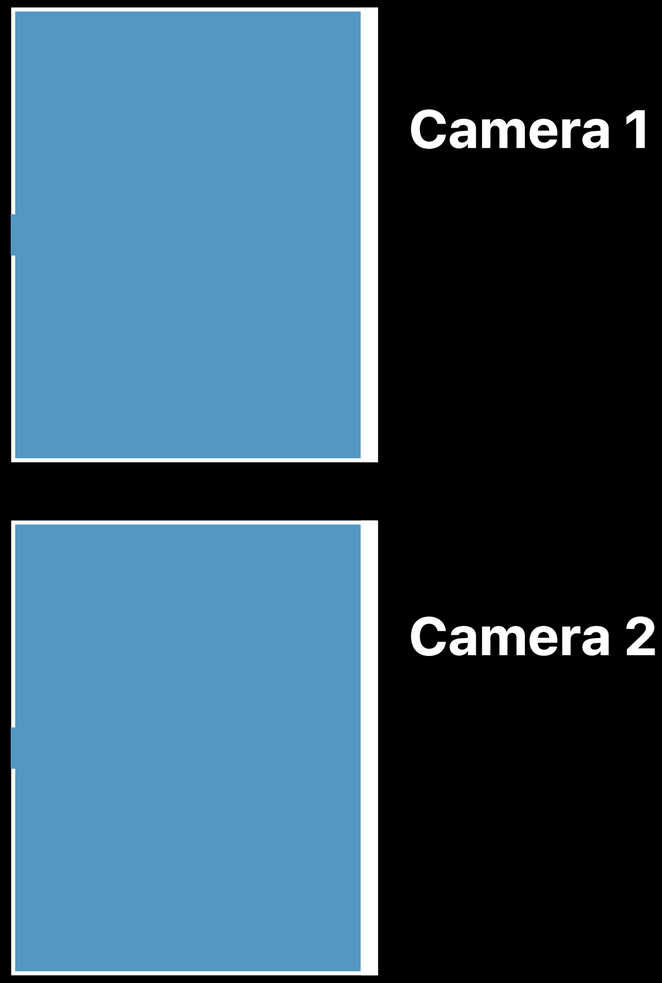


# Depth Map

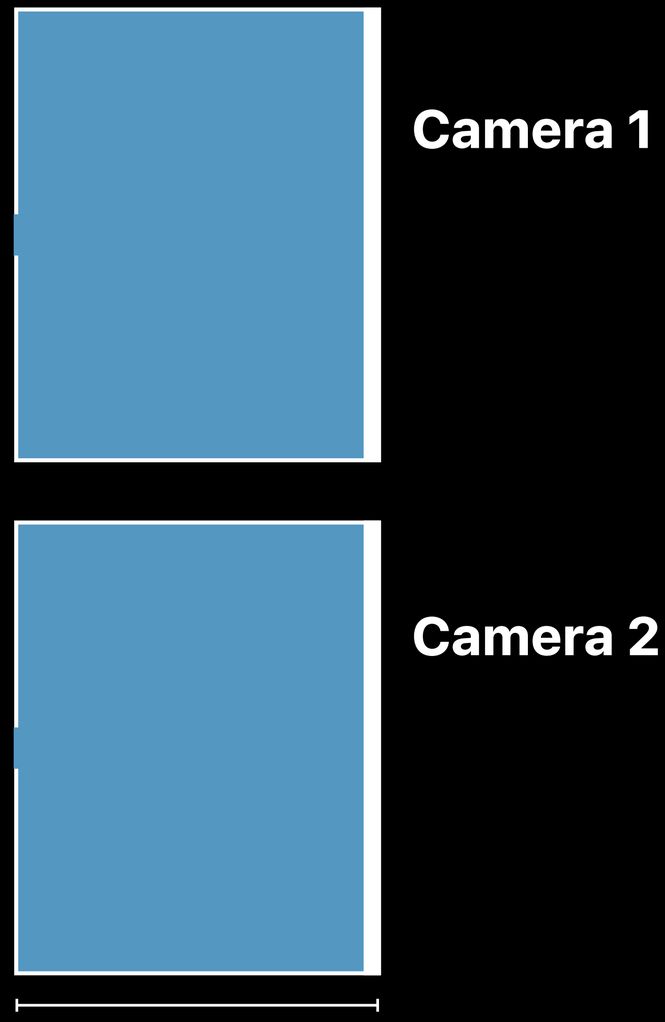


# Disparity

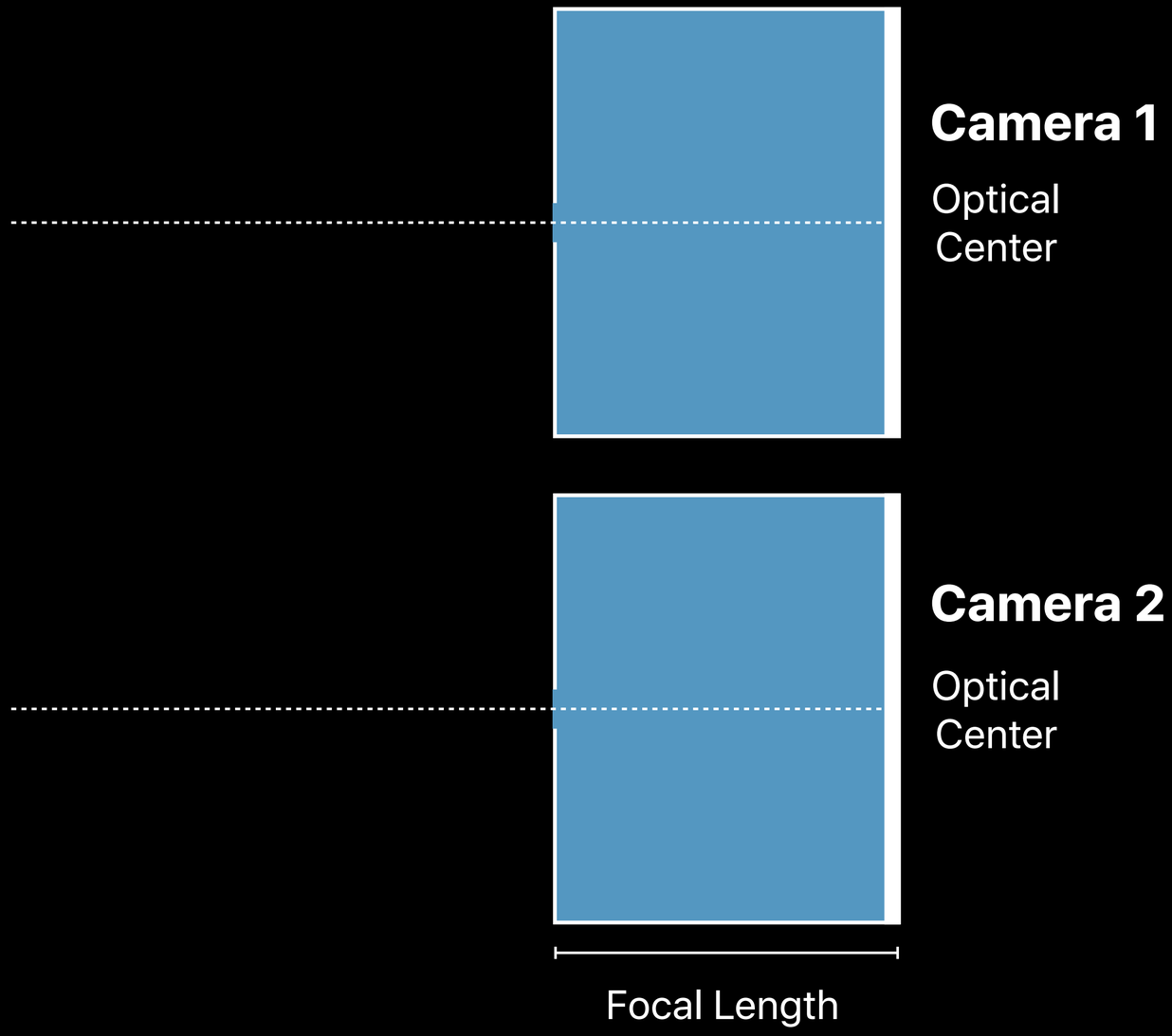
# Disparity Map



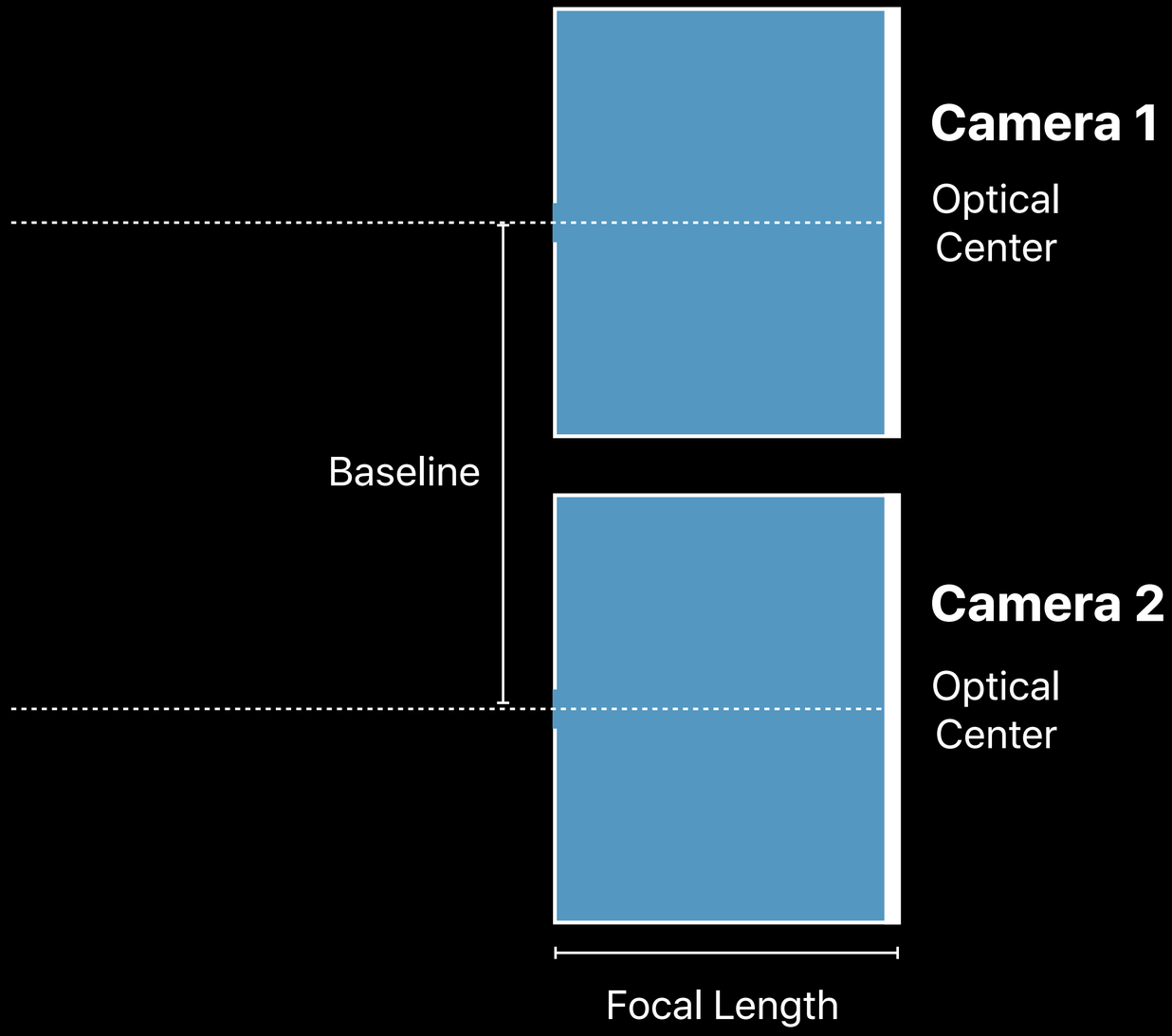
# Disparity Map



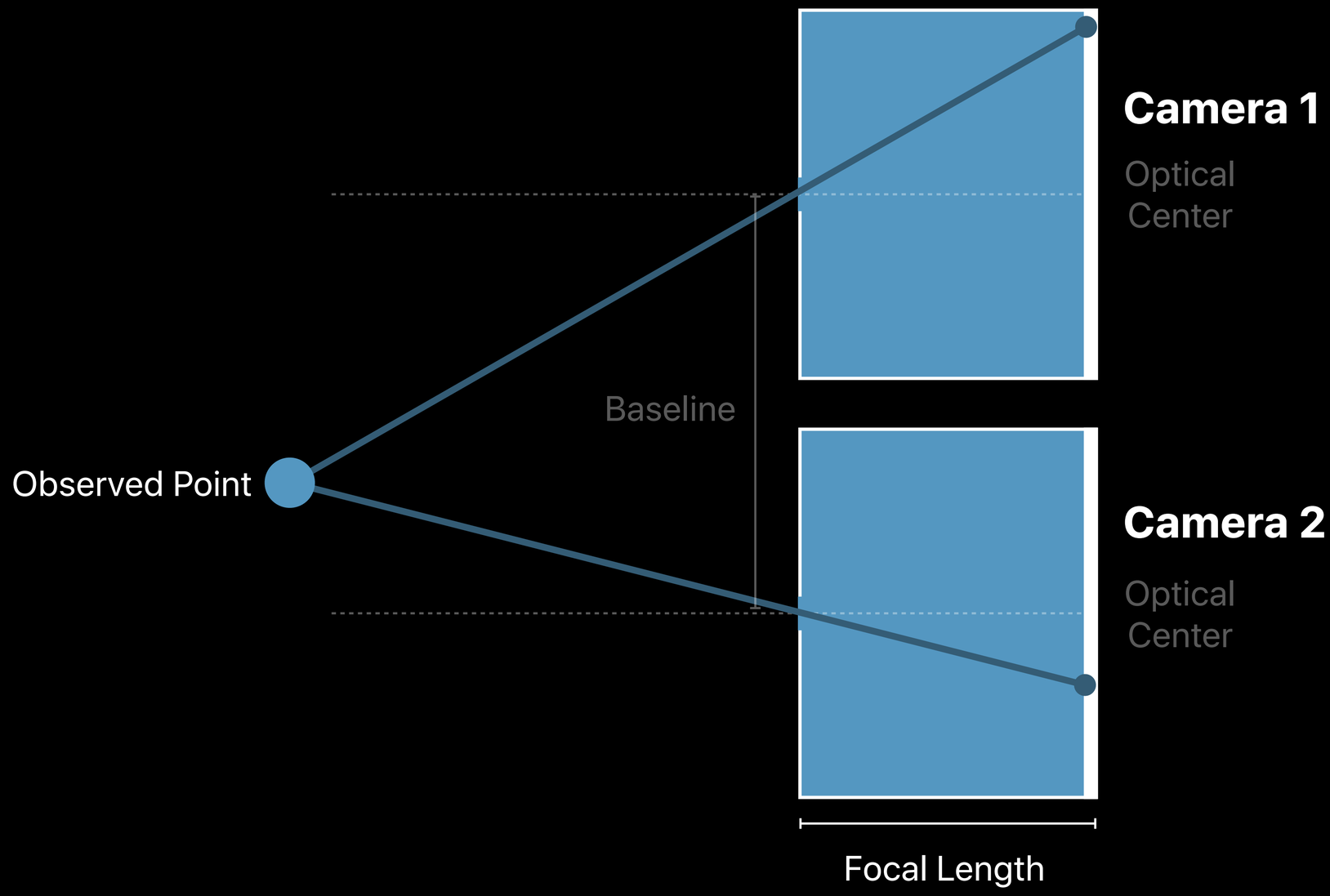
# Disparity Map



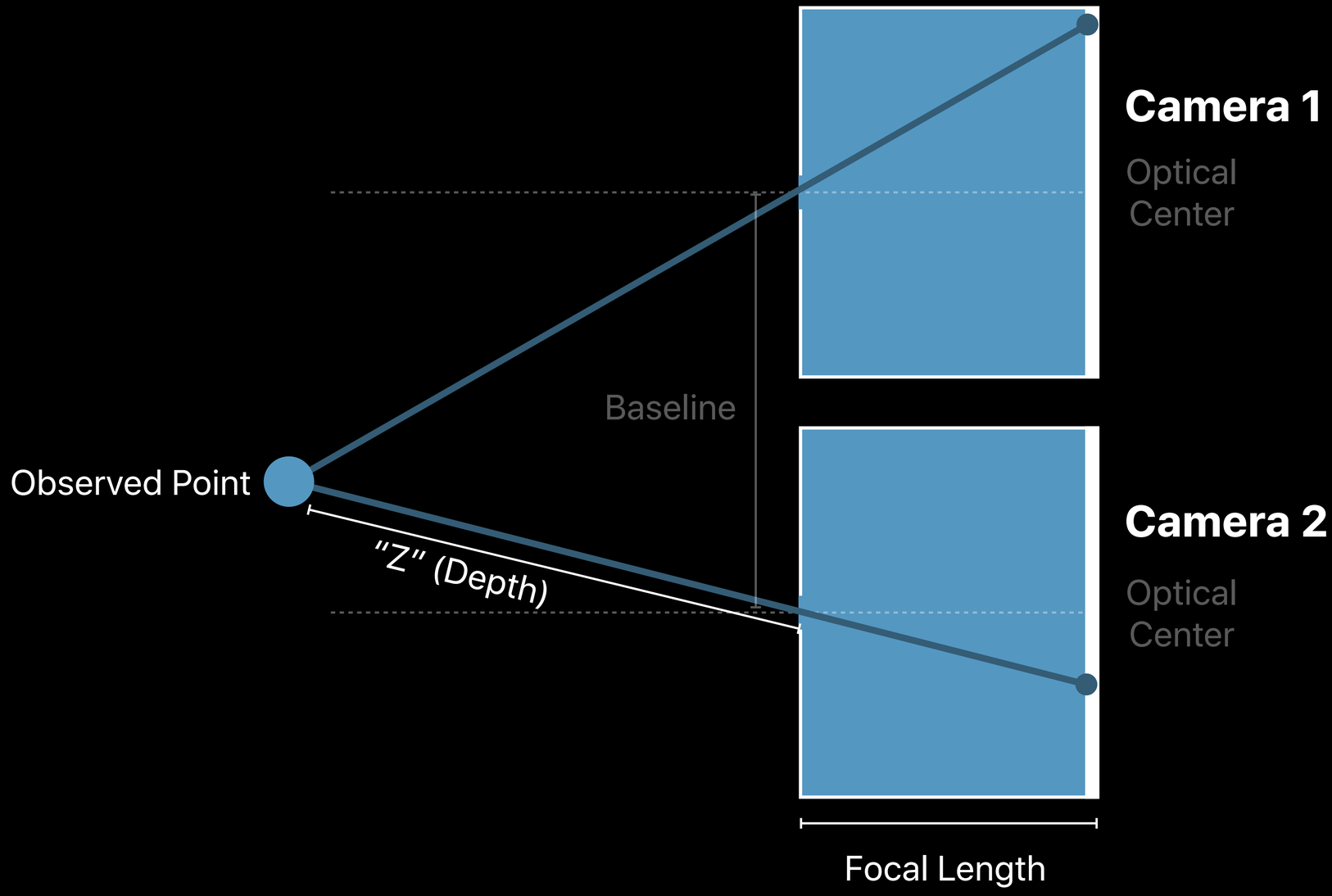
# Disparity Map



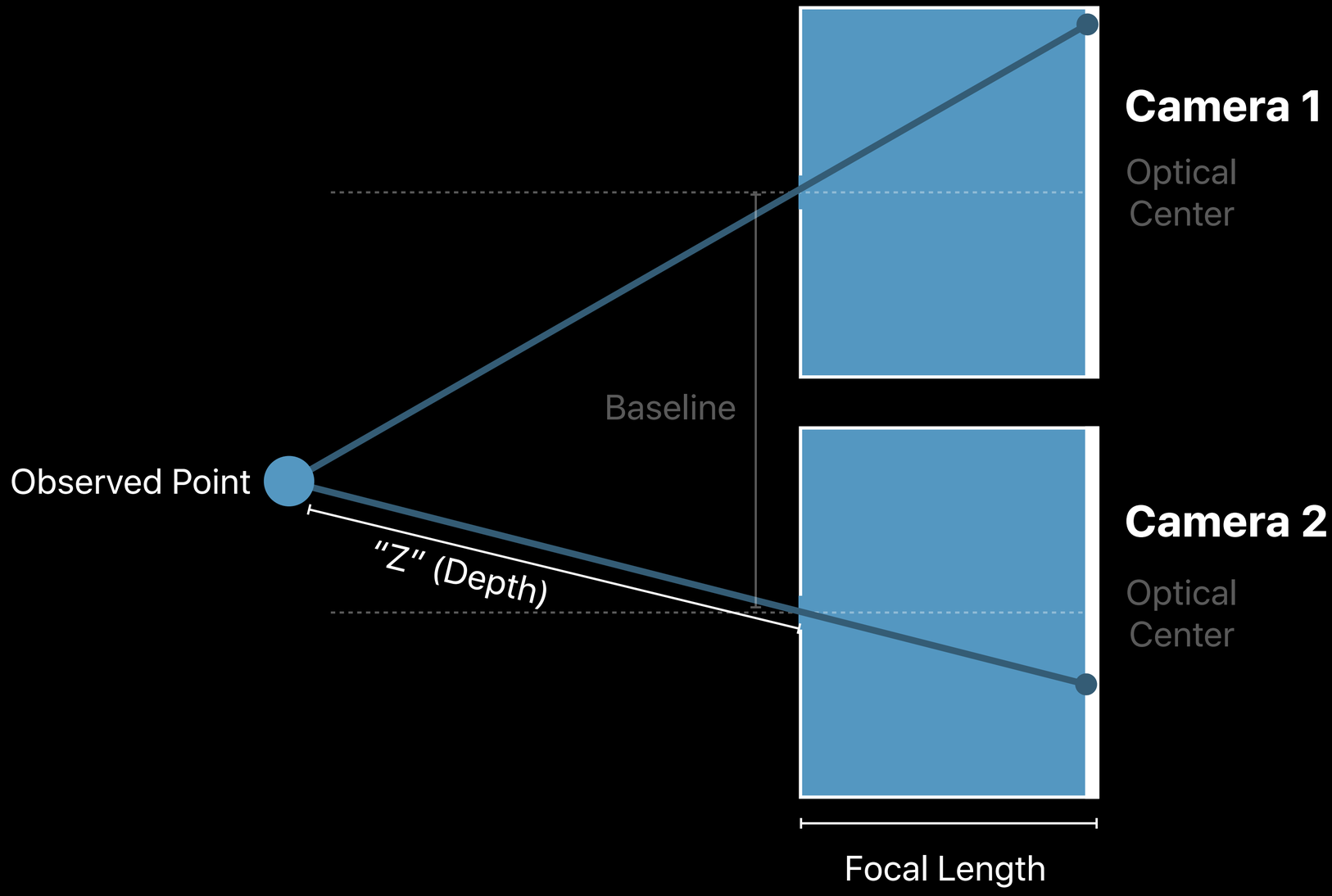
# Disparity Map



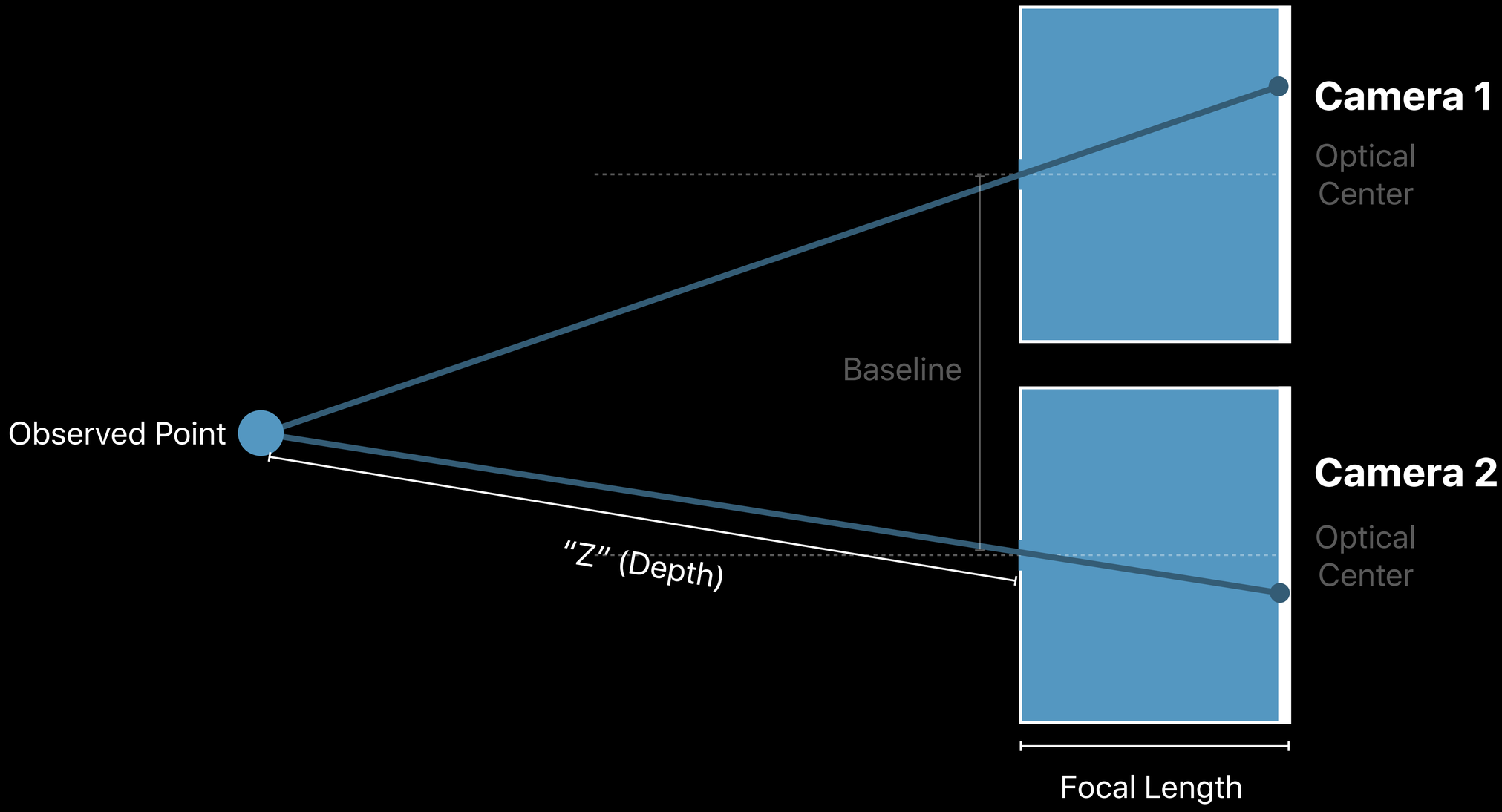
# Disparity Map



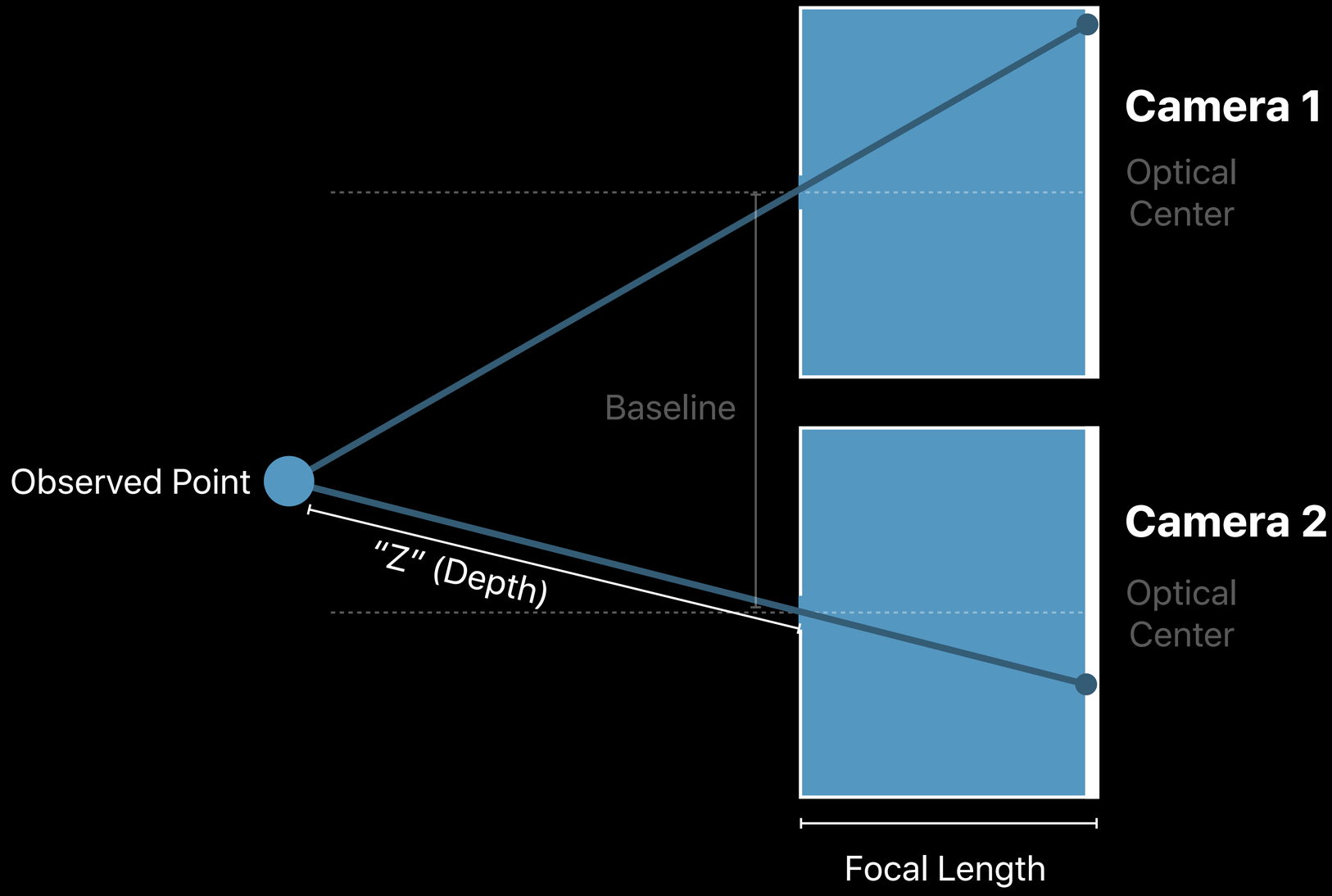
# Disparity Map



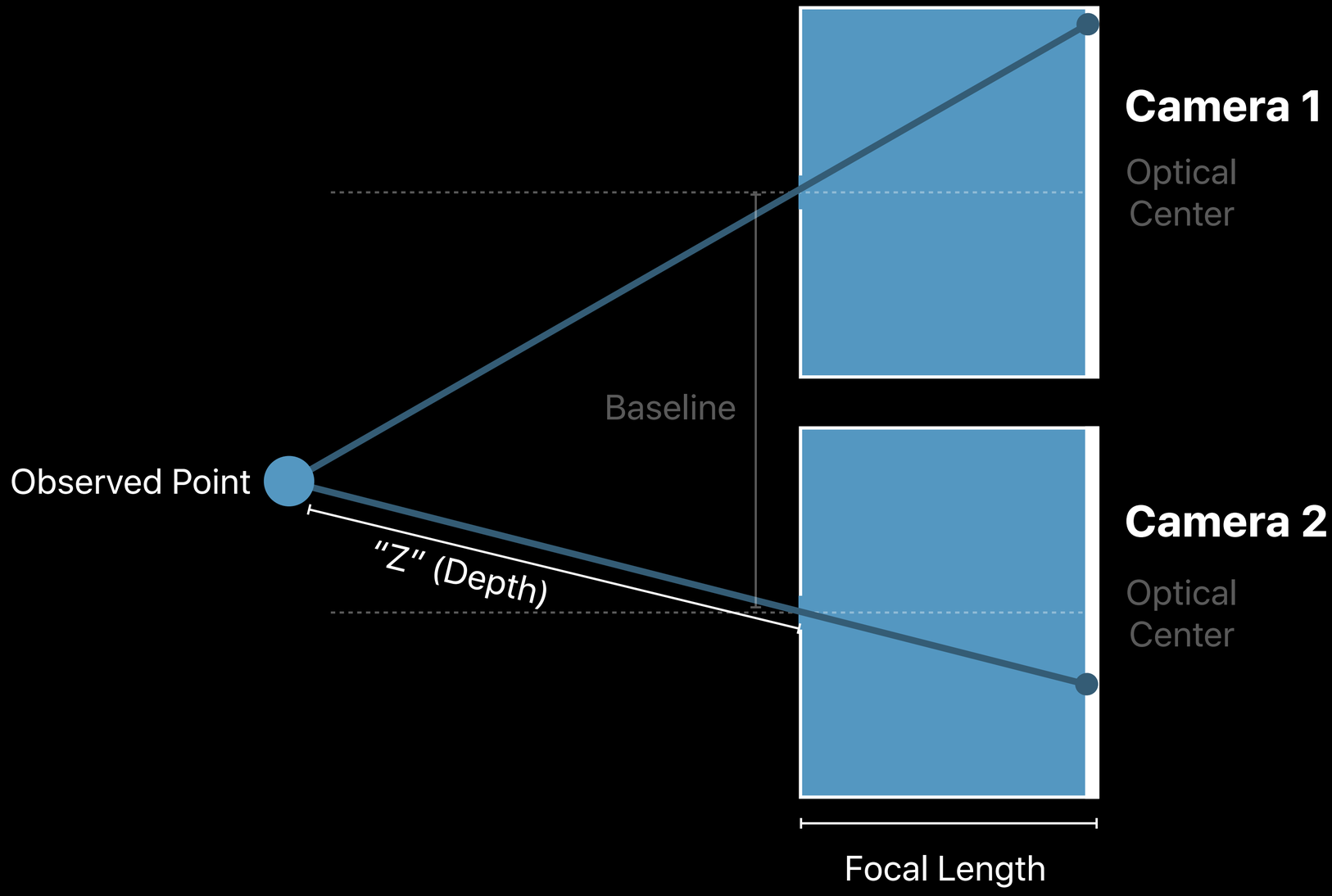
# Disparity Map



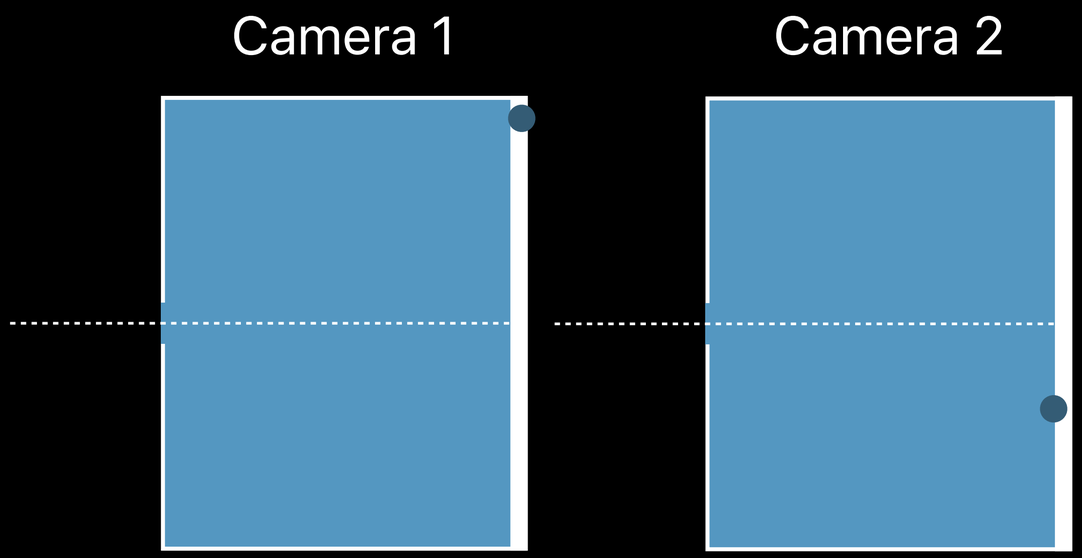
# Disparity Map



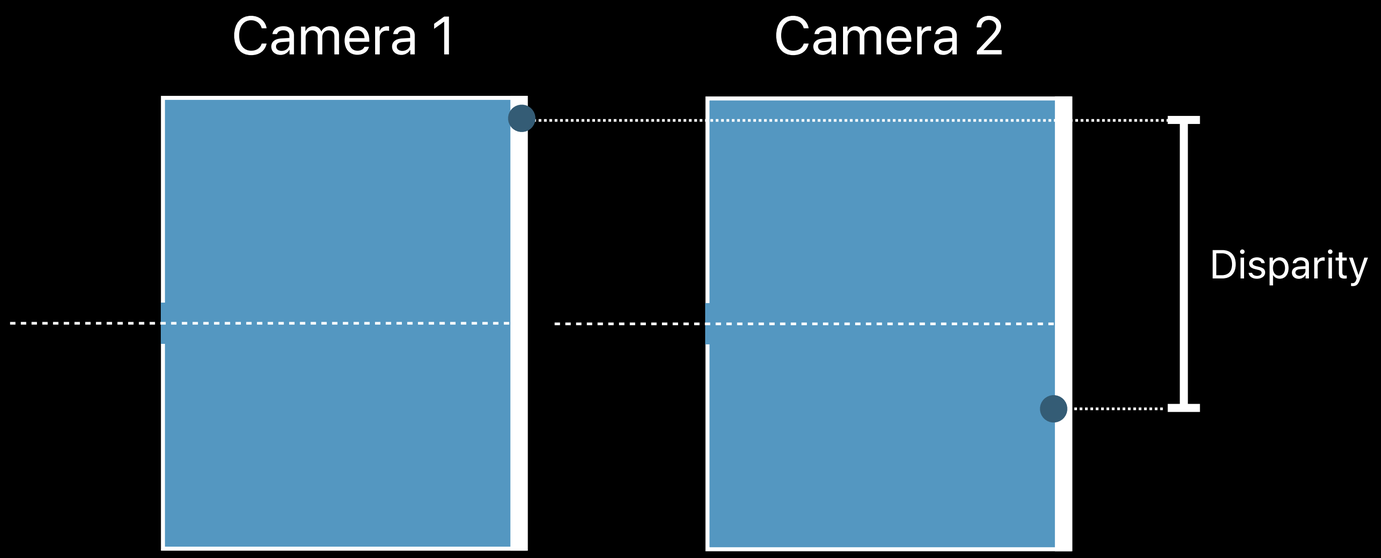
# Disparity Map



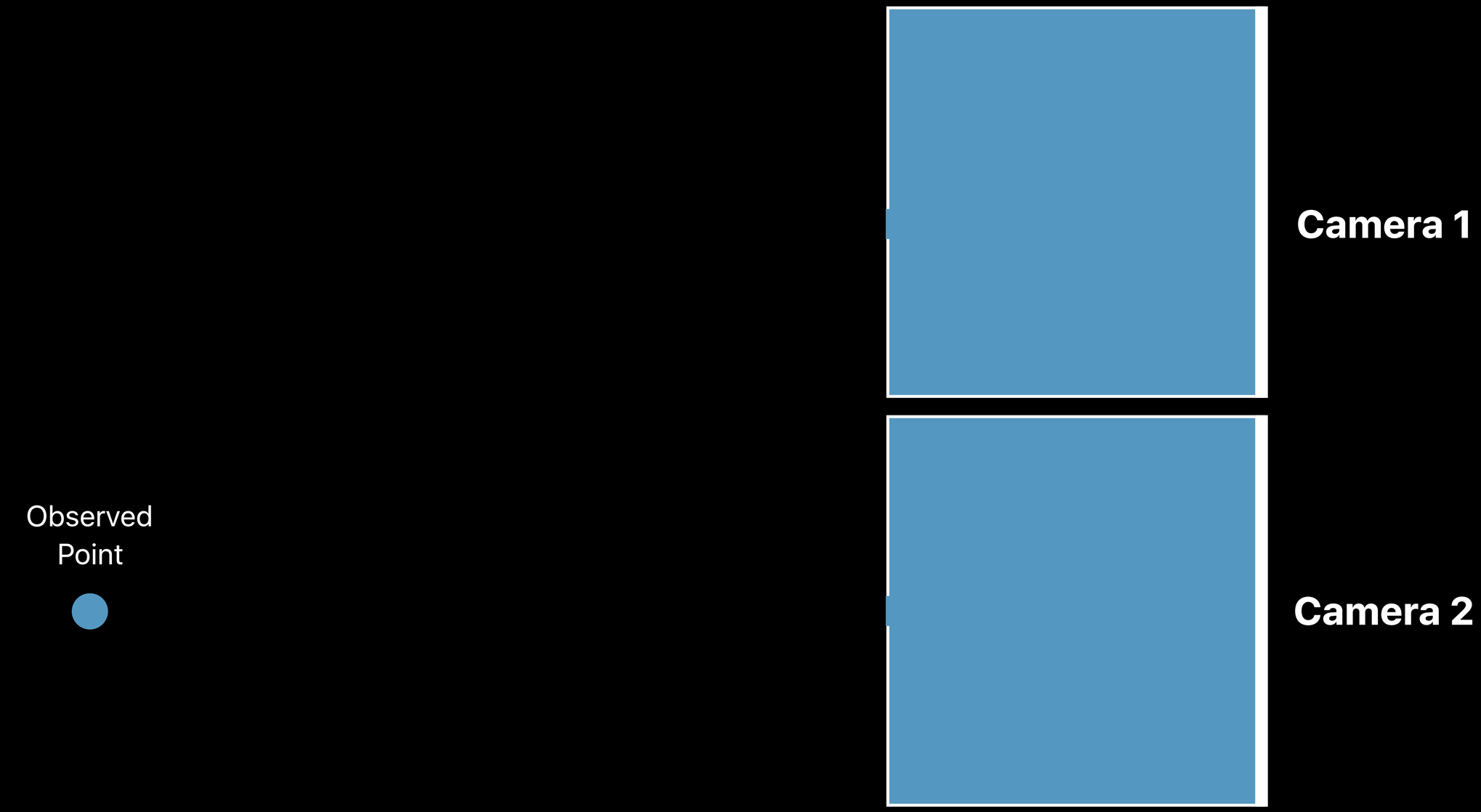
# Disparity Map



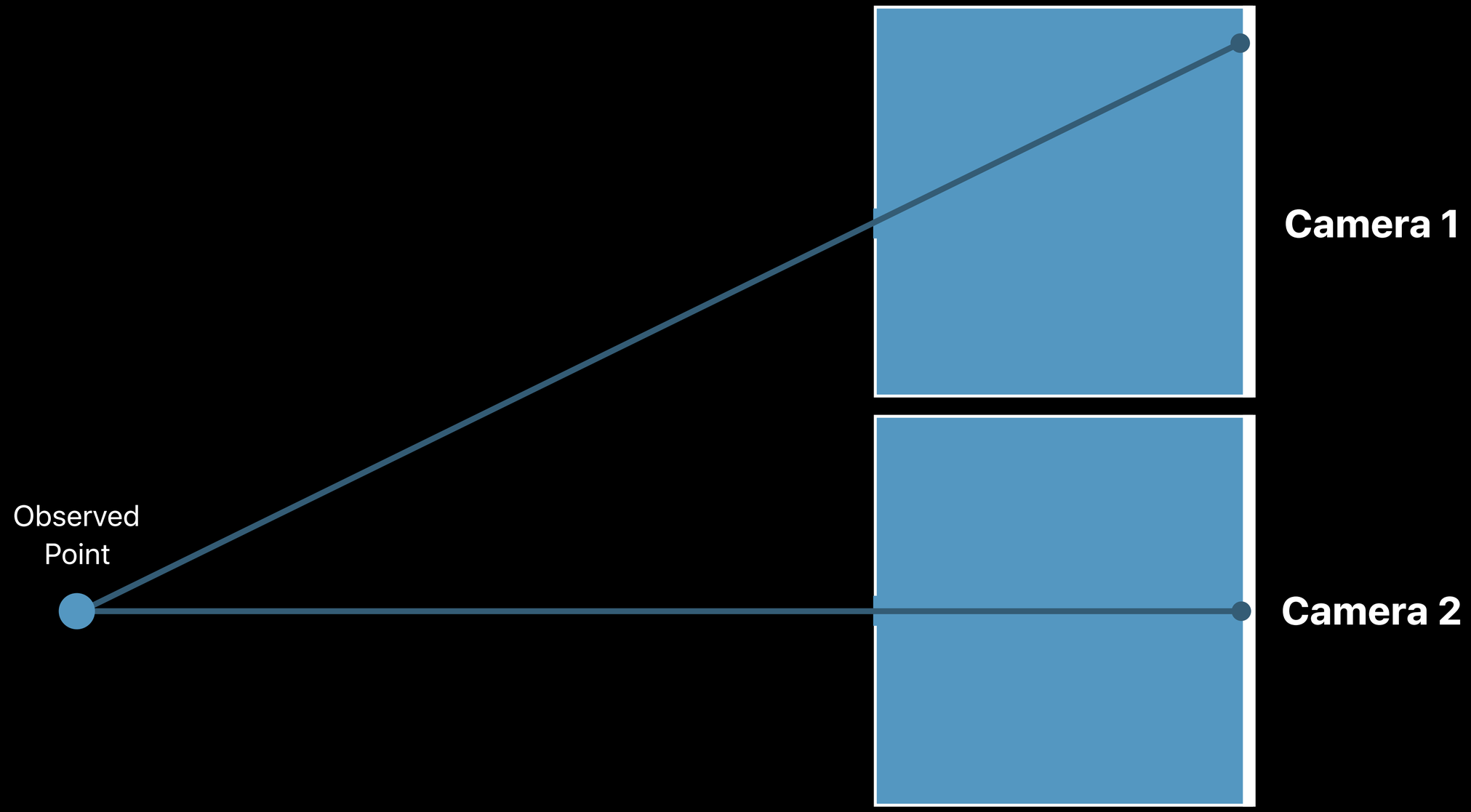
# Disparity Map



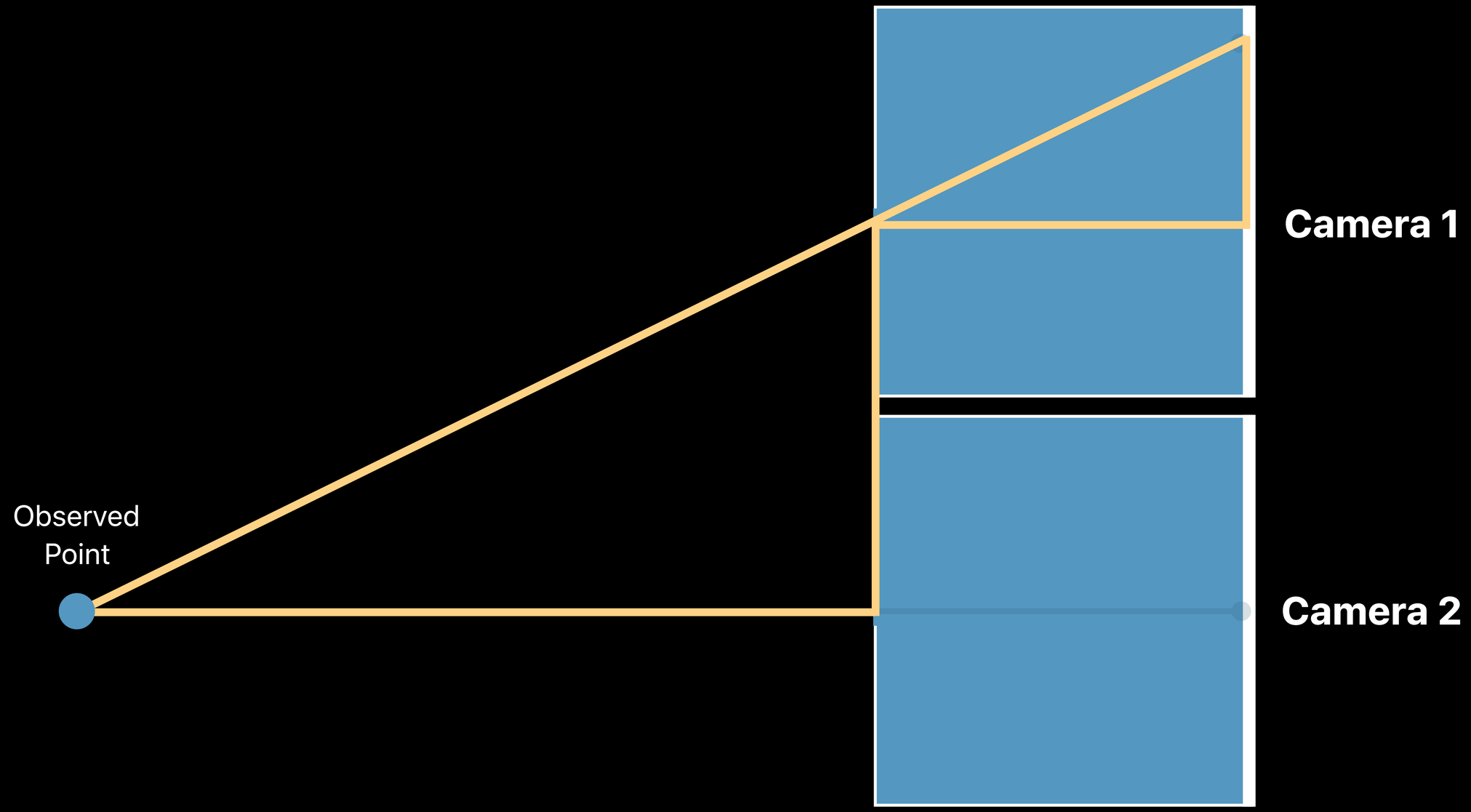
# Removing Despair from Disparity



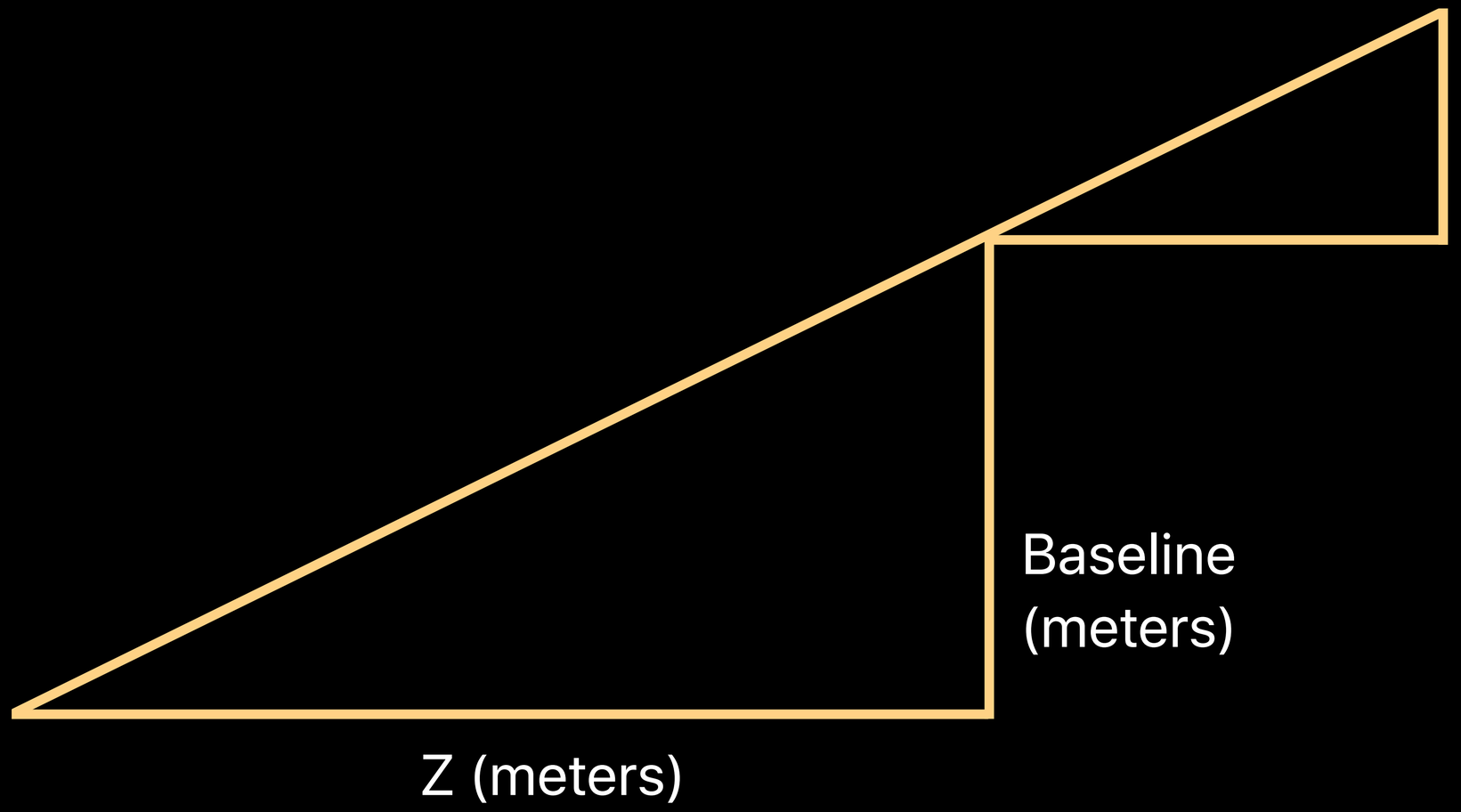
# Removing Despair from Disparity



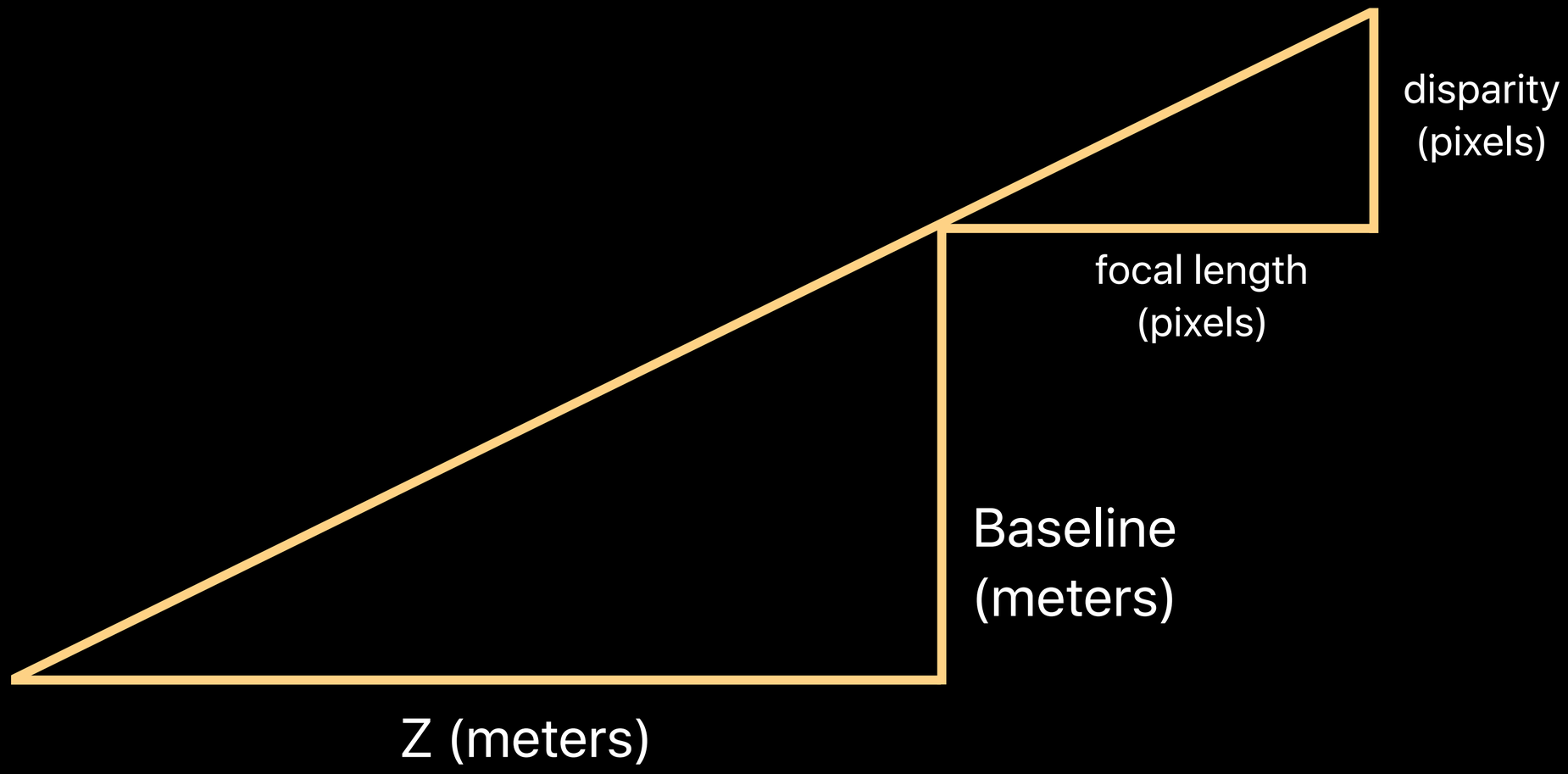
# Removing Despair from Disparity



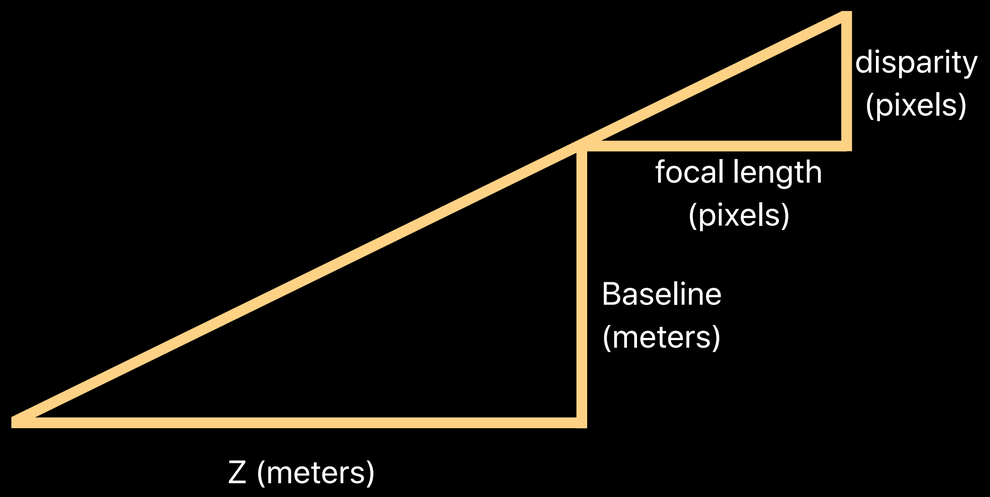
# Removing Despair from Disparity



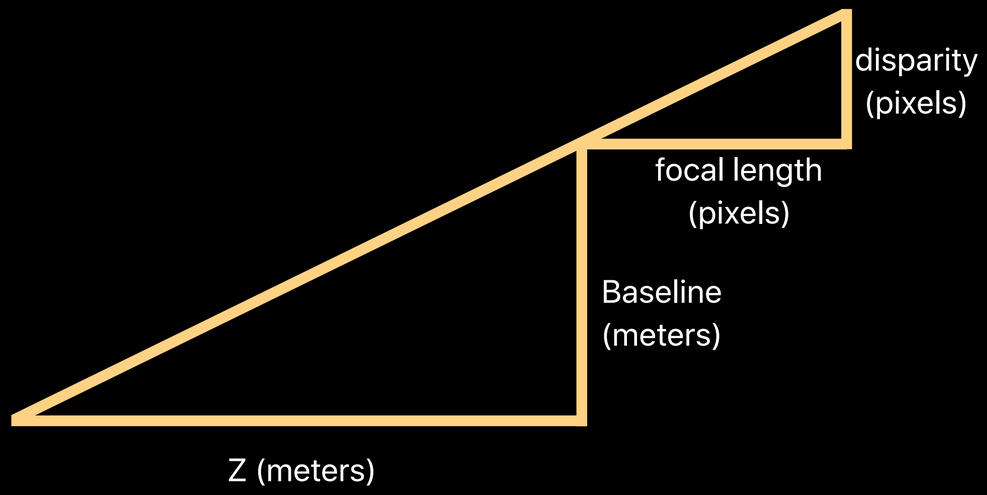
# Removing Despair from Disparity



# Removing Despair from Disparity

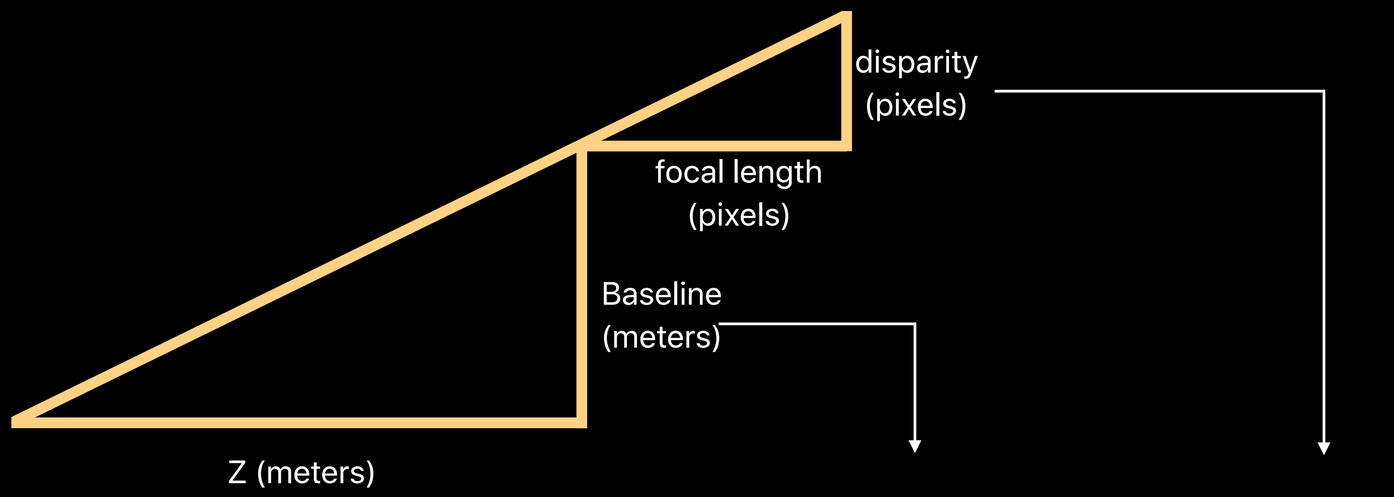


# Removing Despair from Disparity



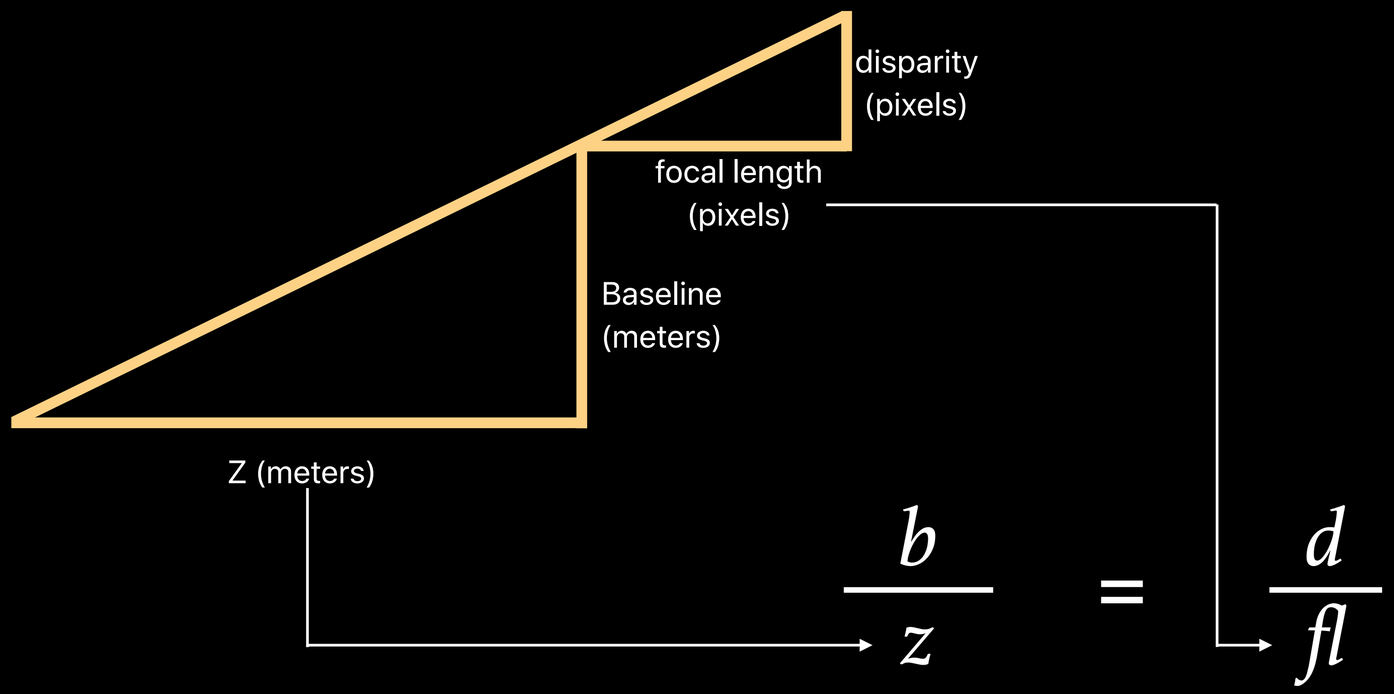

$$\frac{b}{z} = \frac{d}{fl}$$

# Removing Despair from Disparity

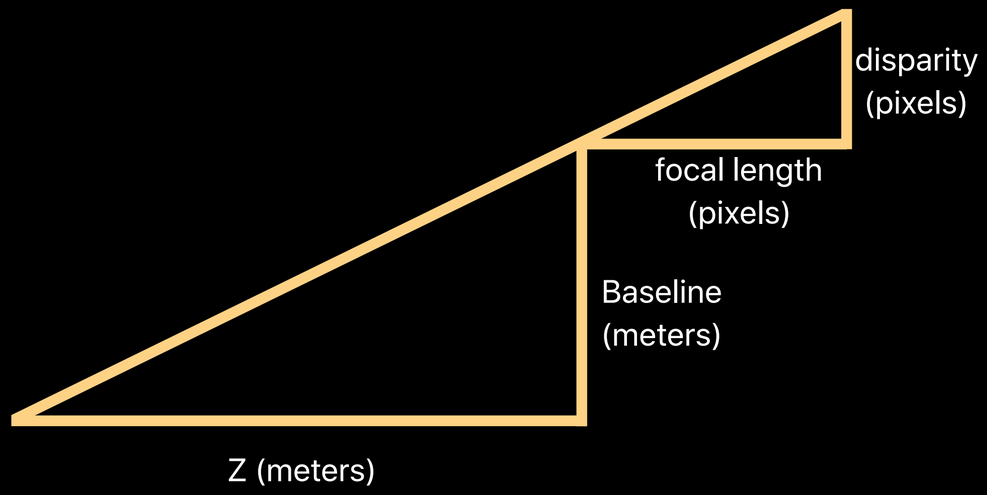



$$\frac{b}{z} = \frac{d}{fl}$$

# Removing Despair from Disparity

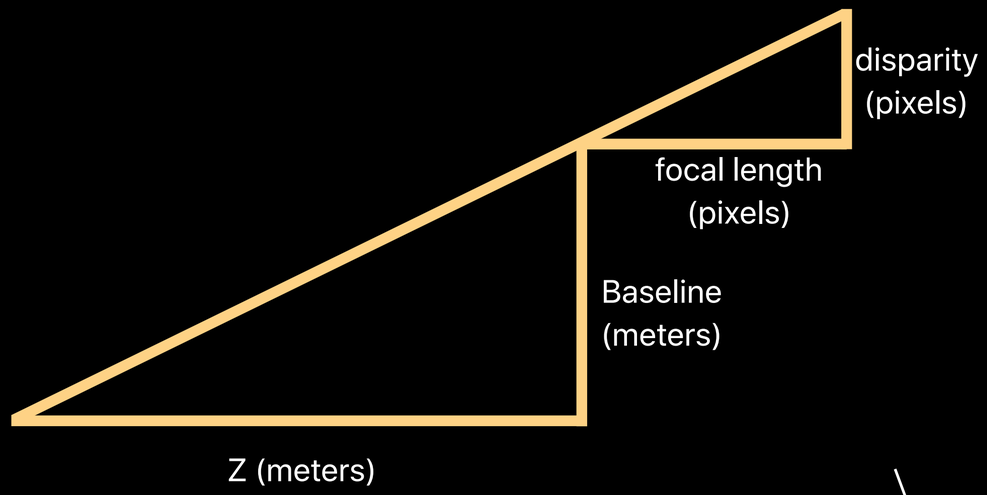


# Removing Despair from Disparity



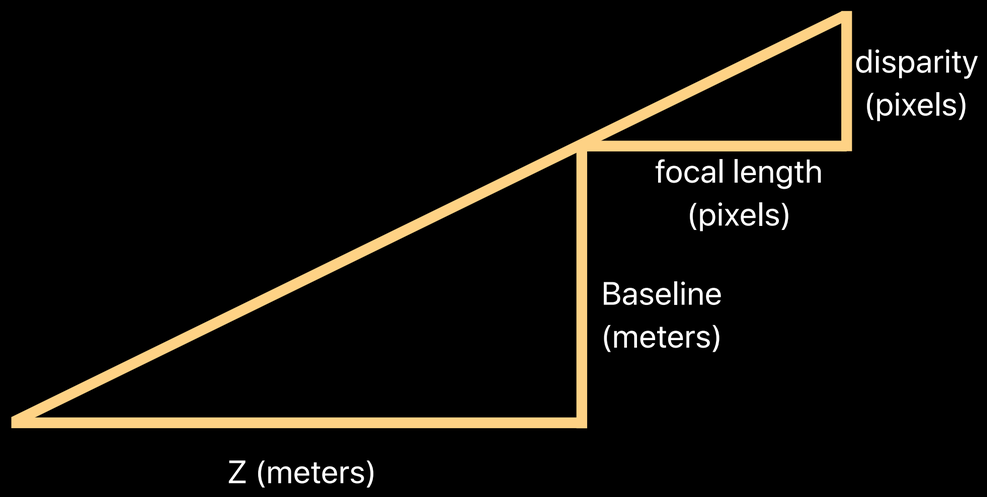

$$\frac{b}{z\ \ b} = \frac{d}{fl\ \ b}$$

# Removing Despair from Disparity



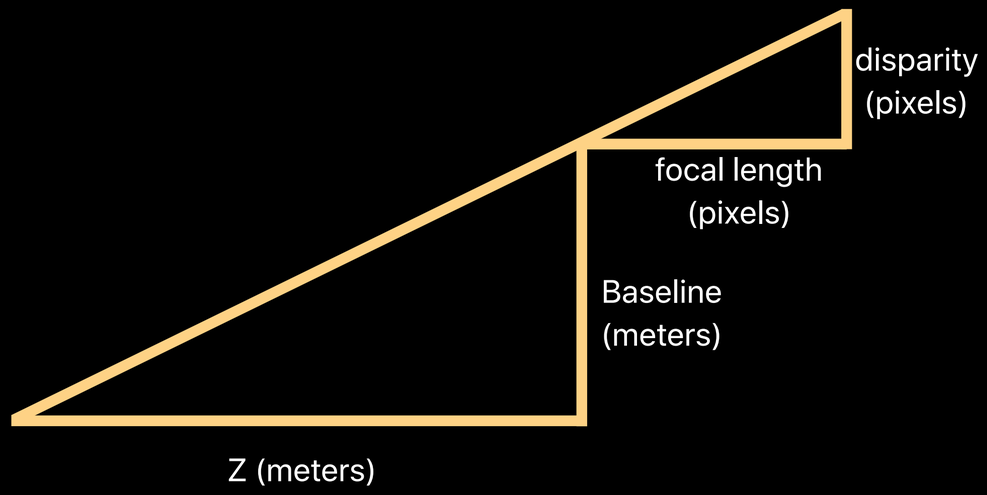

$$\frac{\cancel{b}}{z\ \cancel{b}} = \frac{d}{fl\ \ b}$$

# Removing Despair from Disparity




$$\frac{1}{z} = \frac{d}{fl\ b}$$

# Normalized Disparity




$$\frac{1}{z} = \frac{d}{fl\ b}$$

# 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

NEW

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

```
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

```
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

```
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 }

}
```

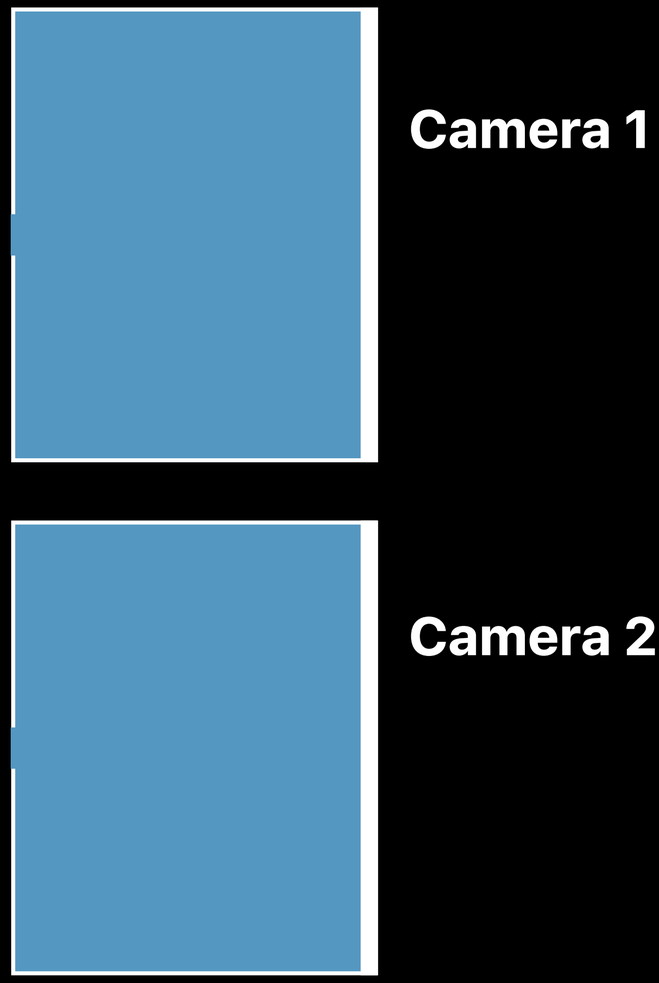
# Introducing AVDepthData

```
@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 }

}
```

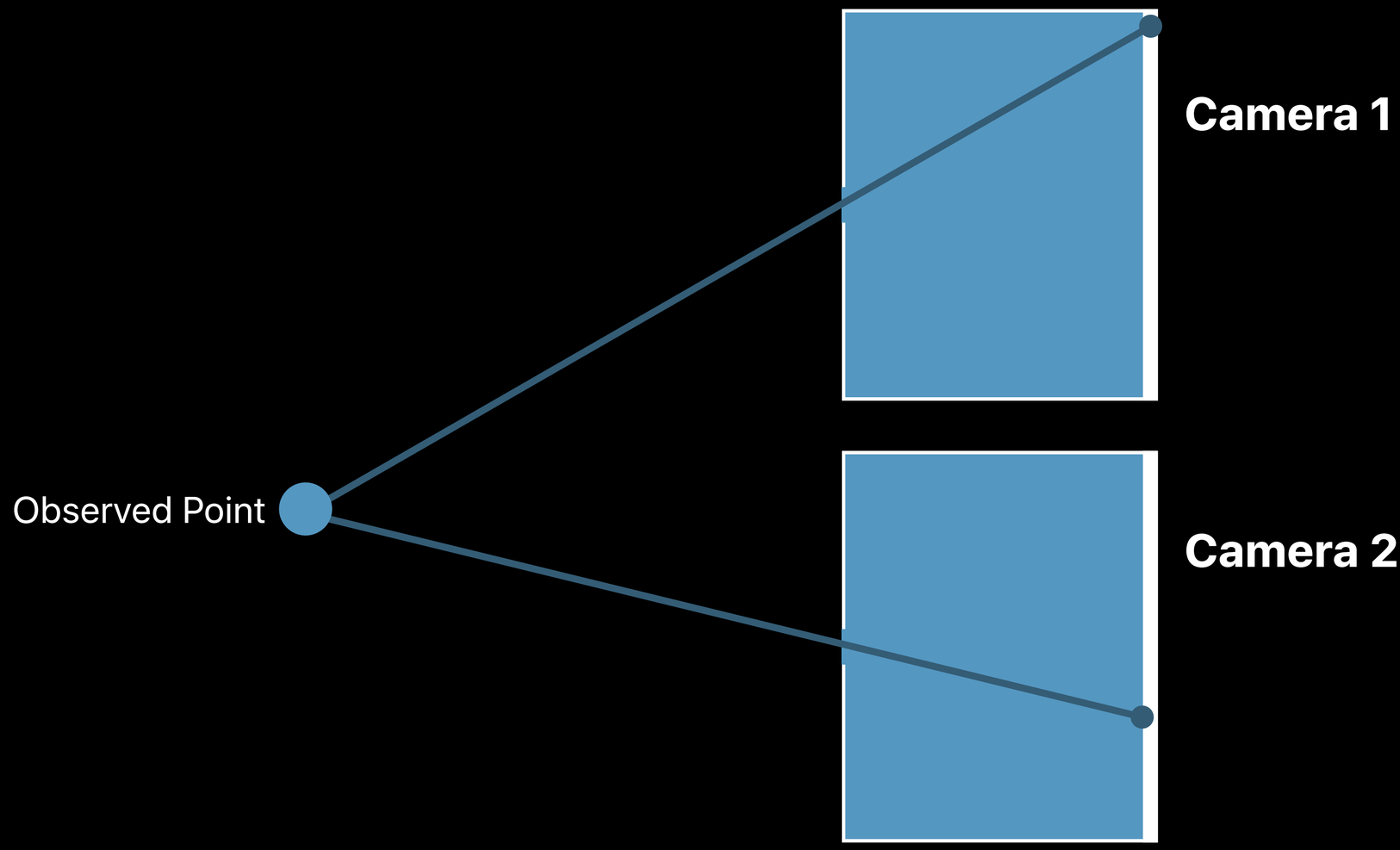
# Introducing AVDepthData

```
@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 }

}
```

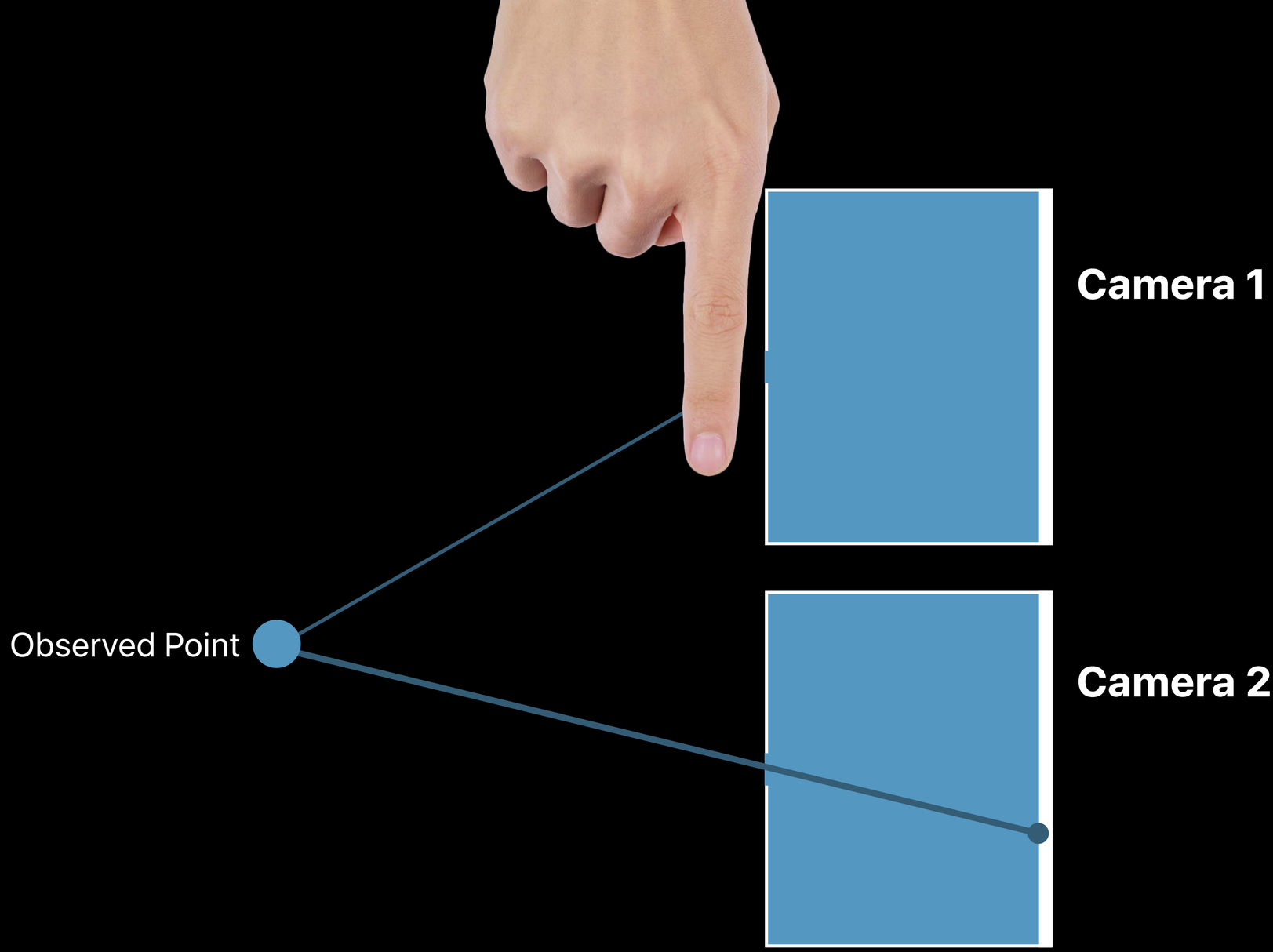
# Holes



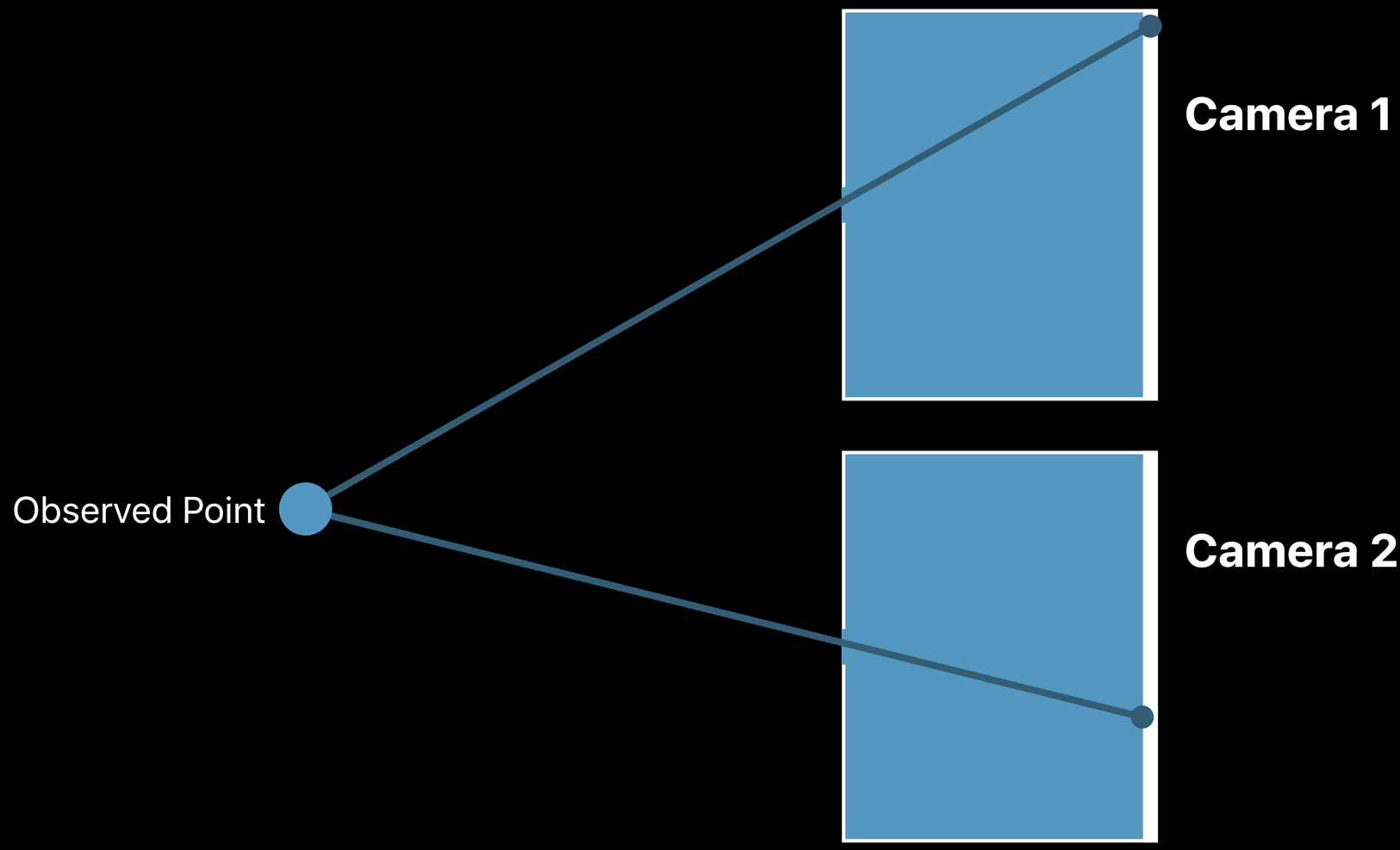
# Holes



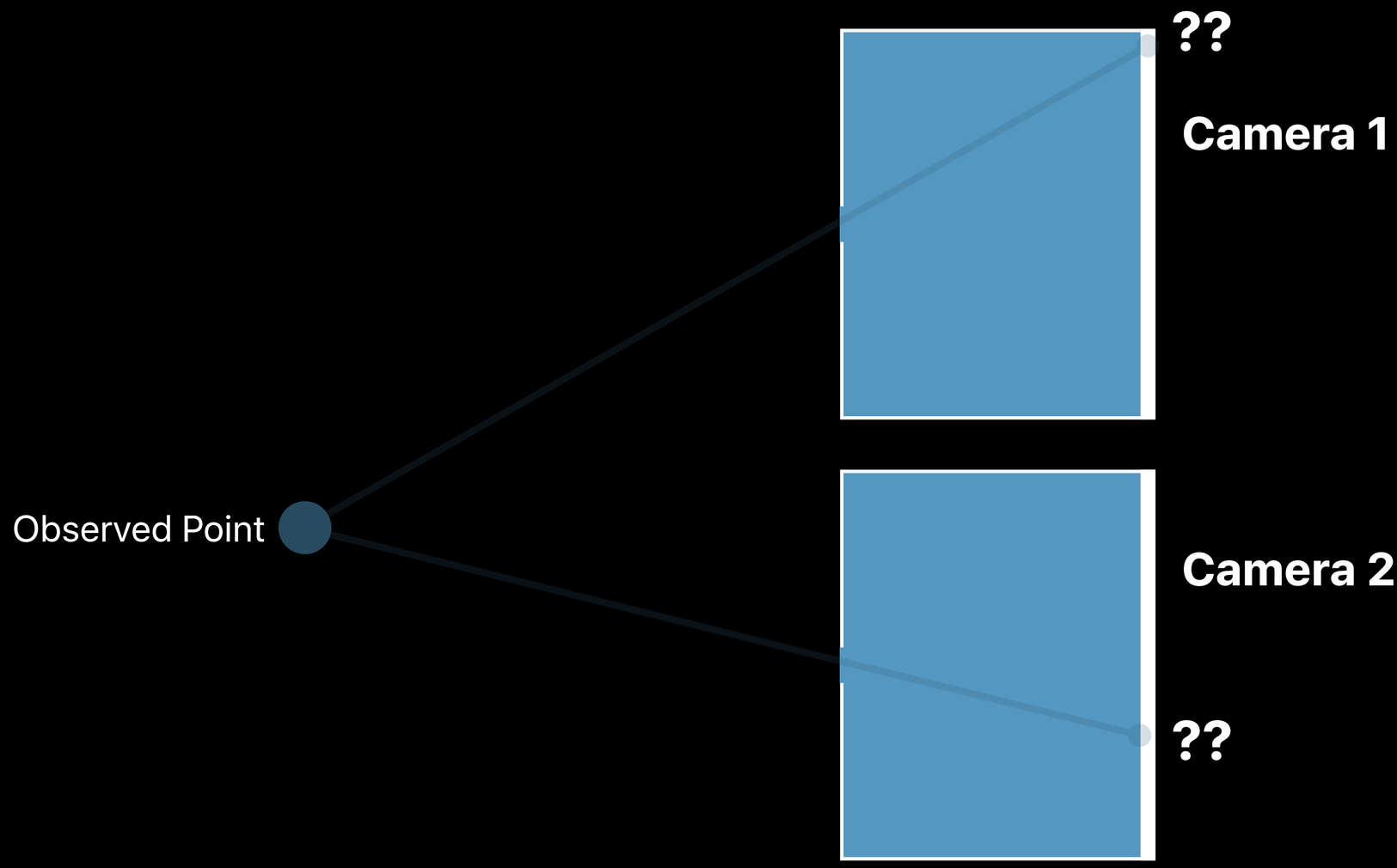
# Holes



# Holes



# Holes

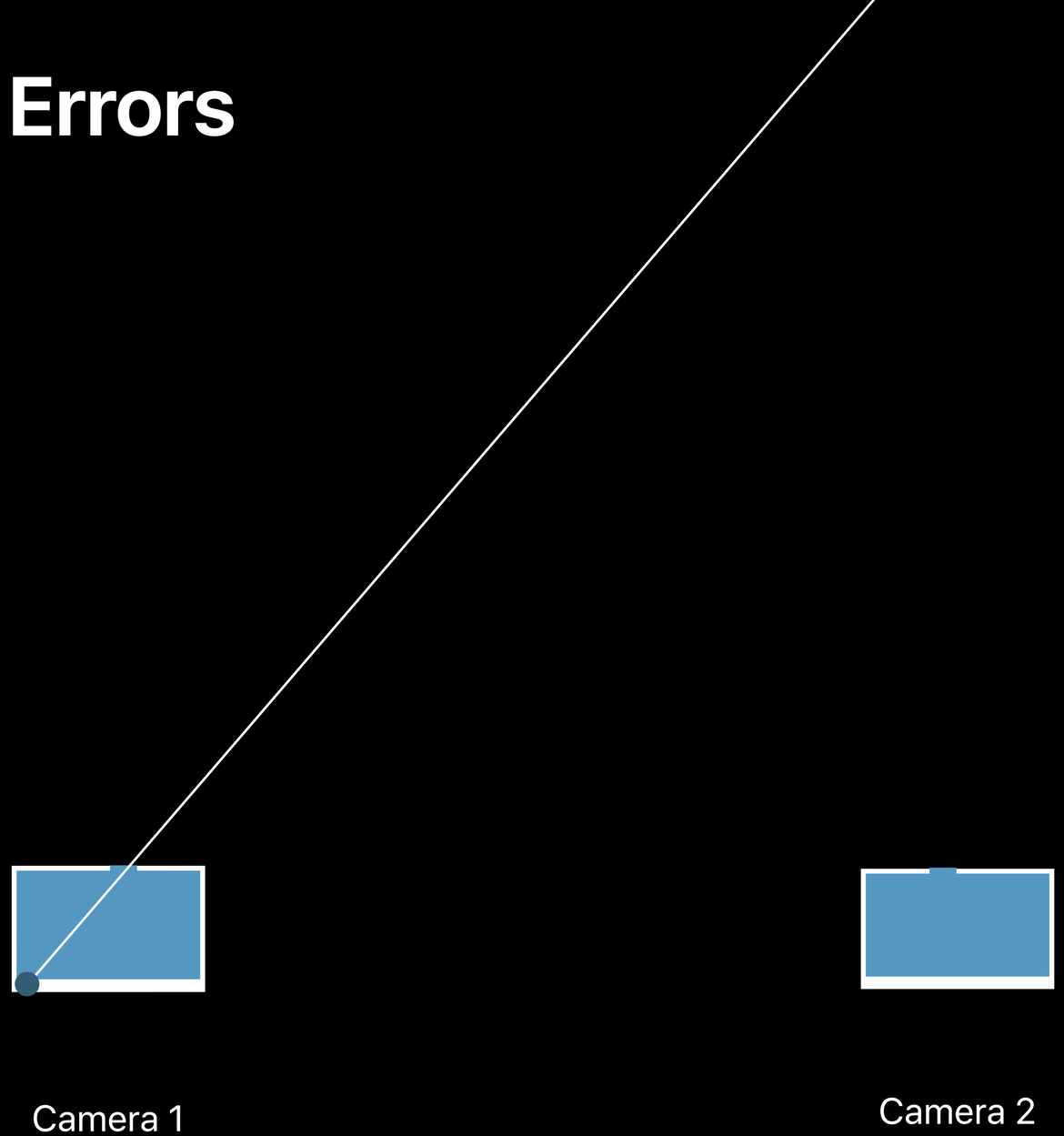


# Hole-Reporting in AVDepthData

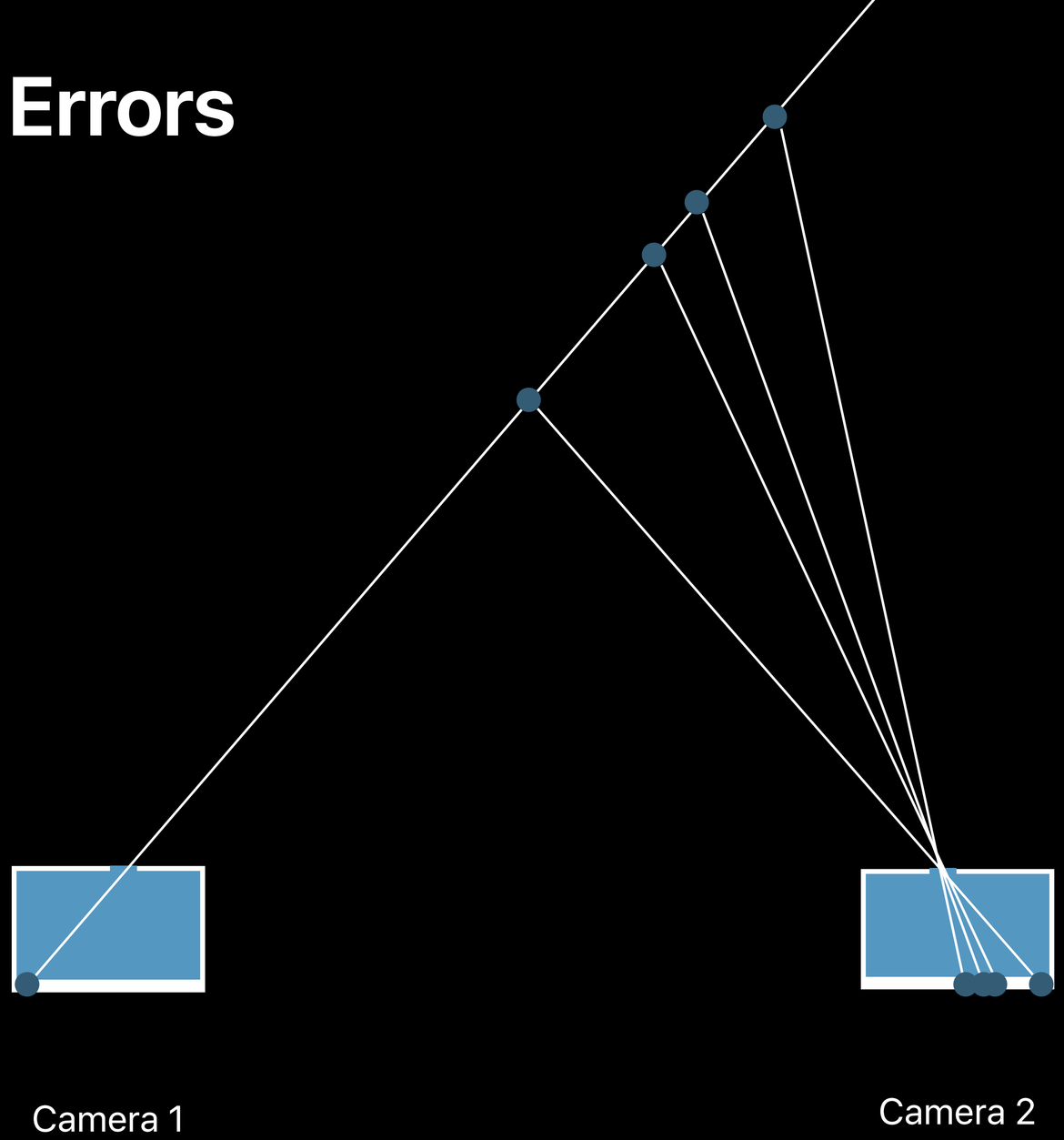
NaN = Invalid depthDataMap value

```
open var isDepthDataFiltered: Bool { get }
```

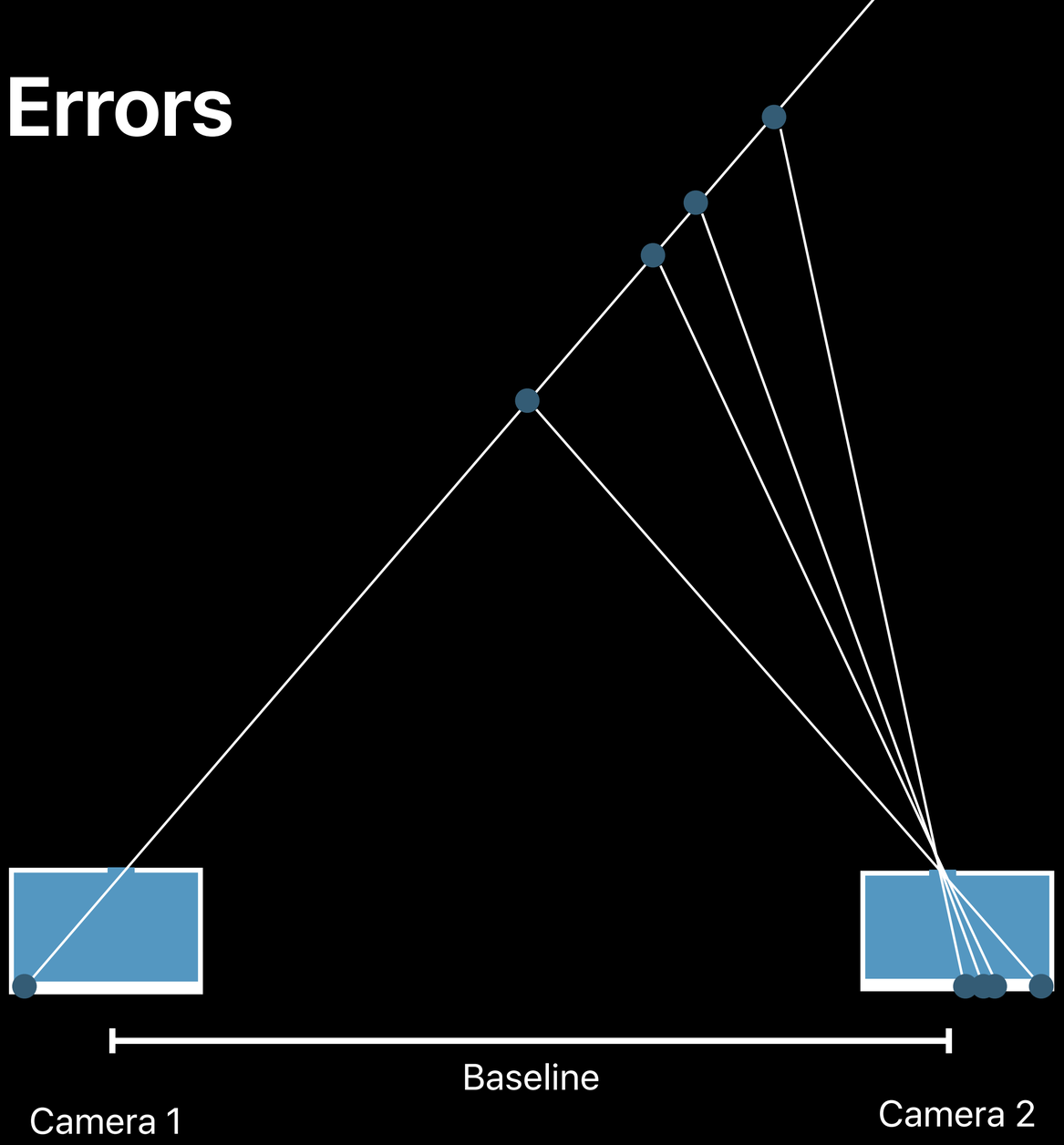
# Calibration Errors



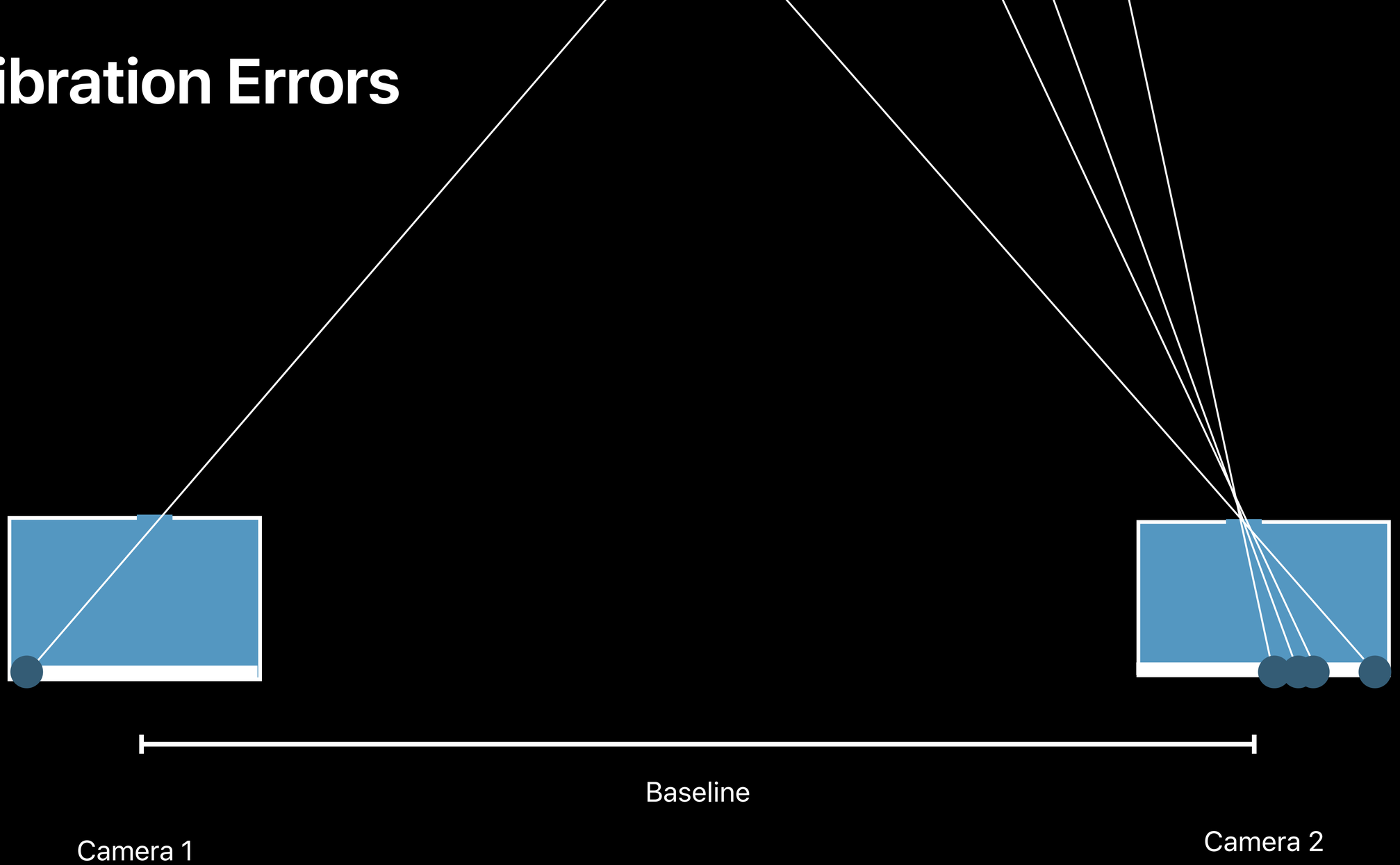
# Calibration Errors



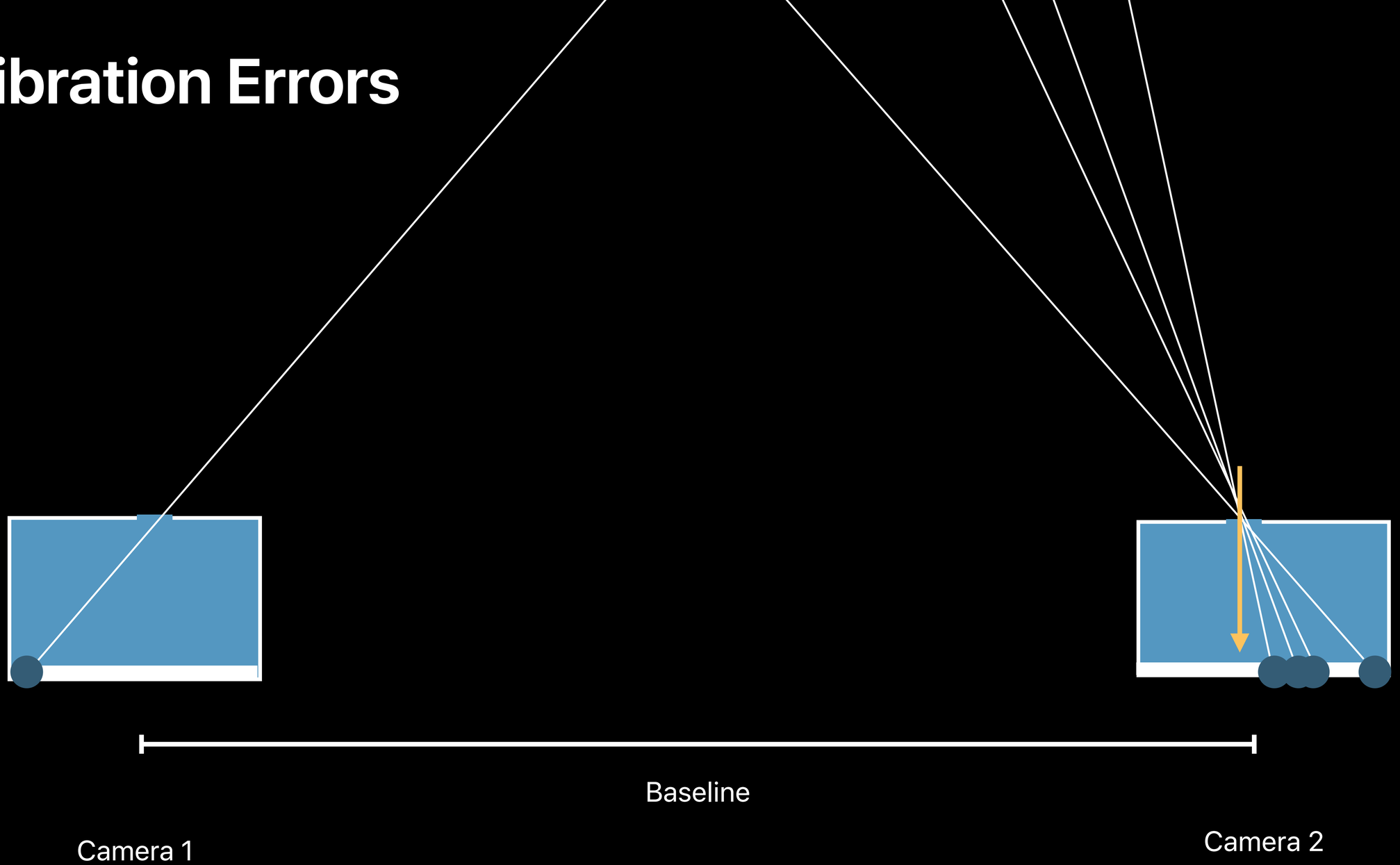
# Calibration Errors



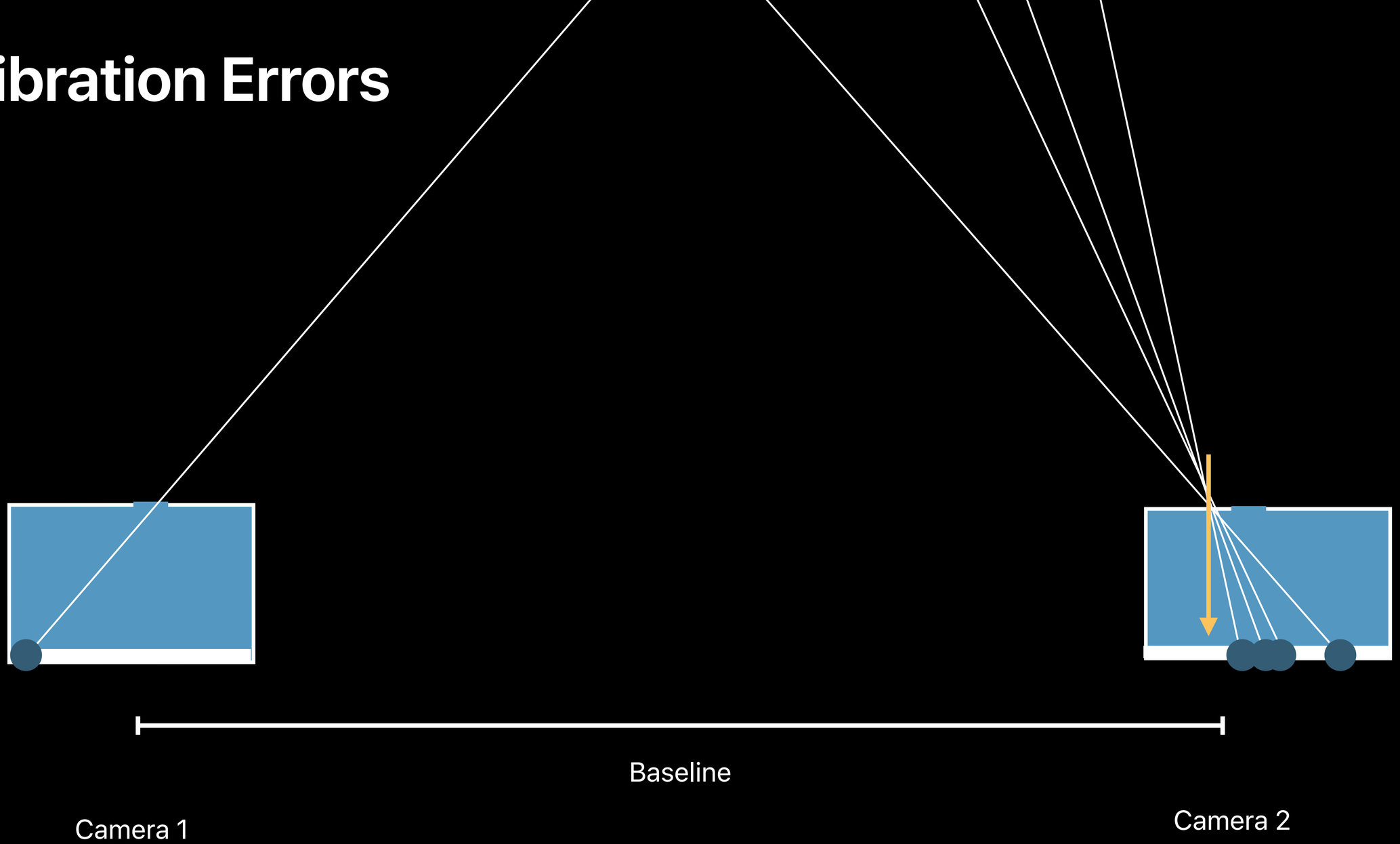
# Calibration Errors



# Calibration Errors



# Calibration Errors



# Calibration Errors

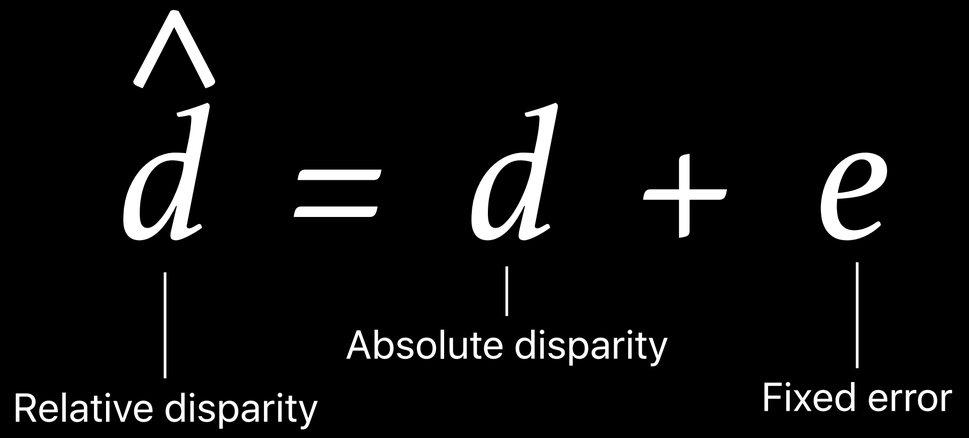
Optical Image Stabilization

Gravity

Focus Coil

# Depth Data Accuracy

```
extension AVDepthData {
    public enum Accuracy: Int {
        case relative
        case absolute
    }
}
```


$\hat{d} = d + e$
Relative disparity
Absolute disparity
Fixed error

$$\hat{d}_1 - \hat{d}_2 = (d_1 + e) - (d_2 + e)$$

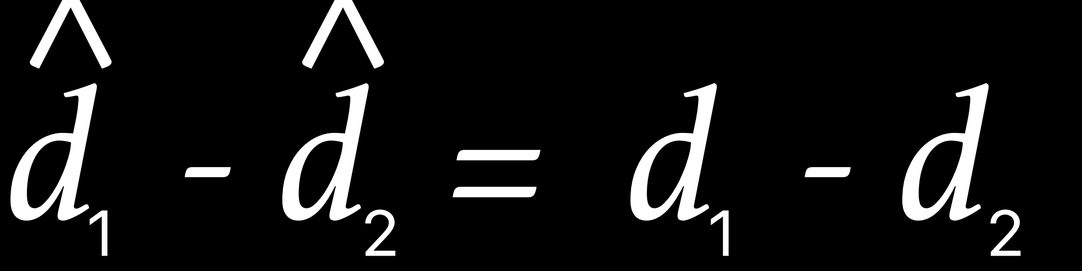
$$\hat{d}_1 - \hat{d}_2 = (d_1 + e) - (d_2 + e)$$

$$\hat{d}_1 - \hat{d}_2 = d_1 - d_2 + e - e$$

$$\hat{d}_1 - \hat{d}_2 = d_1 - d_2 + e - e$$

$$\hat{d}_1 - \hat{d}_2 = d_1 - d_2 + \cancel{e} - \cancel{e}$$

$$\hat{d}_1 - \hat{d}_2 = d_1 - d_2$$


$\hat{d}_1 - \hat{d}_2 = d_1 - d_2$

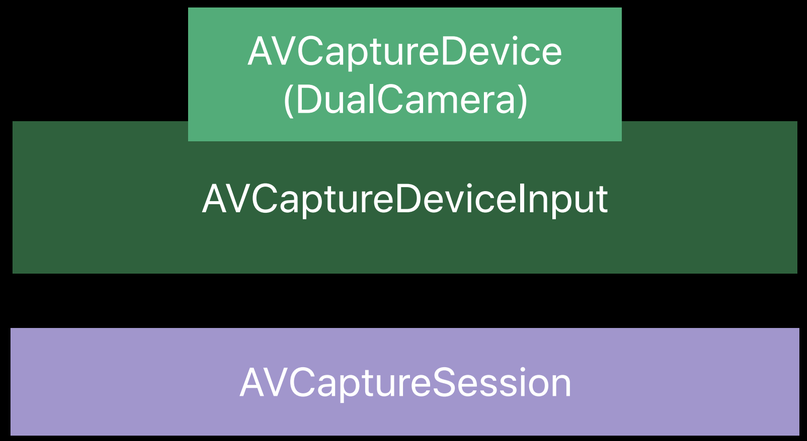
# Streaming Depth Data

# *Demo*

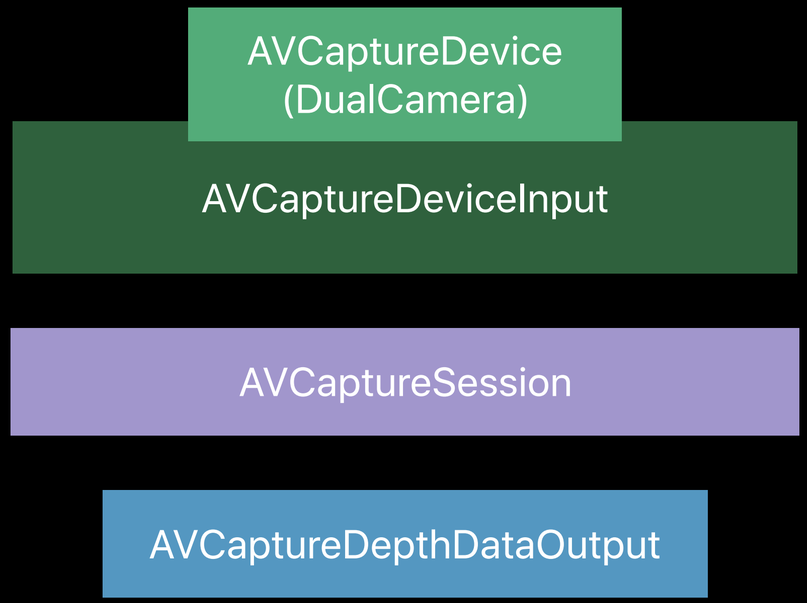
AVCamPhotoFilter

# Introducing AVCaptureDepthDataOutput

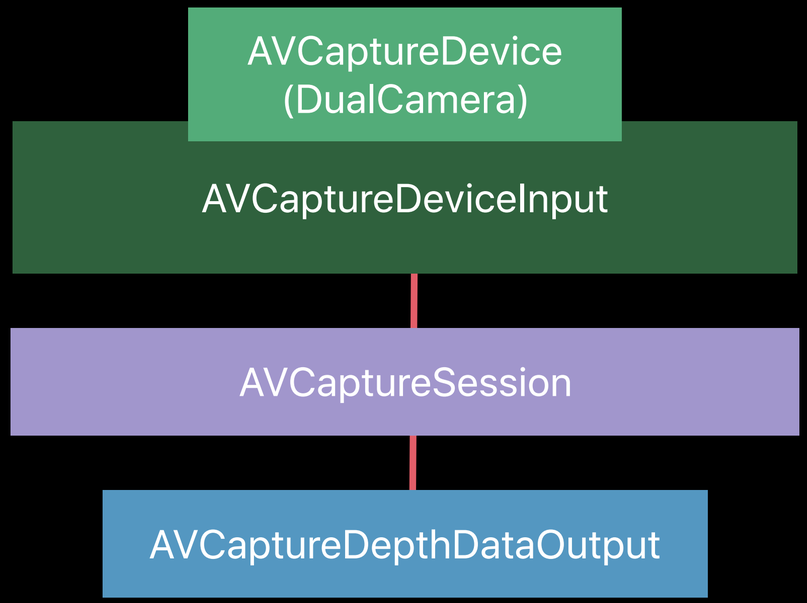
# Introducing AVCaptureDepthDataOutput



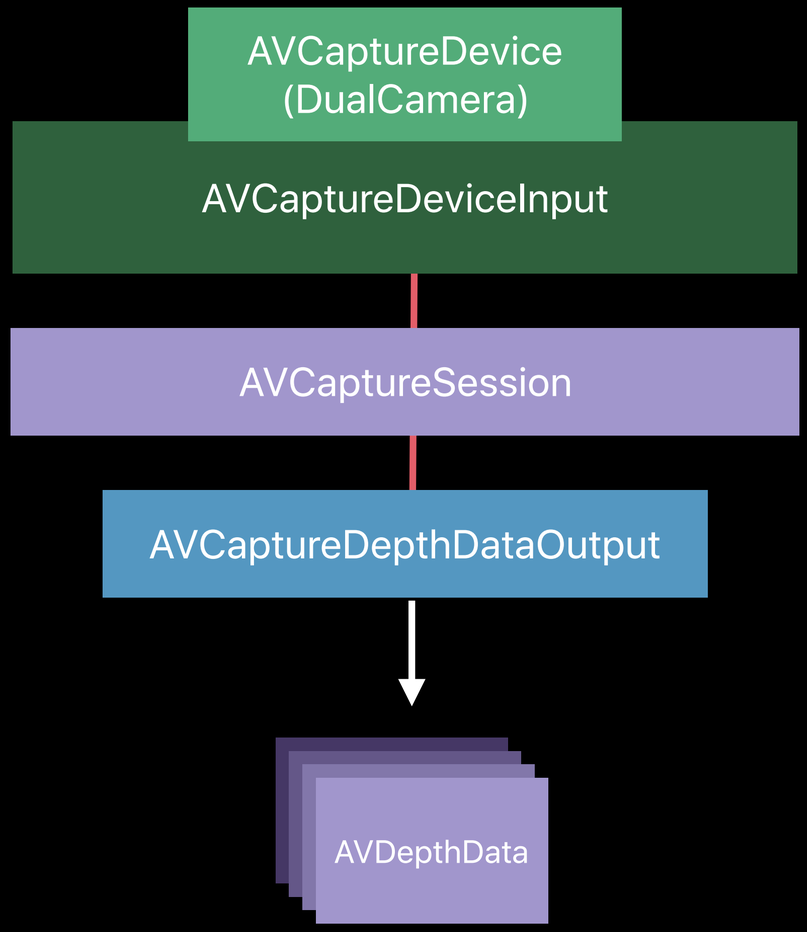
# Introducing AVCaptureDepthDataOutput



# Introducing AVCaptureDepthDataOutput



# Introducing AVCaptureDepthDataOutput



# AVCaptureDepthDataOutput

NEW

Only supported on Dual Camera

# AVCaptureDepthDataOutput

NEW

Only supported on Dual Camera

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

# AVCaptureDepthDataOutput

NEW

Only supported on Dual Camera

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

Dual Camera video formats have a `supportedDepthDataFormats` property

# AVCaptureDepthDataOutput

NEW

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

| Preset or Active Format | AVCaptureVideoDataOutput | AVCaptureDepthDataOutput |
|---|---|---|
| .photo | 1440x1080 @ 30 | 320x240 @ 24 fps<br>160x120 @ 24 fps |
| 1280x720@3-30 FPS | 1280x720 @ 30 | 320x180 @ 24 fps<br>160x90 @ 24 fps |
| .vga640x480 | 640x480 @ 30 | 320x240 @ 24 fps<br>160x120 @ 24 fps |

# Supported Depth Resolutions for Streaming

| Preset or Active Format | AVCaptureVideoDataOutput | AVCaptureDepthDataOutput |
|---|---|---|
| .photo | 1440x1080 @ 30 | 320x240 @ 24 fps<br>160x120 @ 24 fps |
| 1280x720@3-30 FPS | 1280x720 @ 30 | 320x180 @ 24 fps<br>160x90 @ 24 fps |
| .vga640x480 | 640x480 @ 30 | 320x240 @ 24 fps<br>160x120 @ 24 fps |

# Supported Depth Resolutions for Streaming

| Preset or Active Format | AVCaptureVideoDataOutput | AVCaptureDepthDataOutput |
|---|---|---|
| .photo | 1440x1080 @ 30 | 320x240 @ 24 fps<br>160x120 @ 24 fps |
| 1280x720@3-30 FPS | 1280x720 @ 30 | 320x180 @ 24 fps<br>160x90 @ 24 fps |
| .vga640x480 | 640x480 @ 30 | 320x240 @ 24 fps<br>160x120 @ 24 fps |

# Supported Depth Resolutions for Streaming

| Preset or Active Format | AVCaptureVideoDataOutput | AVCaptureDepthDataOutput |
|---|---|---|
| .photo | 1440x1080 @ 30 | 320x240 @ 24 fps<br>160x120 @ 24 fps |
| 1280x720@3-30 FPS | 1280x720 @ 30 | 320x180 @ 24 fps<br>160x90 @ 24 fps |
| .vga640x480 | 640x480 @ 30 | 320x240 @ 24 fps<br>160x120 @ 24 fps |

# Depth Frame Rate Examples

| VideoDataOutput | DepthDataOutput |
| --- | --- |
| 1440x1080 @ 24 | 320x240 @ 24 fps |
| 1440x1080 @ 30 | 320x240 @ 15 fps |

# Depth Frame Rate Examples

| VideoDataOutput | DepthDataOutput |
| --- | --- |
| 1440x1080 @ 24 | 320x240 @ 24 fps |
| 1440x1080 @ 30 | 320x240 @ 15 fps |

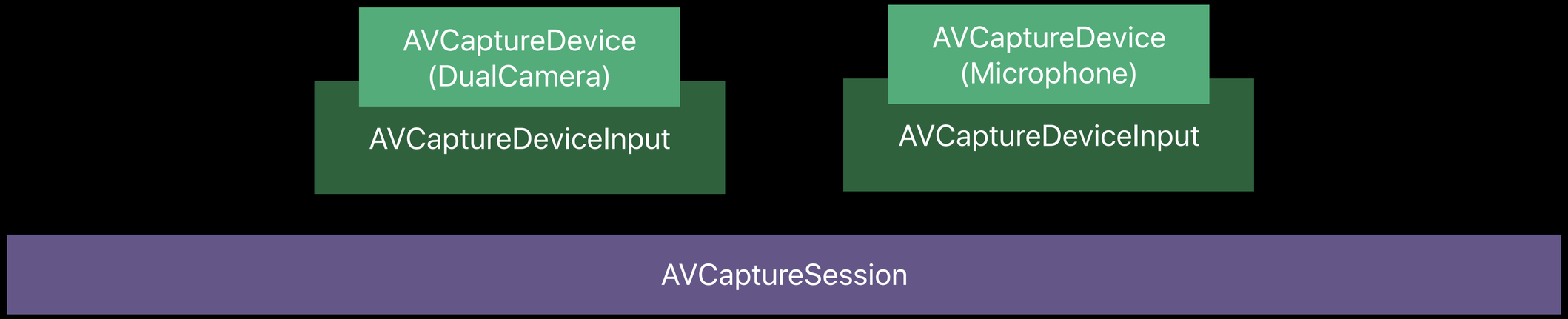
# Depth Frame Rate Examples

| VideoDataOutput | DepthDataOutput |
| --- | --- |
| 1440x1080 @ 24 | 320x240 @ 24 fps |
| 1440x1080 @ 30 | 320x240 @ 15 fps |

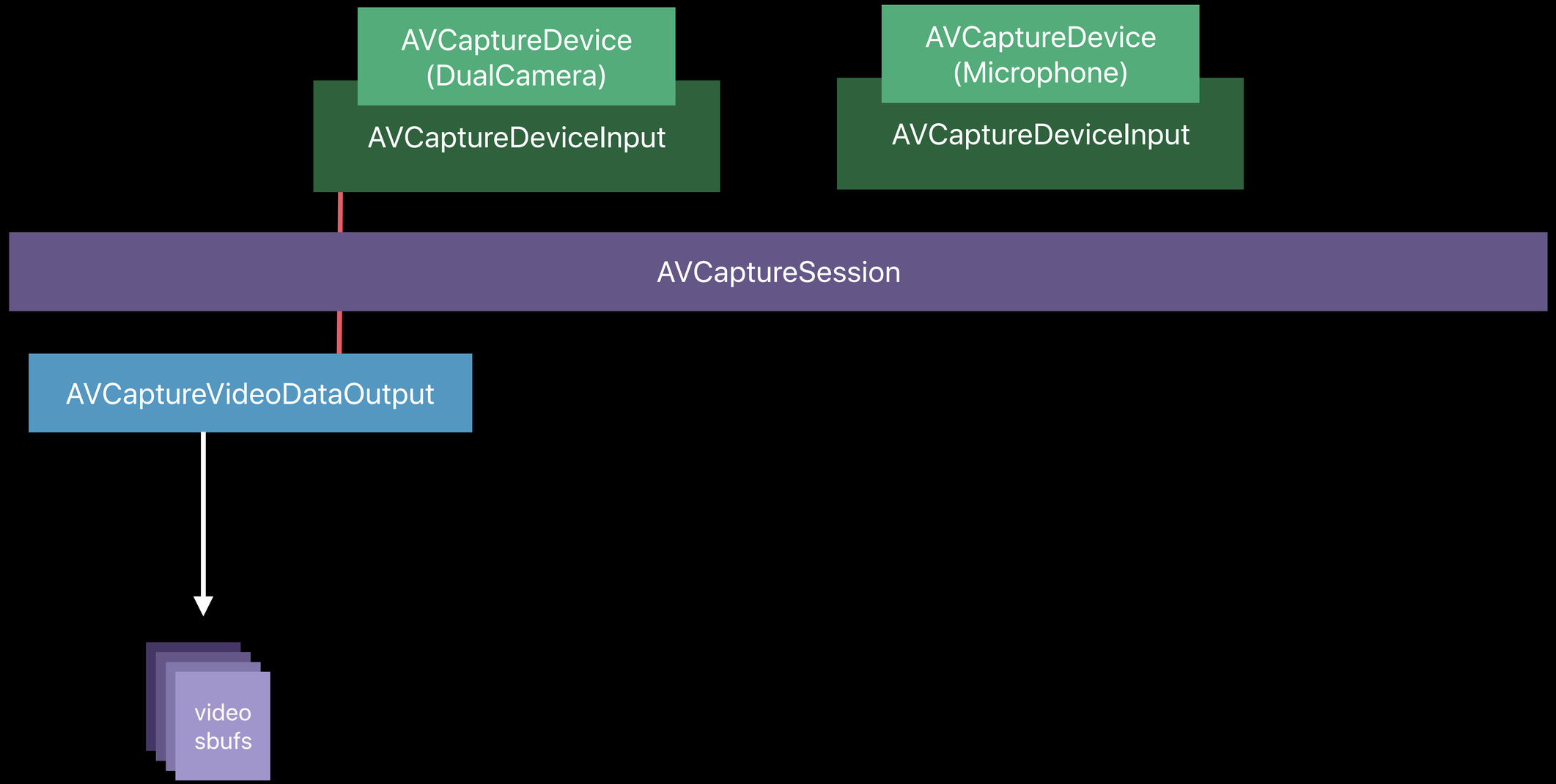
# Depth Filtering with AVCaptureDepthDataOutput

```
open var isFilteringEnabled: Bool
```

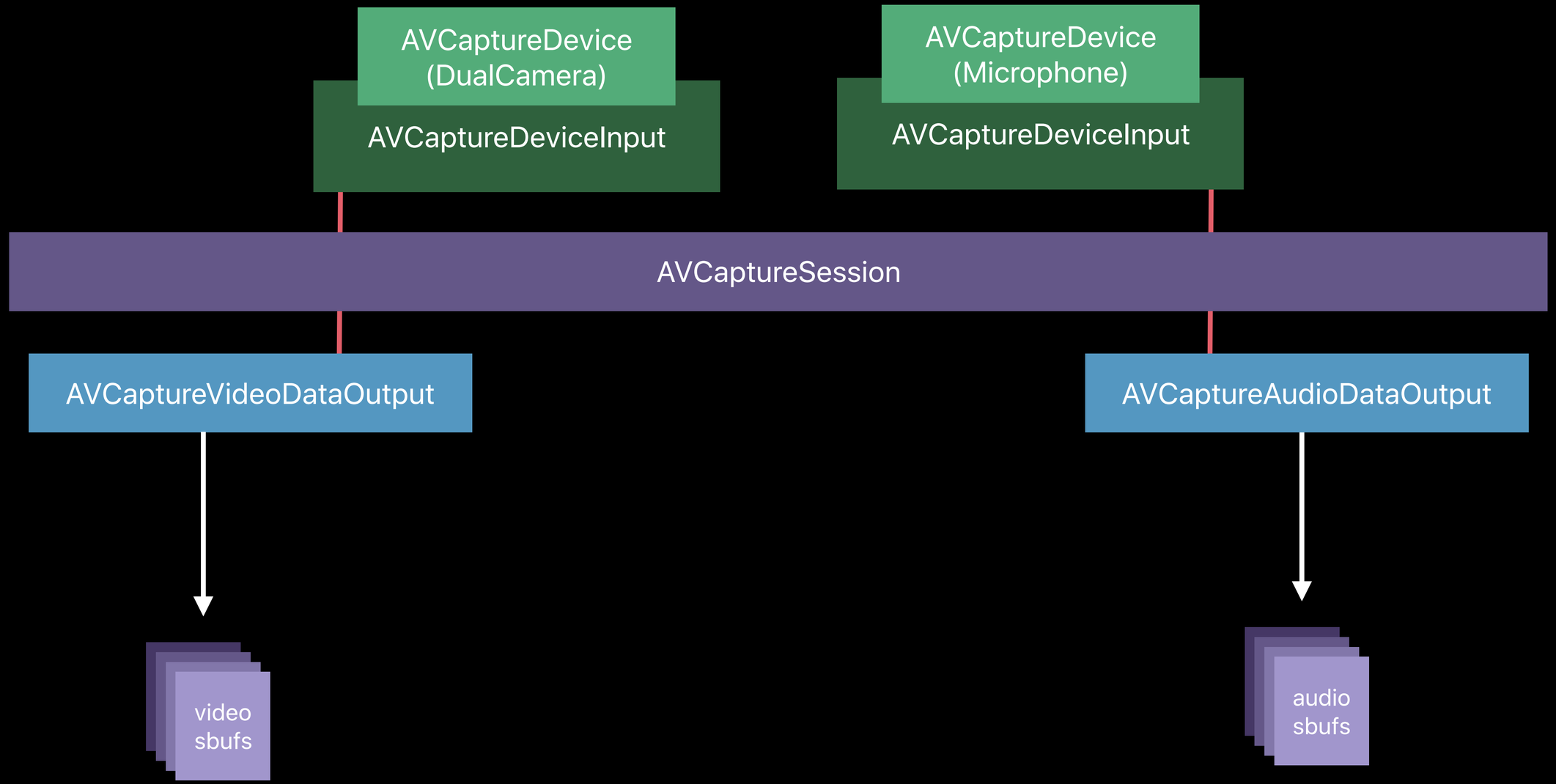
# Unsynchronized Data Output



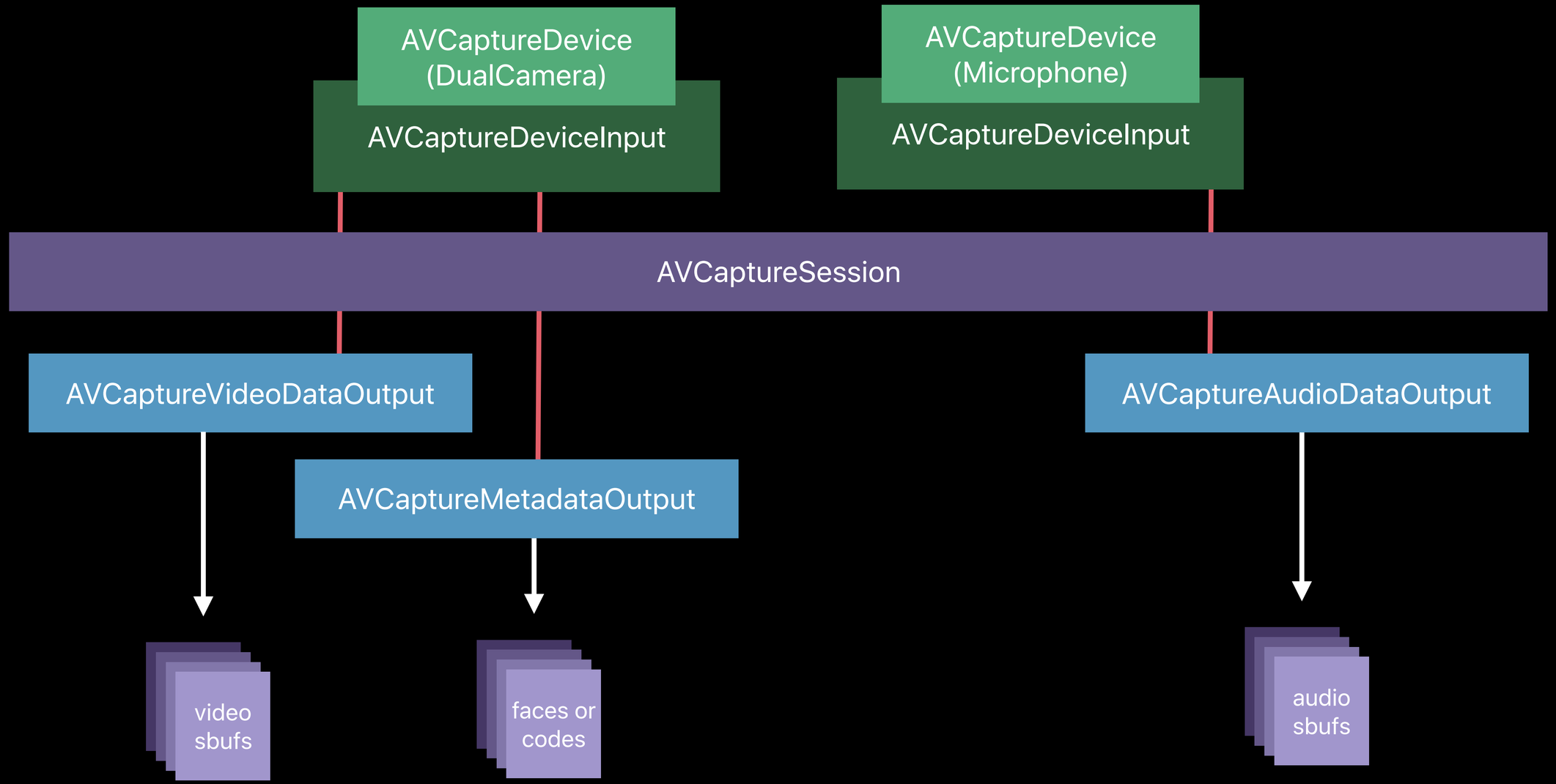
# Unsynchronized Data Output



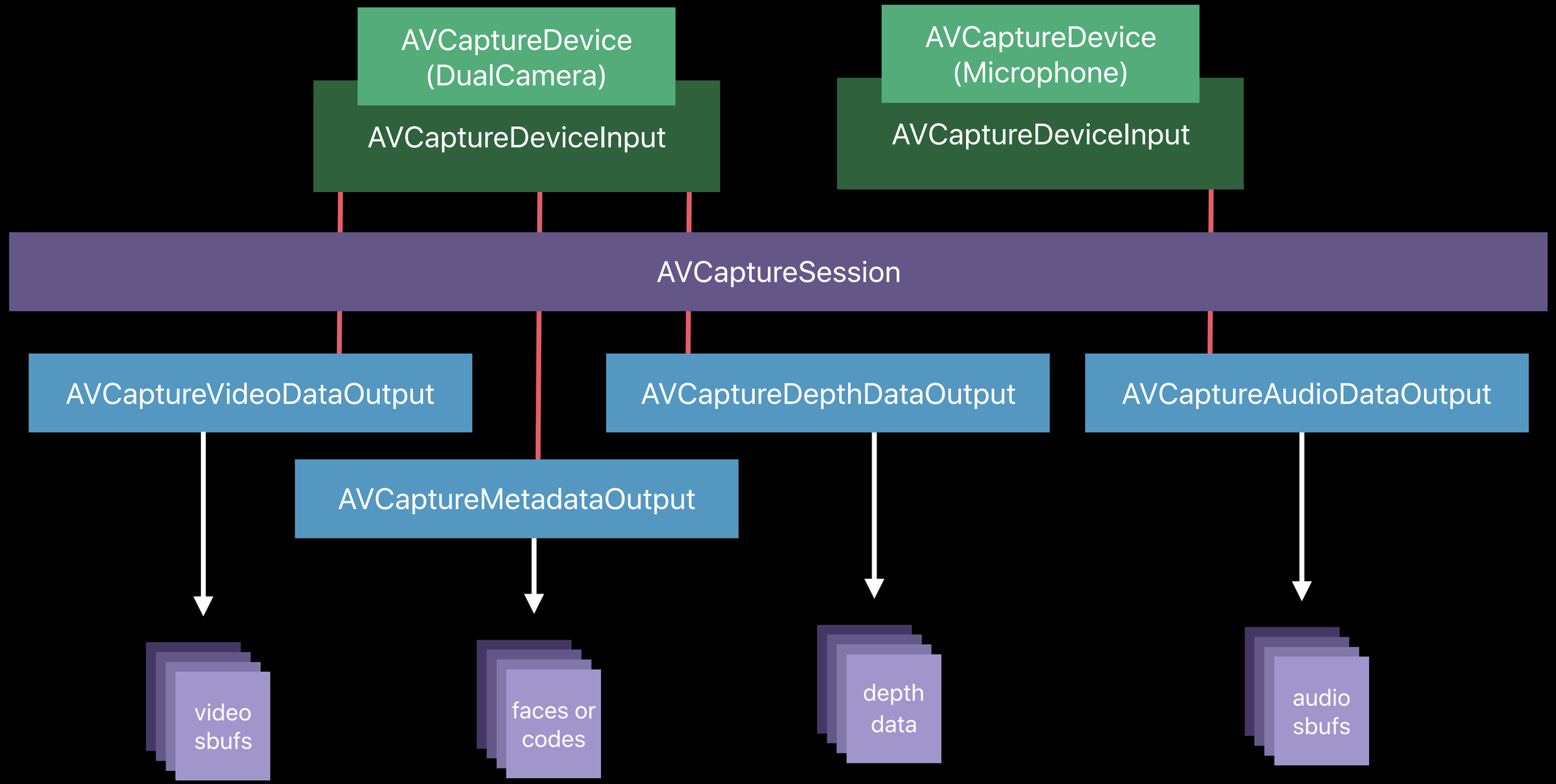
# Unsynchronized Data Output



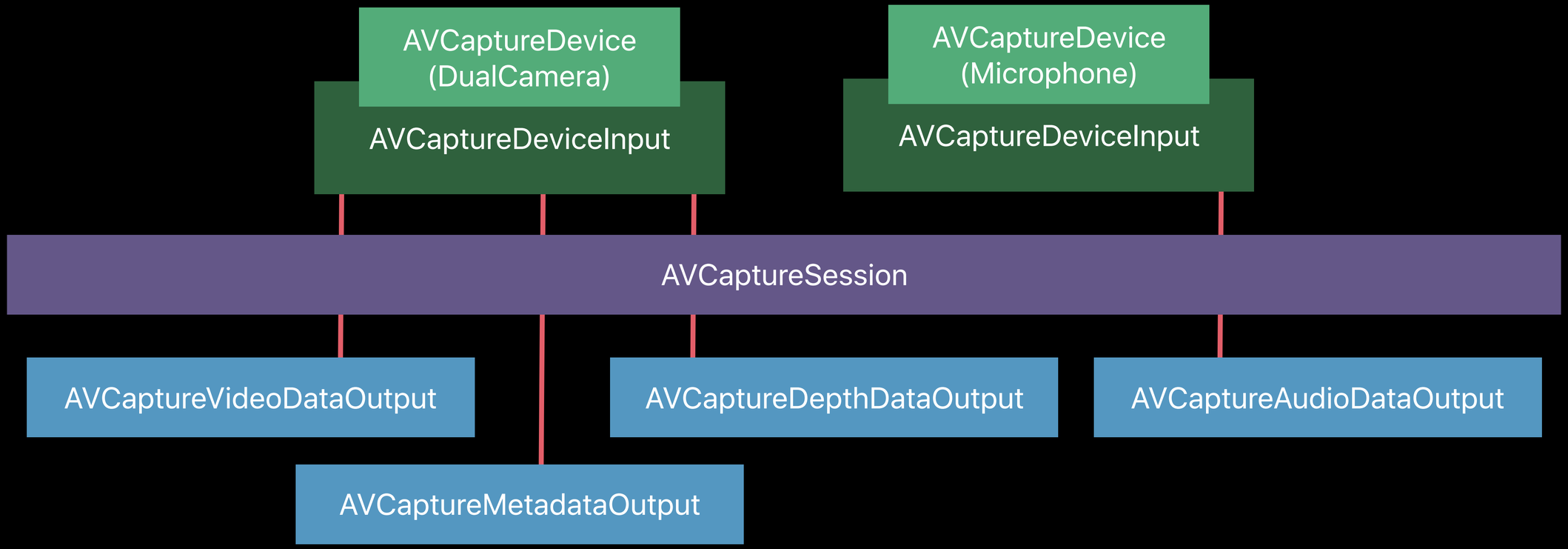
# Unsynchronized Data Output



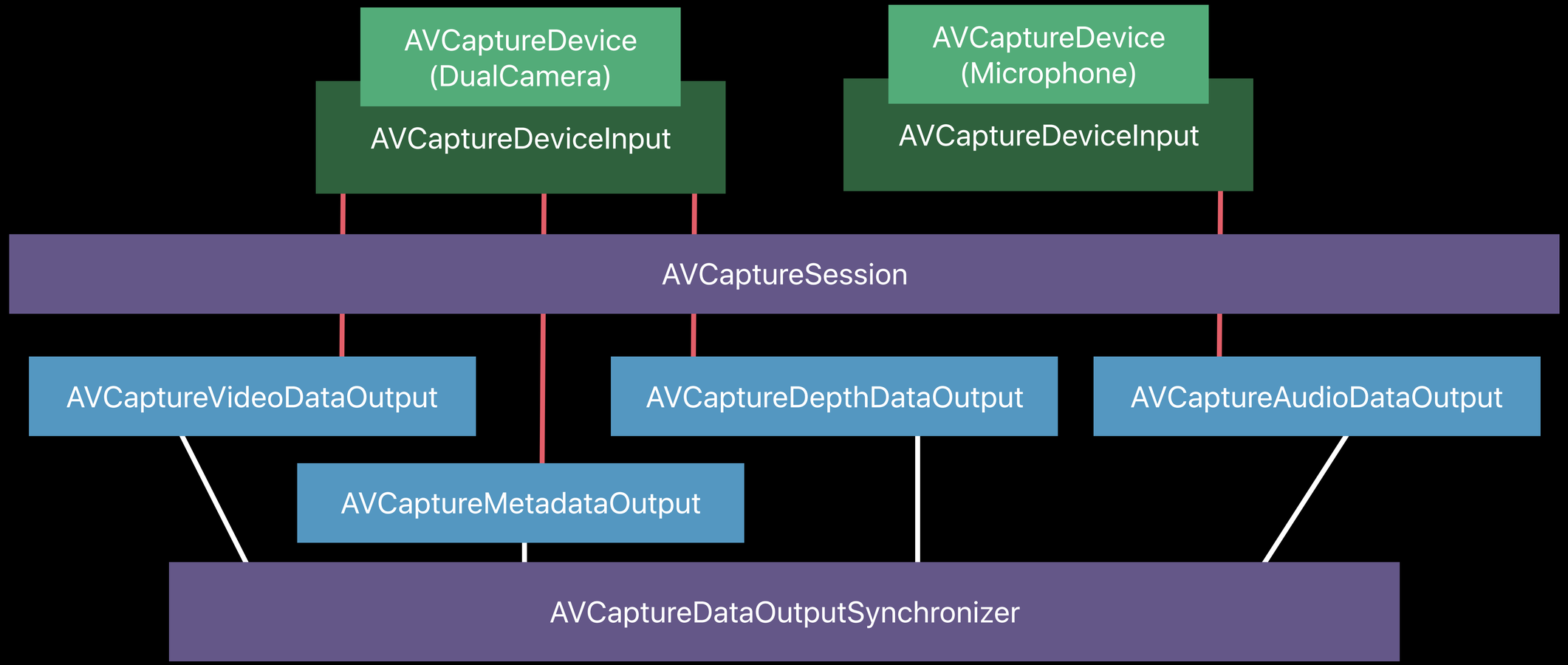
# Unsynchronized Data Output



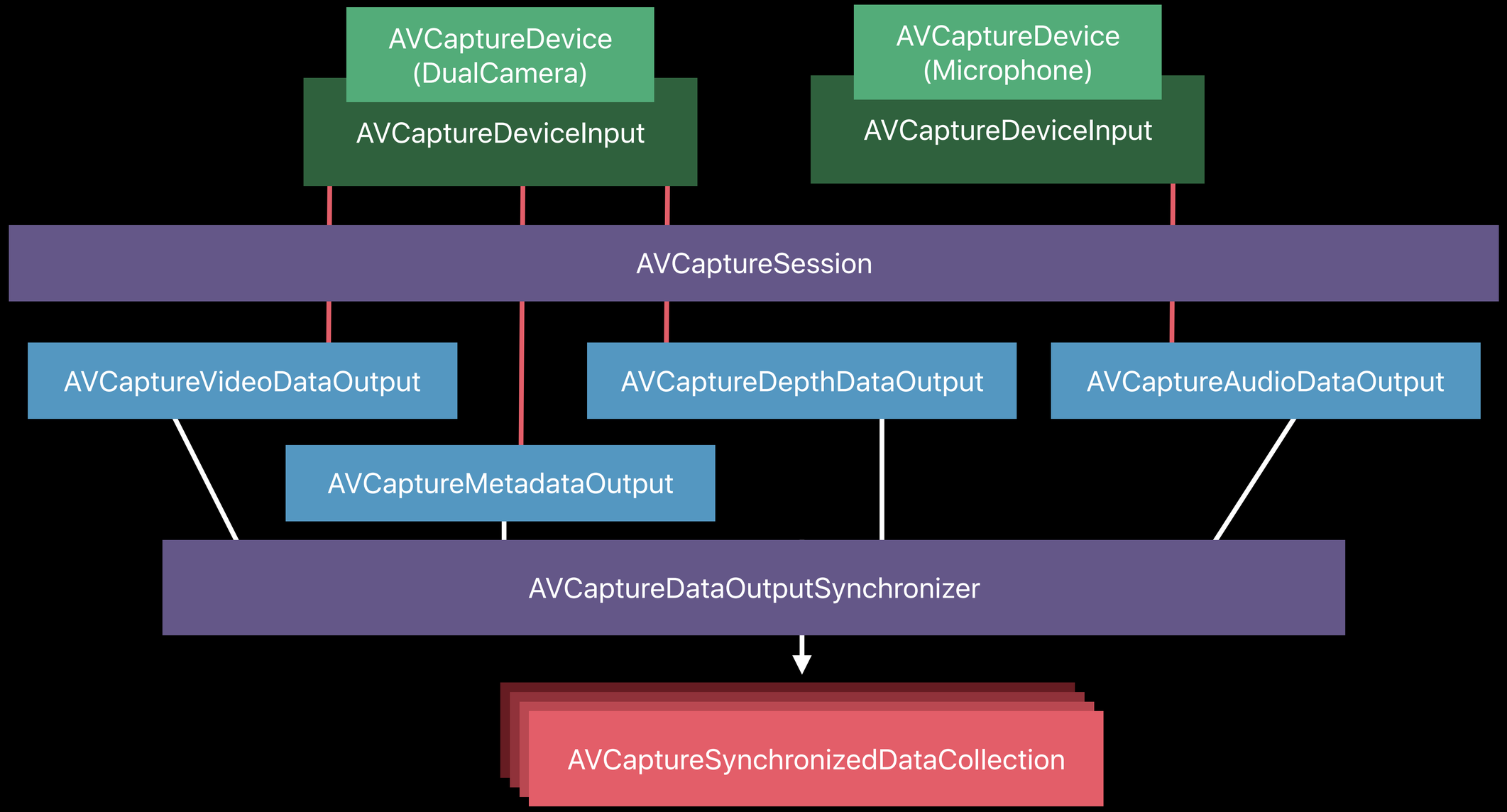
# Synchronized Data Output



# Synchronized Data Output



# Synchronized Data Output



```
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 {
        // ...
    }
}
```

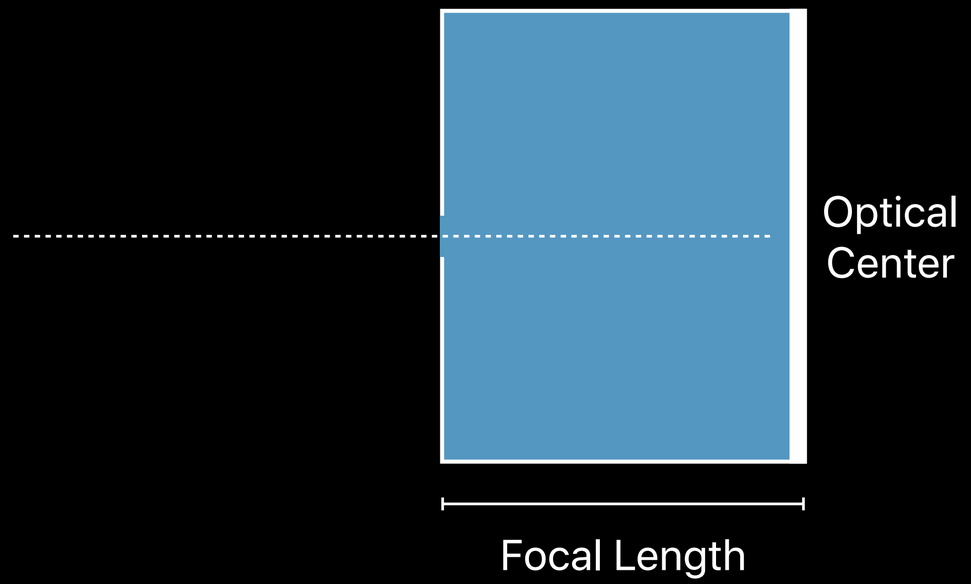
```
func dataOutputSynchronizer(_ synchronizer: AVCaptureDataOutputSynchronizer, didOutput
                            synchronizedDataCollection: AVCaptureSynchronizedDataCollection) {

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

    if let synDepth = synchronizedDataCollection[self.ddo] as? AVCaptureSynchronizedDepthData {
        // ...
    }
}
```

# AVCamPhotoFilter

Sample code

# Streaming Camera Intrinsics

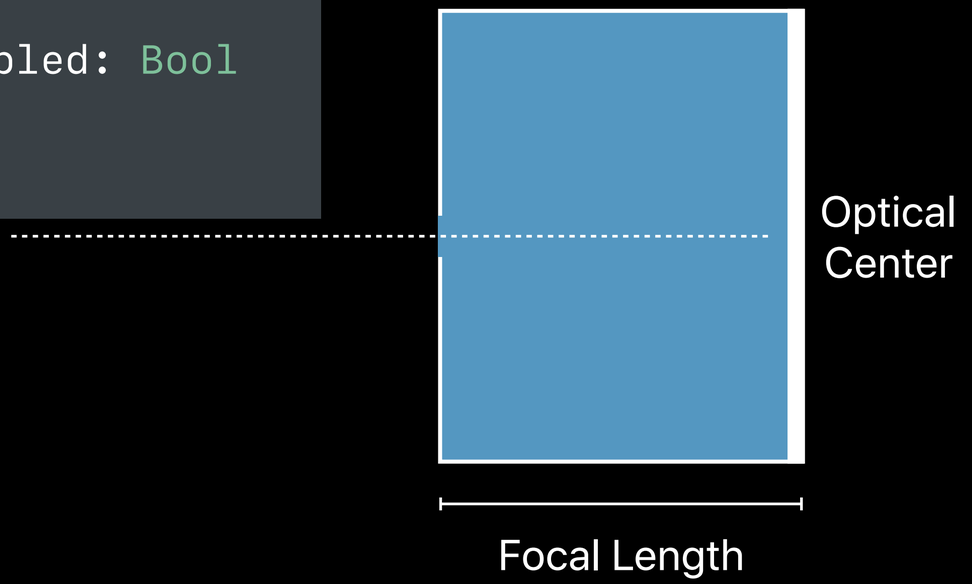
NEW



# Streaming Camera Intrinsics

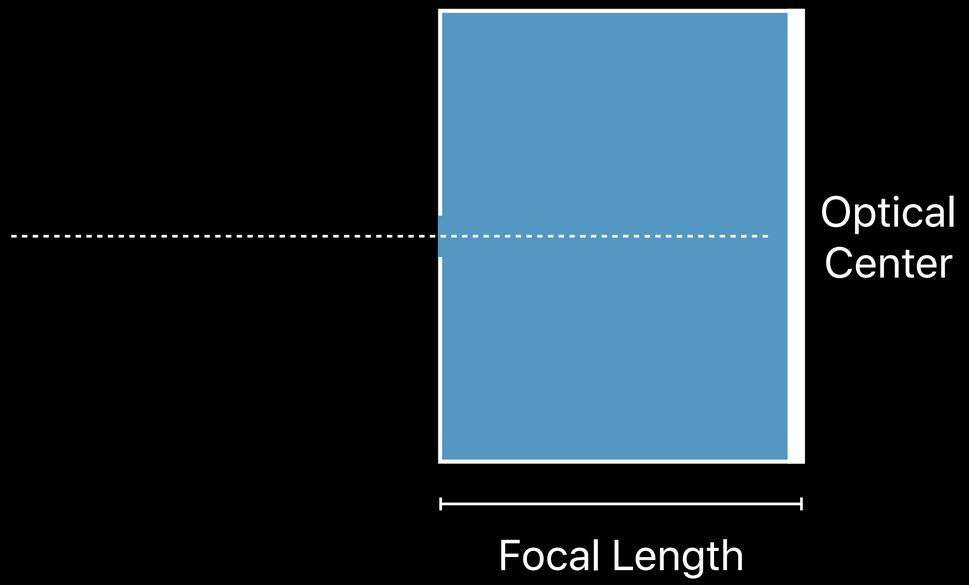
NEW

```
open class AVCaptureConnection : NSObject {

    open var isCameraIntrinsicMatrixDeliveryEnabled: Bool

}
```



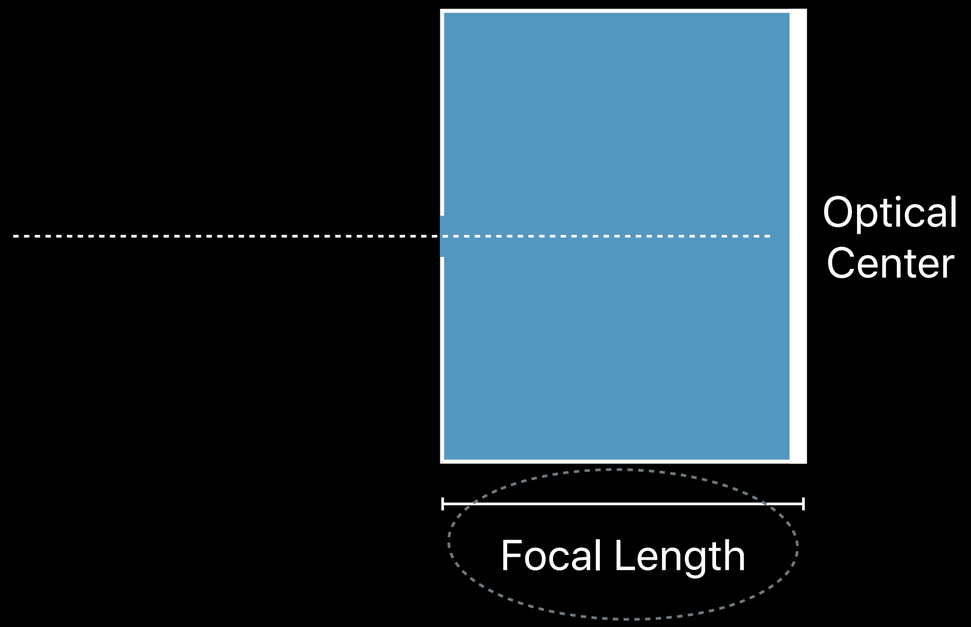
# Streaming Camera Intrinsics

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



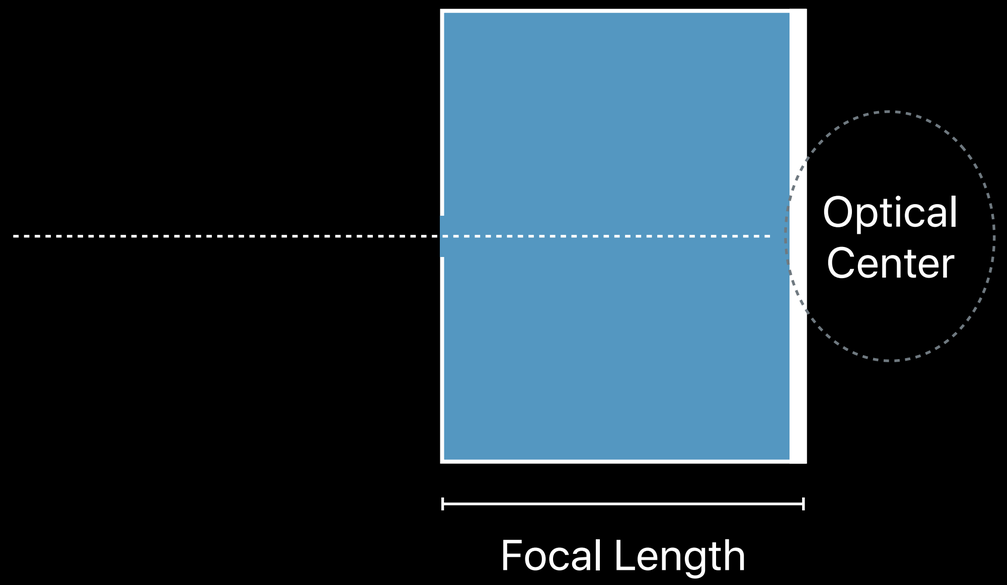
# Streaming Camera Intrinsics

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



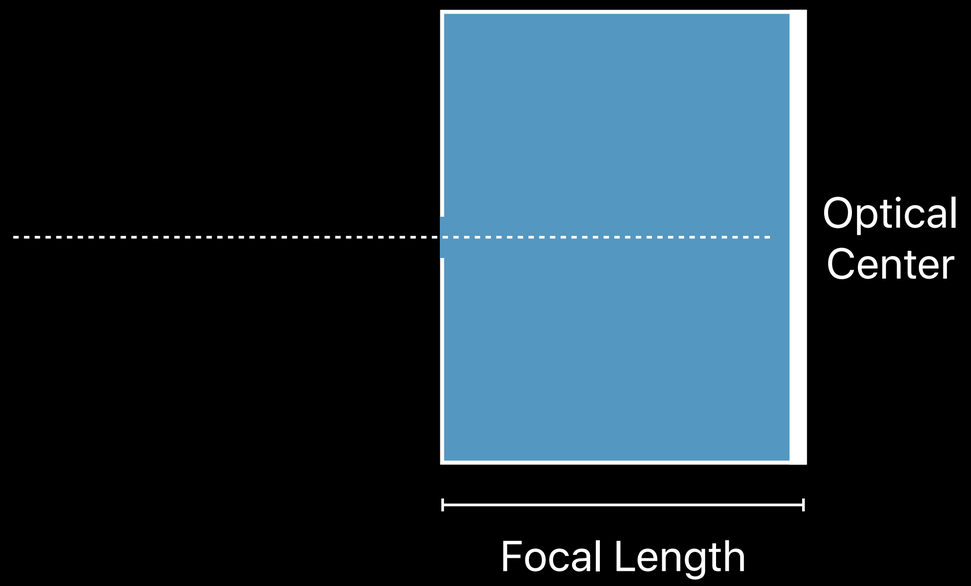
# Streaming Camera Intrinsics

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



# 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

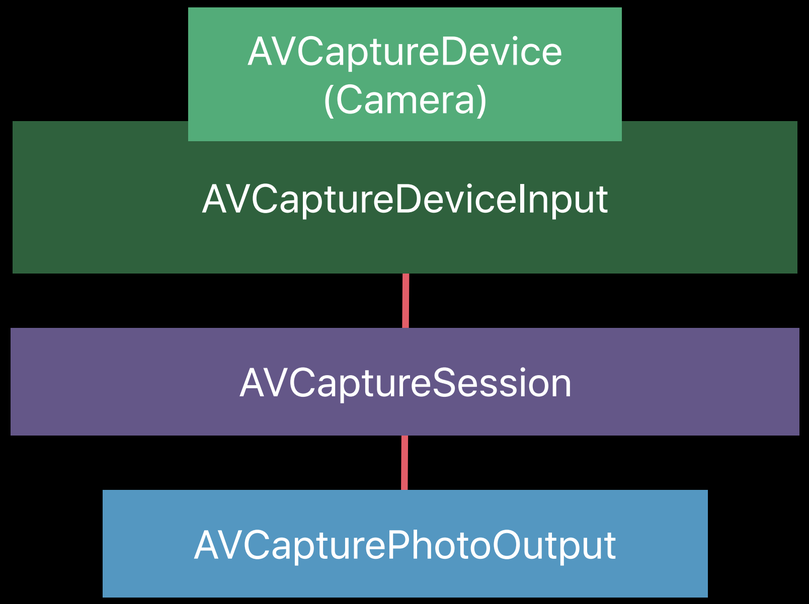
AVCam

Wiggle Me

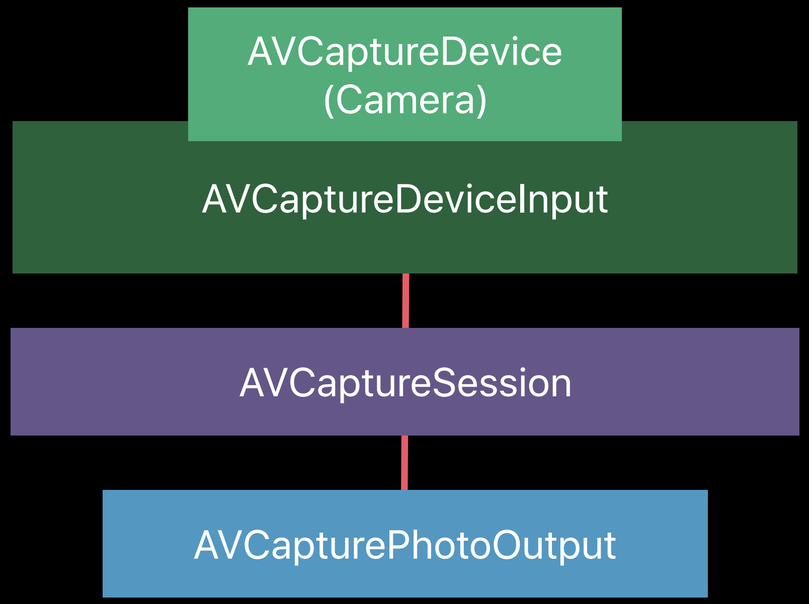
# Photos with Depth

- Flash
- Auto still image stabilization
- Auto exposure brackets
- Live photos

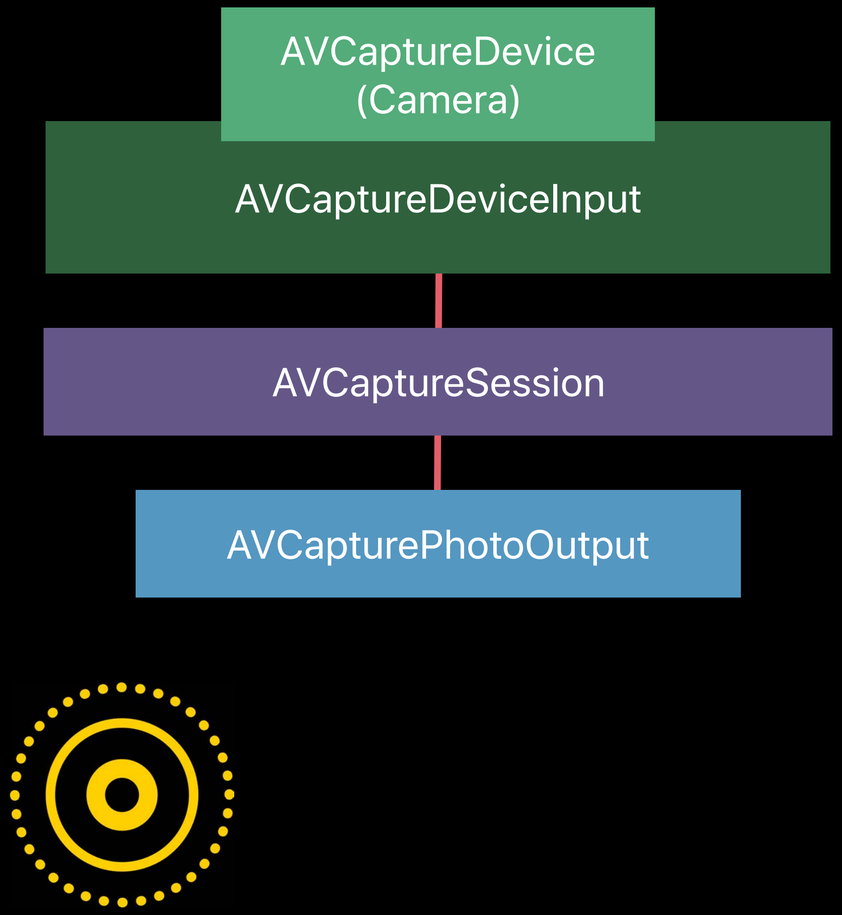
# Capturing Photos with Depth



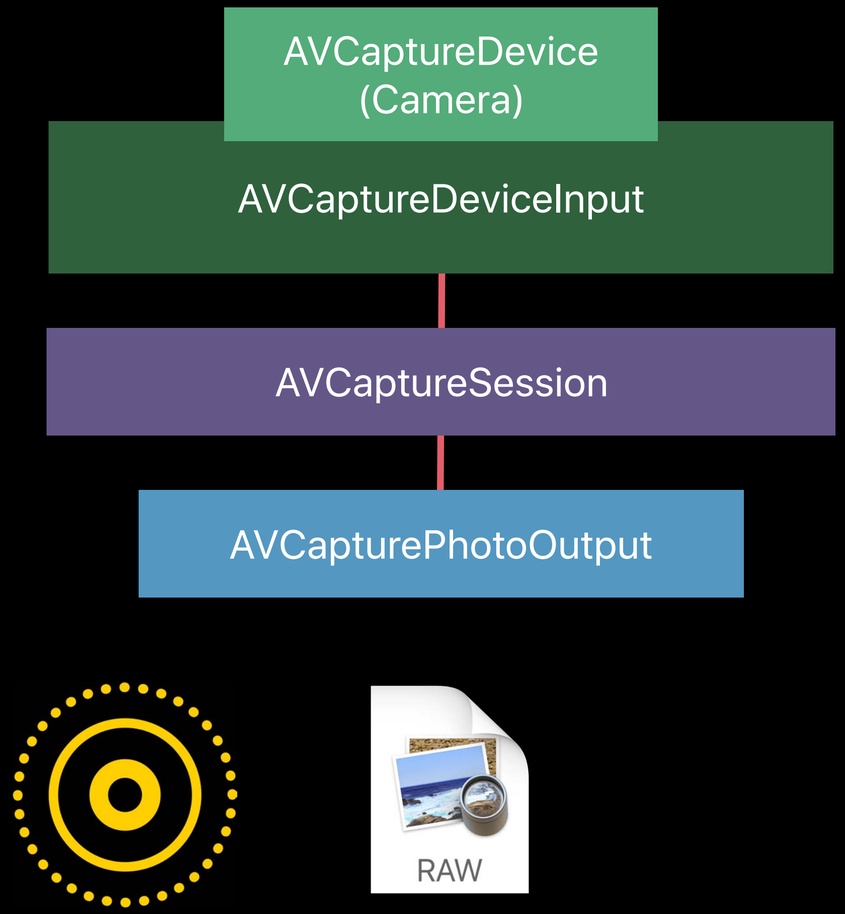
# Capturing Photos with Depth



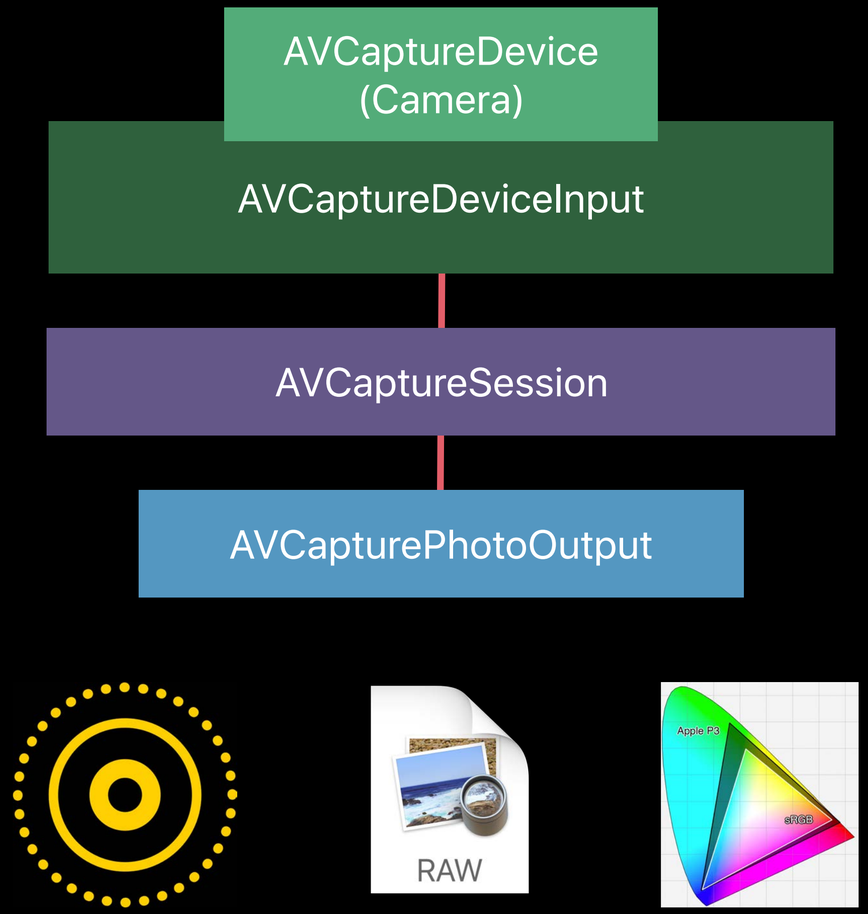
# Capturing Photos with Depth



# Capturing Photos with Depth



# Capturing Photos with Depth



```
func photoOutput(_ output: AVCapturePhotoOutput,
                 didFinishProcessingPhoto photo: AVCapturePhoto,
                 error: Error?)
```

# Requesting Depth with Photos

# Requesting Depth with Photos

Before starting the capture session, opt in!

```
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!

```
photoOutput.isDepthDataDeliveryEnabled = true
```

Request depth on a per-photo basis

```
photoSettings.isDepthDataDeliveryEnabled = true
```

Work with the resulting AVDepthData in your AVCapturePhoto

```
open class AVCapturePhoto: NSObject {
    open var depthData: AVDepthData? { get }
}
```

# High Res Photo Depth Maps

| Preset or Active Format | Streaming Video Resolution | High Res Photo Resolution | Streaming Depth Resolution | Photo Depth Map Resolution |
|---|---|---|---|---|
| .photo | 1440x1080 | 4032x3024 | 320x240<br>160x120 | 768x576 |
| 1280x720@3-30 FPS | 1280x720 | 4096x2304 | 320x180<br>160x90 | 768x432 |
| .vga640x480 | 640x480 | 4032x3024 | 320x240<br>160x120 | 768x576 |

# High Res Photo Depth Maps

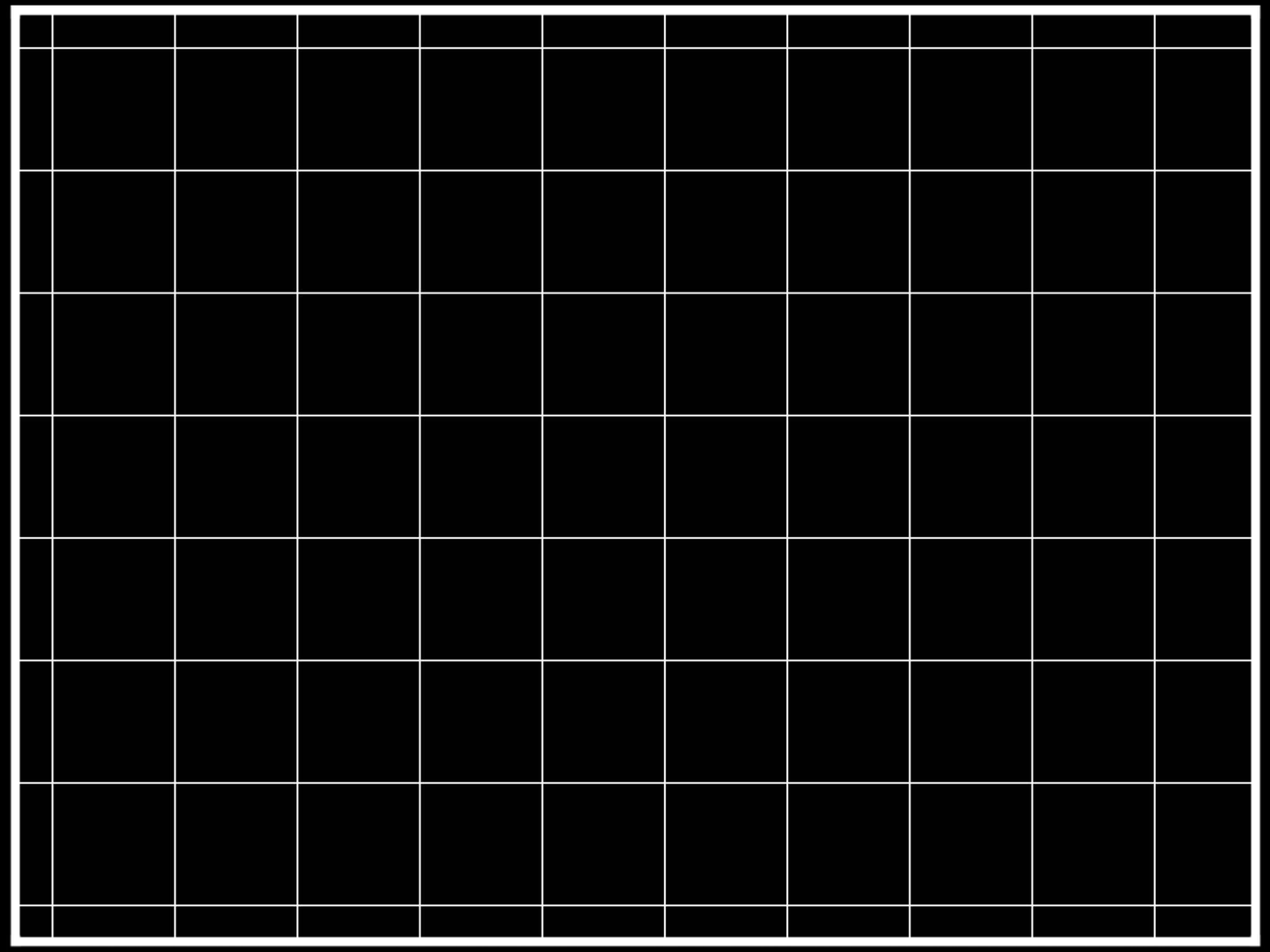
| Preset or Active Format | Streaming Video Resolution | High Res Photo Resolution | Streaming Depth Resolution | Photo Depth Map Resolution |
|---|---|---|---|---|
| .photo | 1440x1080 | 4032x3024 | 320x240<br>160x120 | 768x576 |
| 1280x720@3-30 FPS | 1280x720 | 4096x2304 | 320x180<br>160x90 | 768x432 |
| .vga640x480 | 640x480 | 4032x3024 | 320x240<br>160x120 | 768x576 |

# High Res Photo Depth Maps

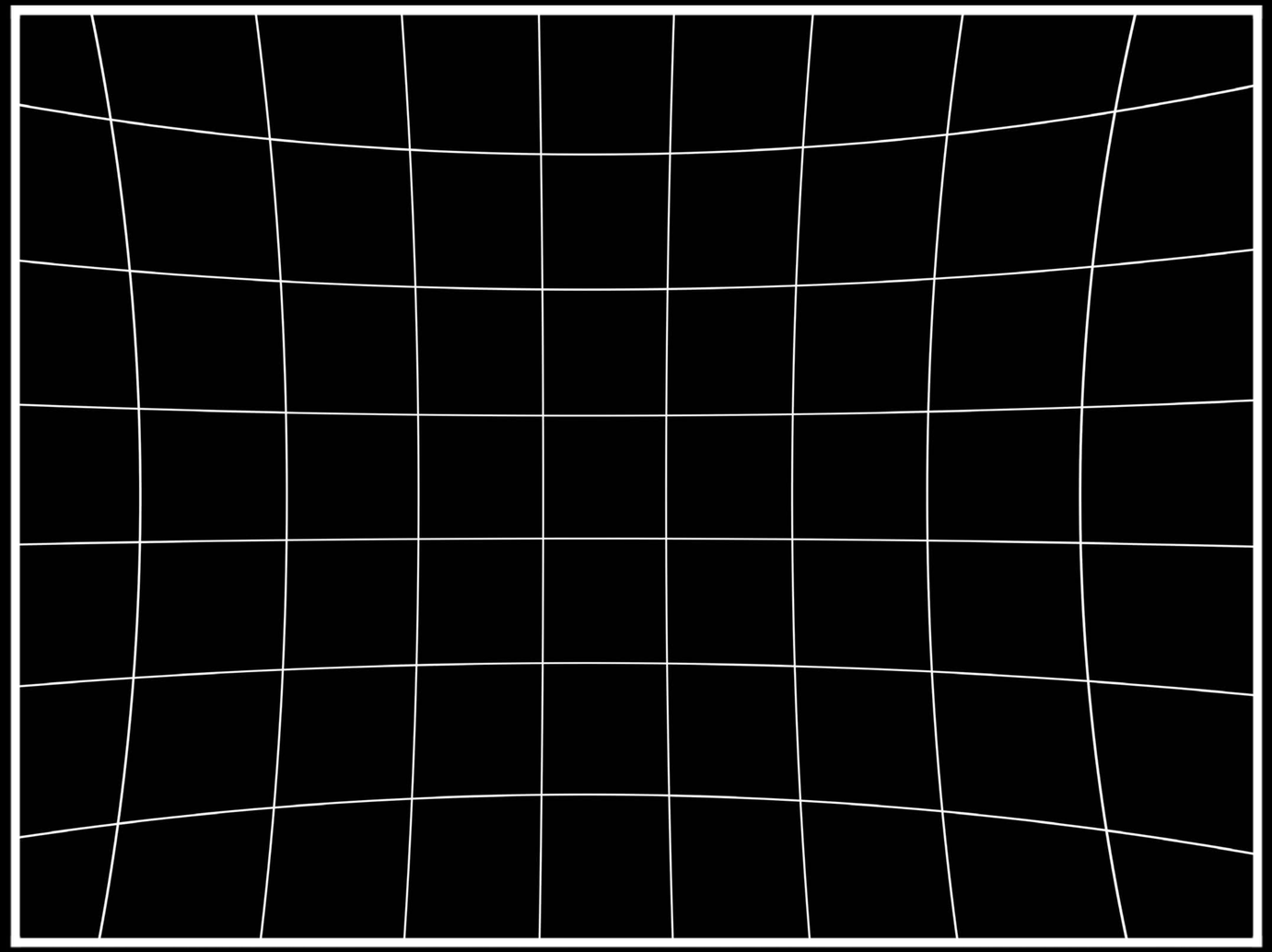
| Preset or Active Format | Streaming Video Resolution | High Res Photo Resolution | Streaming Depth Resolution | Photo Depth Map Resolution |
|---|---|---|---|---|
| .photo | 1440x1080 | 4032x3024 | 320x240<br>160x120 | 768x576 |
| 1280x720@3-30 FPS | 1280x720 | 4096x2304 | 320x180<br>160x90 | 768x432 |
| .vga640x480 | 640x480 | 4032x3024 | 320x240<br>160x120 | 768x576 |

# Rectilinear vs. Lens Distorted Images

# Rectilinear vs. Lens Distorted Images

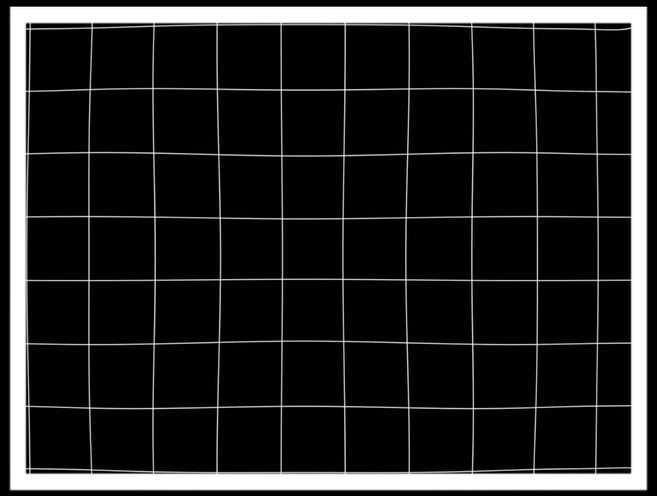


# Rectilinear vs. Lens Distorted Images

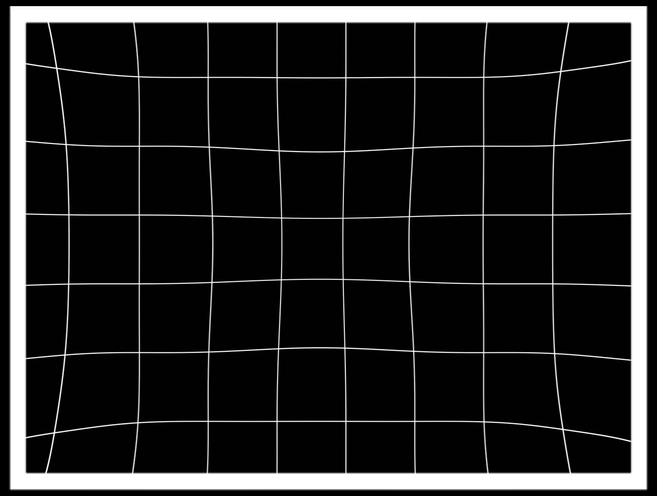


# Depth Map Distortions

Tele

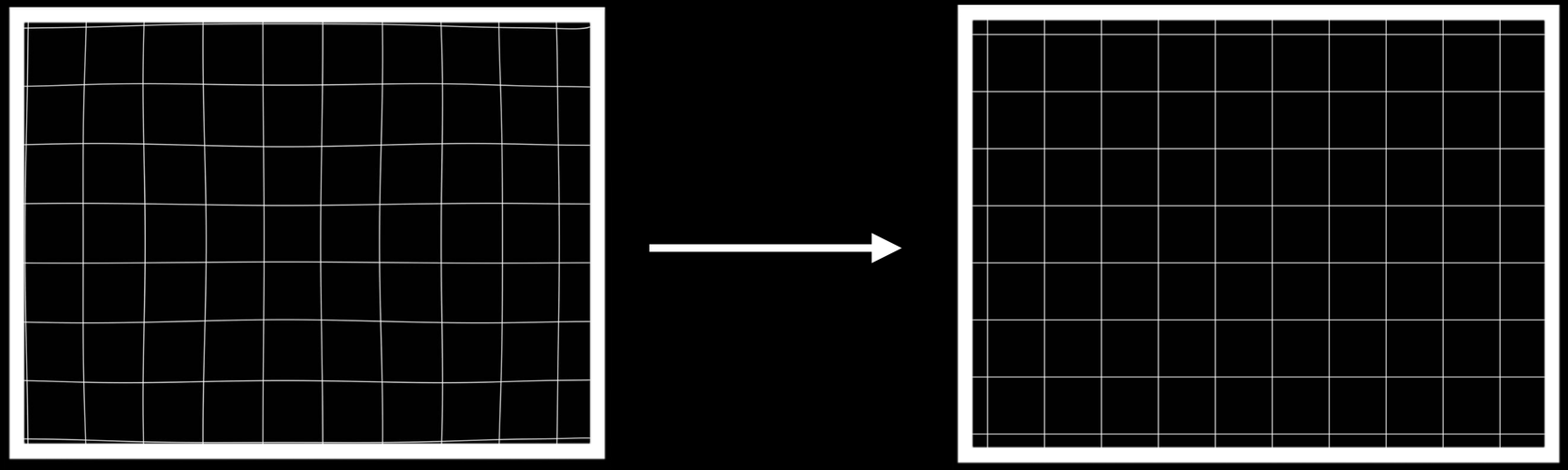


Wide

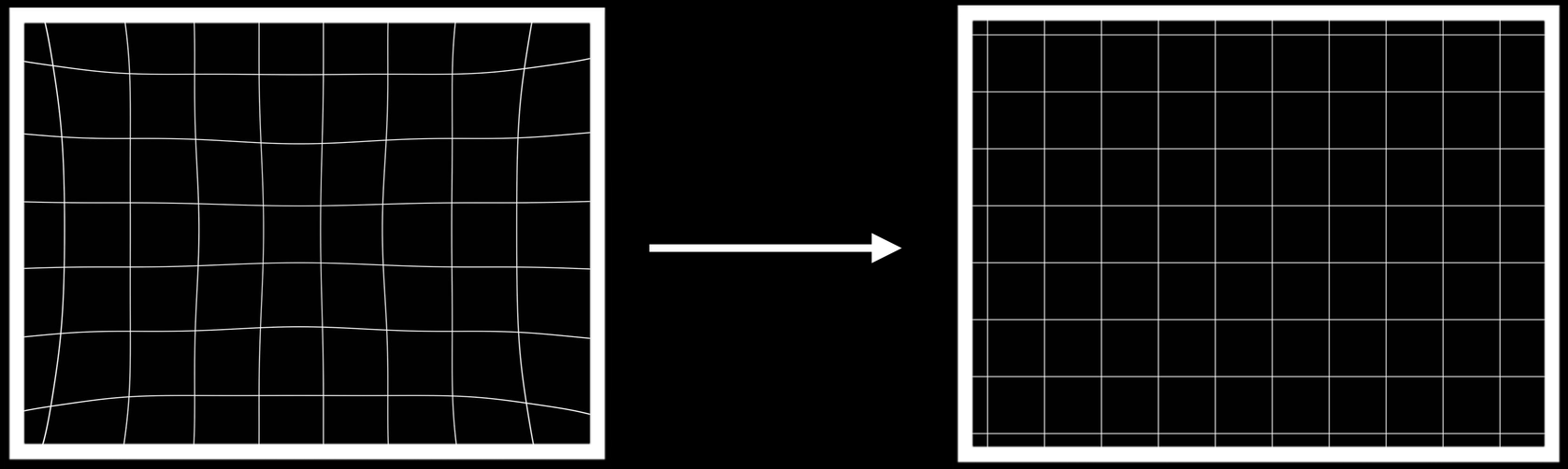


# Depth Map Distortions

Tele



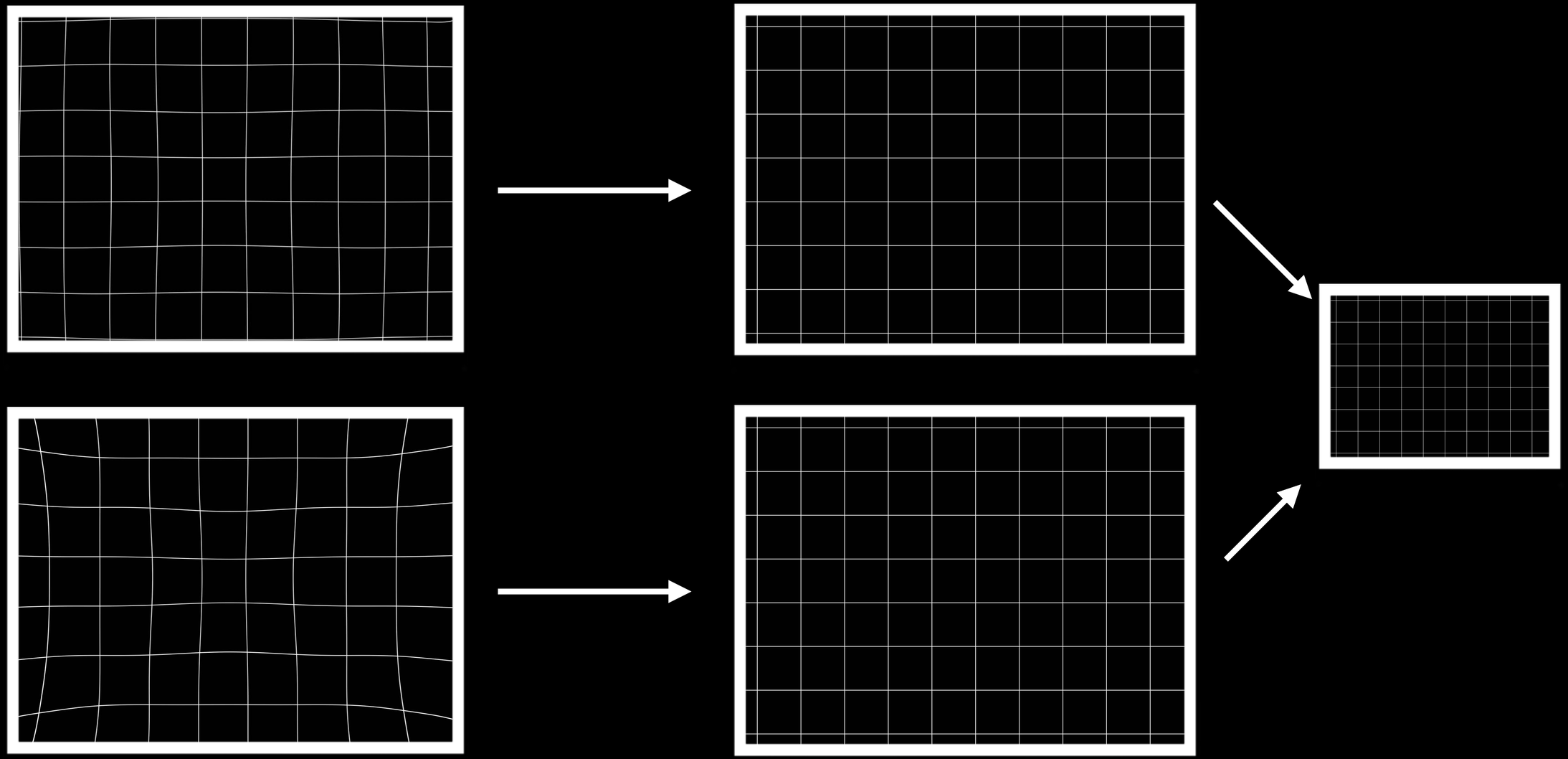
Wide



# Depth Map Distortions

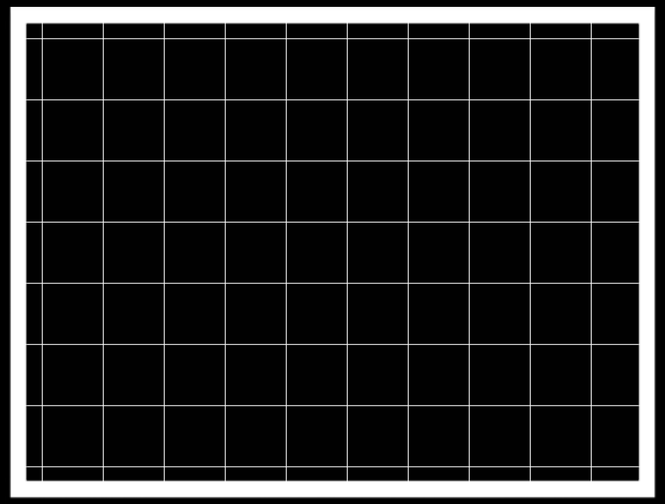
Tele

Wide



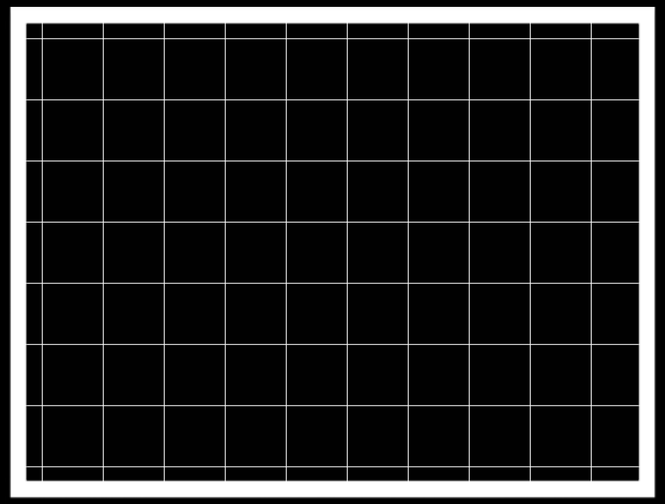
# Depth Map Distortions

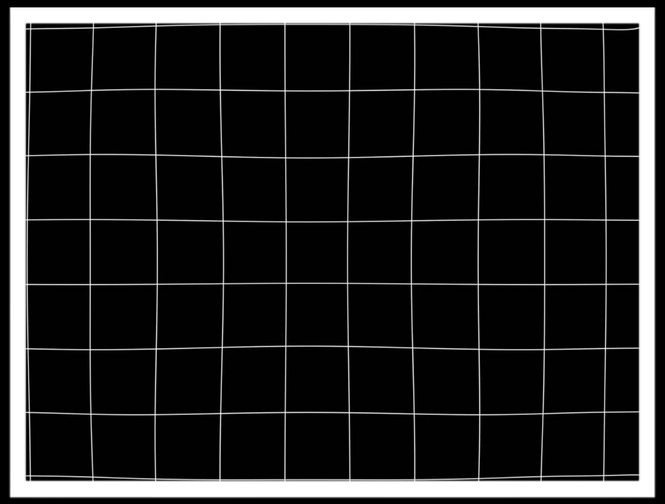
Rectilinear
Disparity Map



# 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

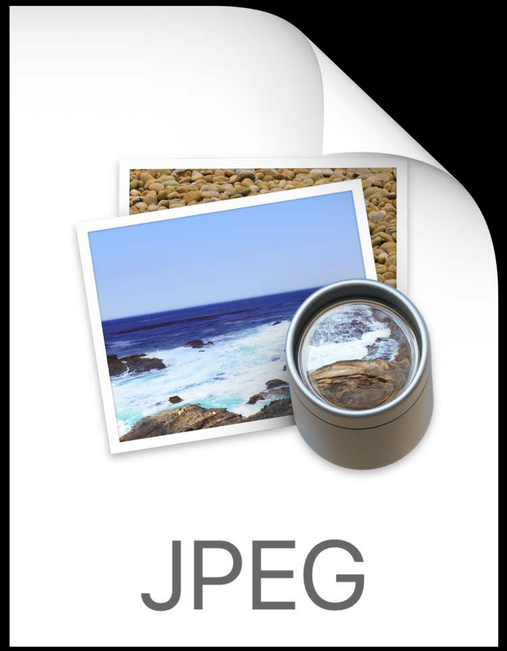
HEIF HEVC (HEIC)

- Map is an auxiliary image
- Encoded as monochrome HEVC
- Metadata is XMP in the auxiliary image

# Depth in Image Files

JPEG

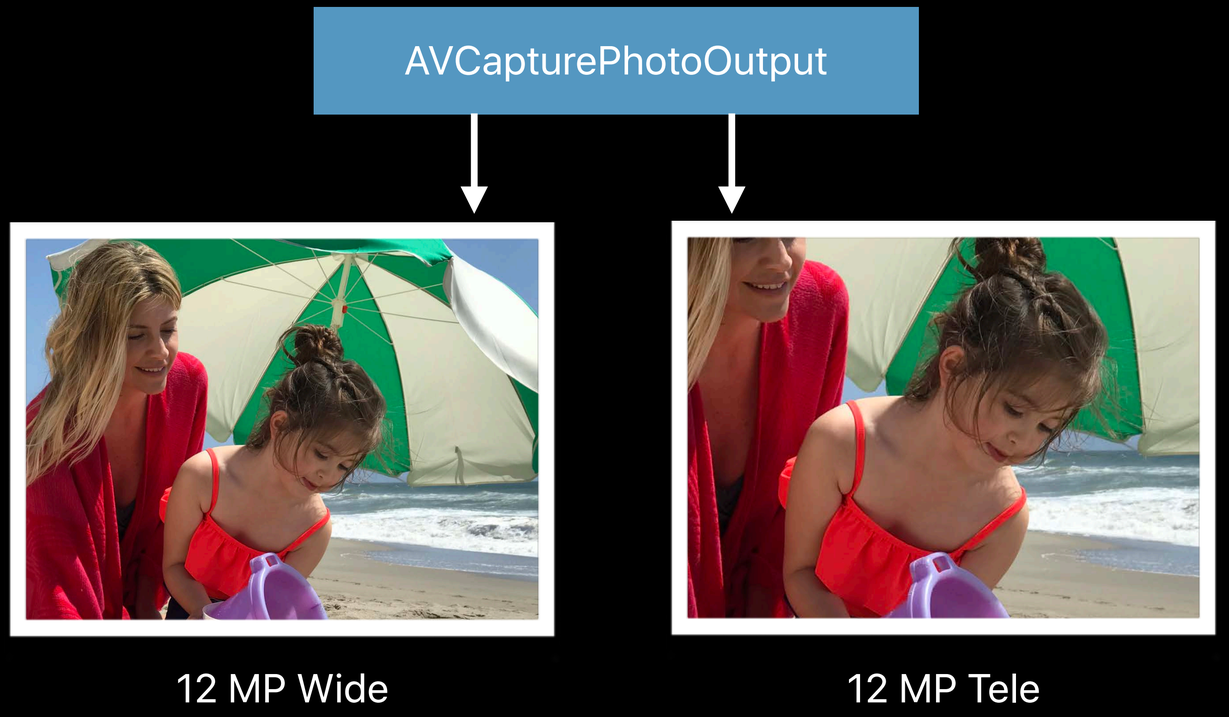
- Map is 8-bit lossy JPEG (filtered) or 16-bit lossless (unfiltered) MPO
- Metadata is XMP in the second image



# Dual Photo Capture

# Dual Photo Capture

NEW



# Requesting Dual Photo Delivery

# Requesting Dual Photo Delivery

Before starting the capture session, opt in!

```
photoOutput.isDualCameraDualPhotoDeliveryEnabled = true
```

# Requesting Dual Photo Delivery

Before starting the capture session, opt in!

```
photoOutput.isDualCameraDualPhotoDeliveryEnabled = true
```

Request Dual Photo capture on a per-request basis

```
photoSettings.isDualCameraDualPhotoDeliveryEnabled = true
```

# Requesting Dual Photo Delivery

Before starting the capture session, opt in!

```
photoOutput.isDualCameraDualPhotoDeliveryEnabled = true
```

Request Dual Photo capture on a per-request basis

```
photoSettings.isDualCameraDualPhotoDeliveryEnabled = true
```

Number of AVCapturePhoto callbacks doubles

```
let sourceDeviceType = photo.sourceDeviceType
```

# Dual Photo Capture

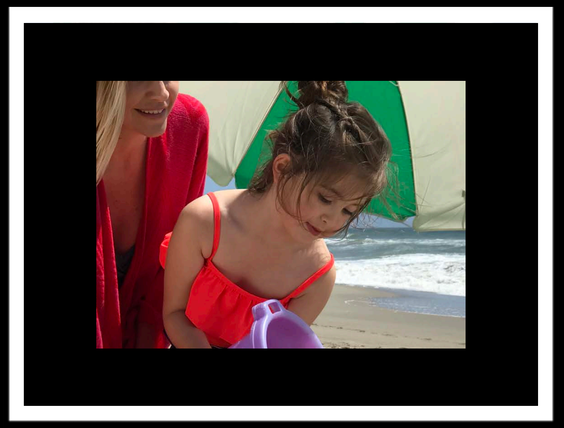
Supports

- Flash captures
- Auto still image stabilization
- Auto exposure brackets
- Depth delivery

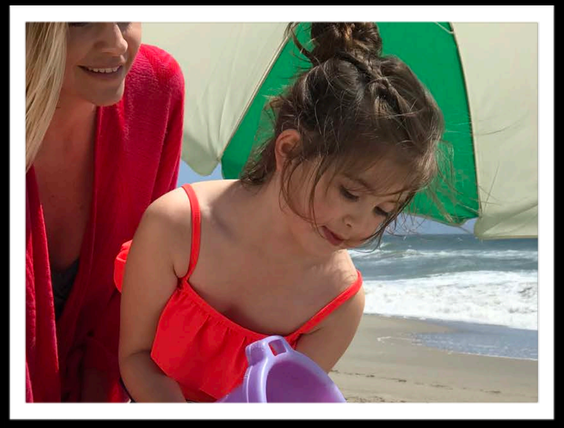
# Dual Photo Capture + Zoom

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

NEW

Property of AVDepthData

# Introducing AVCameraCalibrationData

NEW

Property of AVDepthData

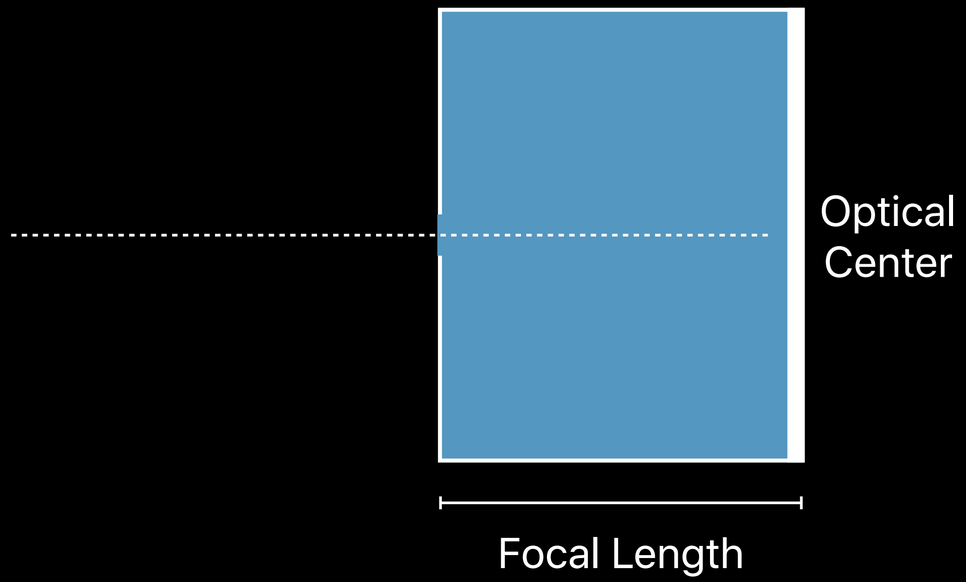
Property of AVCapturePhoto (if you opt in):

```
photoSettings.isCameraCalibrationDeliveryEnabled = true
```

# AVCameraCalibrationData

```
open var intrinsicMatrix: matrix_float3x3 { get }
```

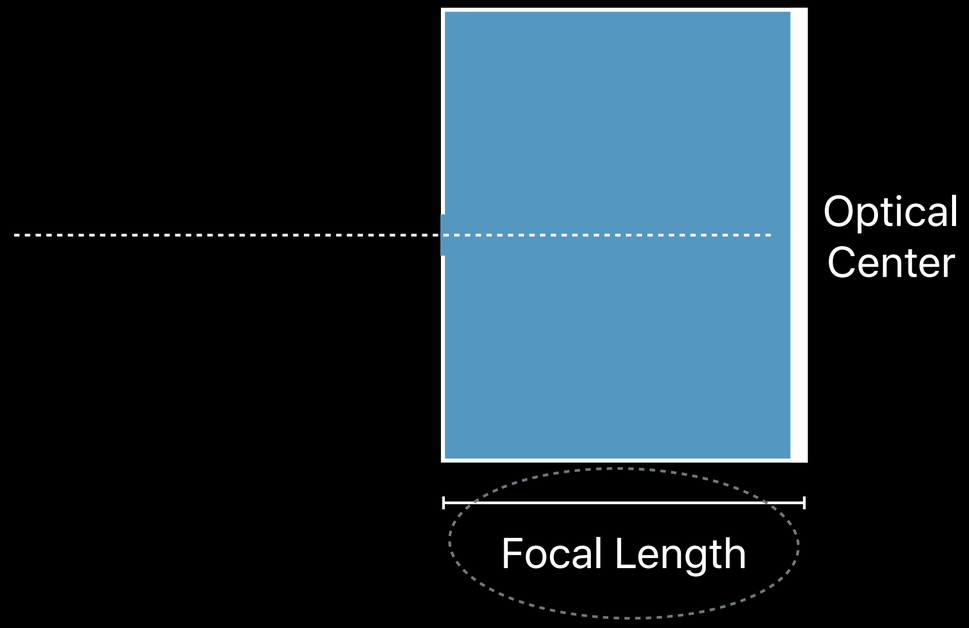
$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$



# AVCameraCalibrationData

```
open var intrinsicMatrix: matrix_float3x3 { get }
```

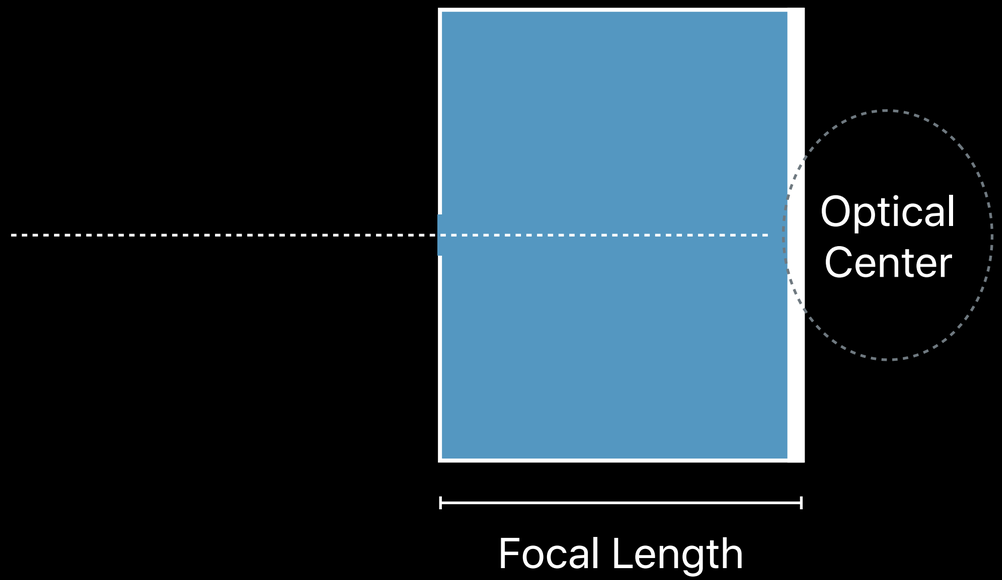
$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$



# AVCameraCalibrationData

```
open var intrinsicMatrix: matrix_float3x3 { get }
```

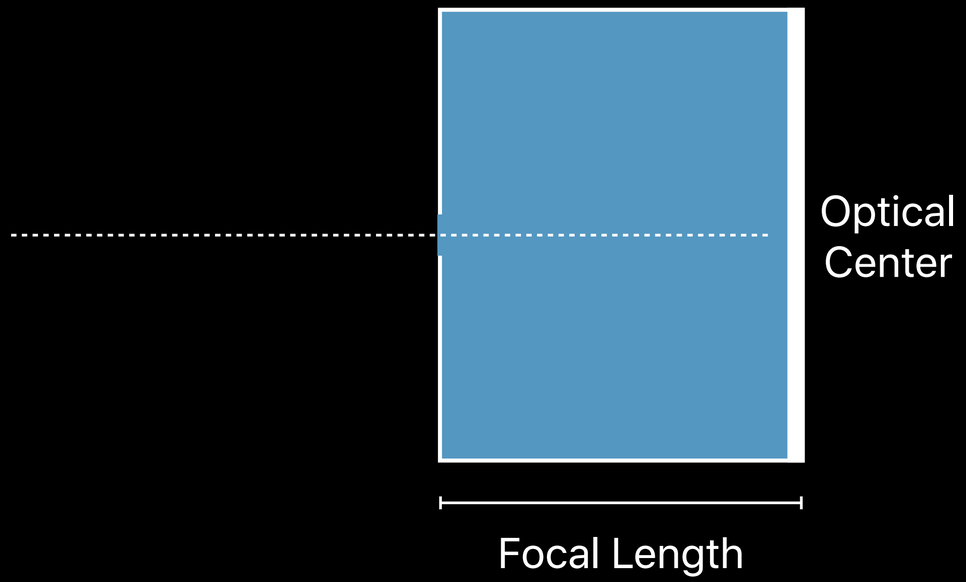
$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$



# 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 }
```

# AVCameraCalibrationData

```
open var extrinsicMatrix: matrix_float4x3 { get }
```

# AVCameraCalibrationData

```
open var extrinsicMatrix: matrix_float4x3 { get }
```

$$[R \mid t] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right]$$

# AVCameraCalibrationData

```
open var extrinsicMatrix: matrix_float4x3 { get }
```

$$[R \mid t] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right]$$

Rotation

# AVCameraCalibrationData

```
open var extrinsicMatrix: matrix_float4x3 { get }
```

$$[R \mid t] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right]$$

Translation

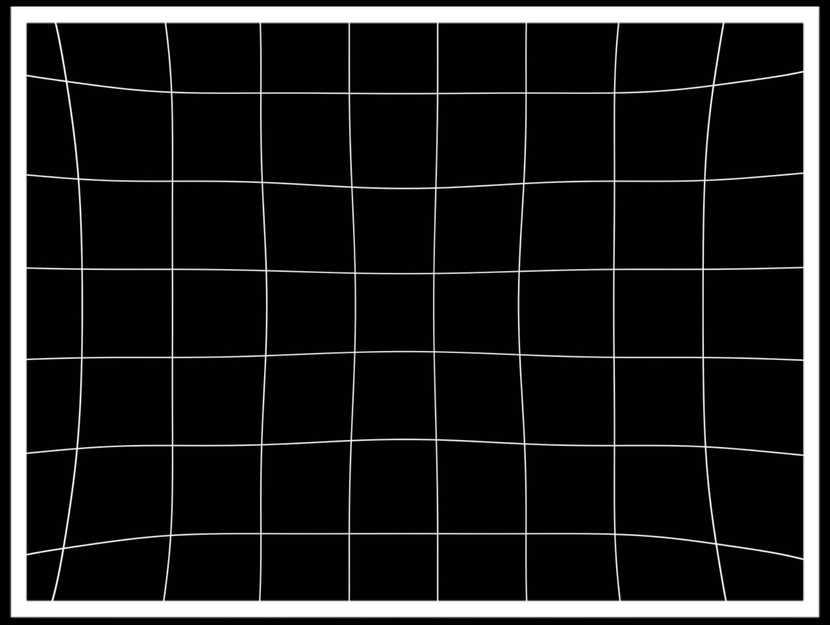
# AVCameraCalibrationData

```
open var extrinsicMatrix: matrix_float4x3 { get }
```

$$[R \mid t] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right]$$

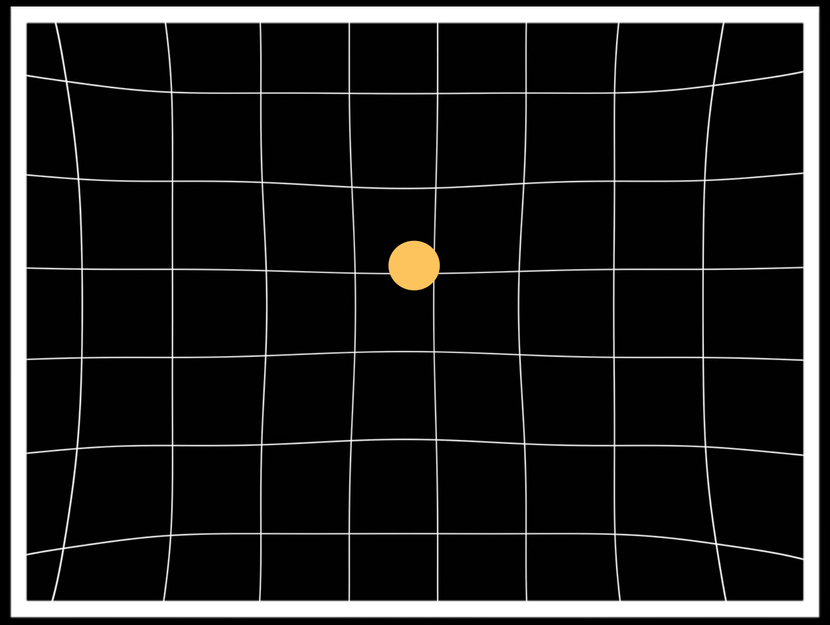
Tele camera is the world origin

# AVCameraCalibrationData



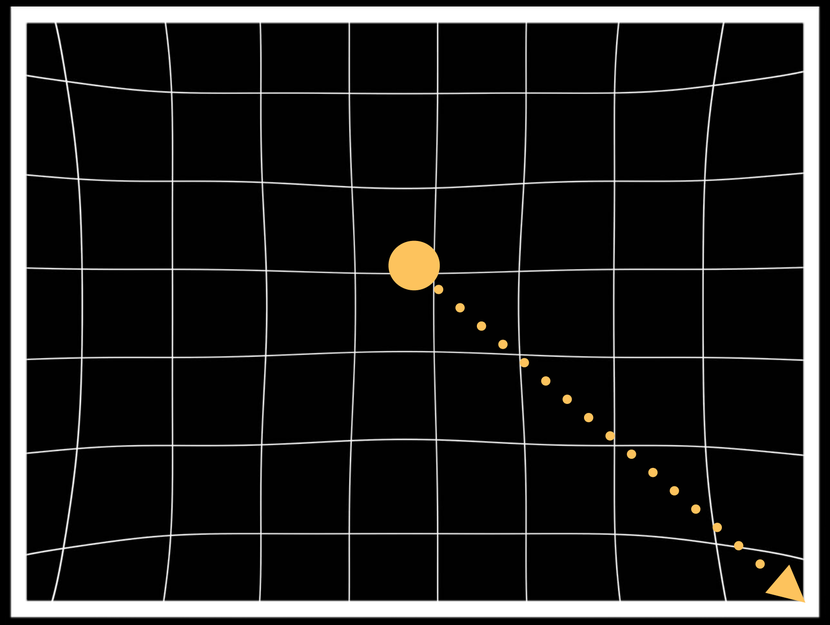
# AVCameraCalibrationData

```
open var lensDistortionCenter: CGPoint { get }
```



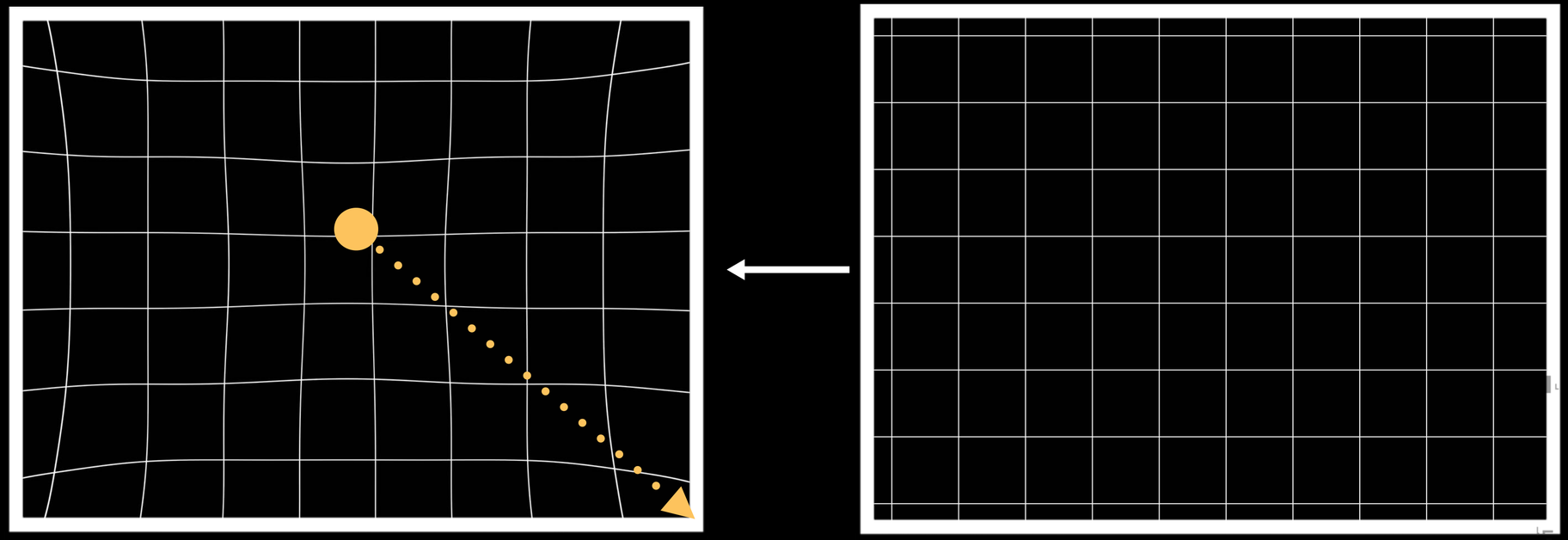
# AVCameraCalibrationData

```
open var lensDistortionCenter: CGPoint { get }

open var lensDistortionLookupTable: Data { get }
```



# AVCameraCalibrationData

```
open var lensDistortionCenter: CGPoint { get }

open var lensDistortionLookupTable: Data { get }
```

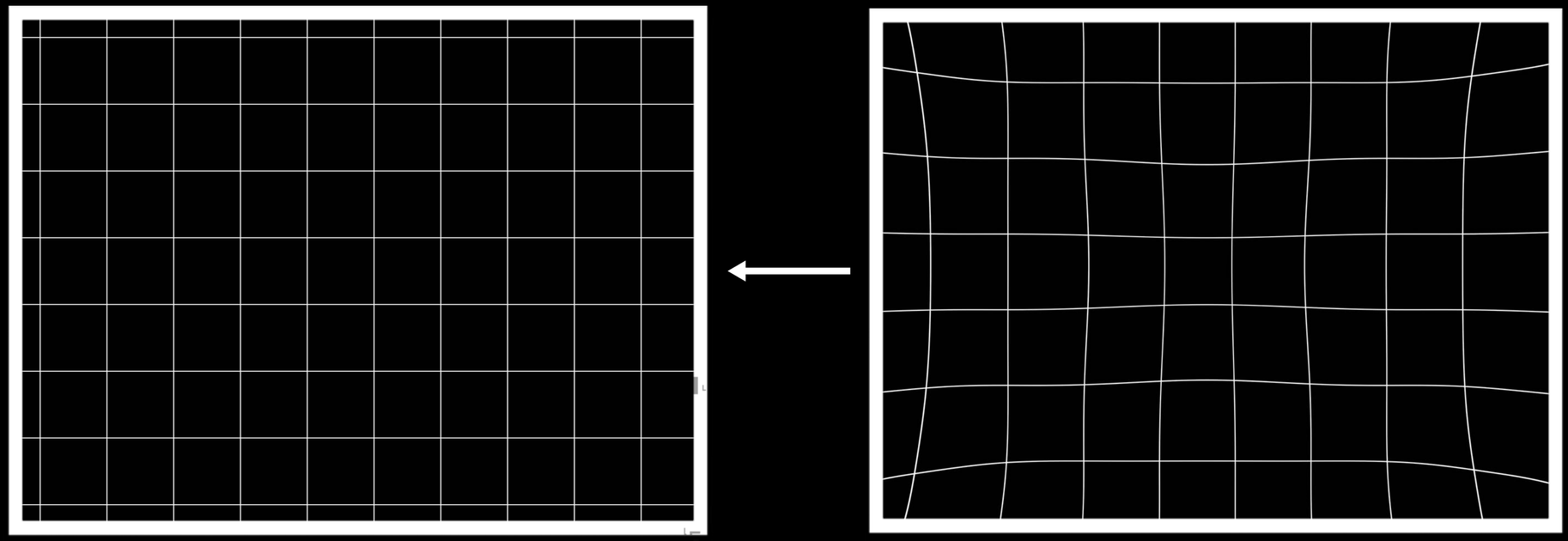


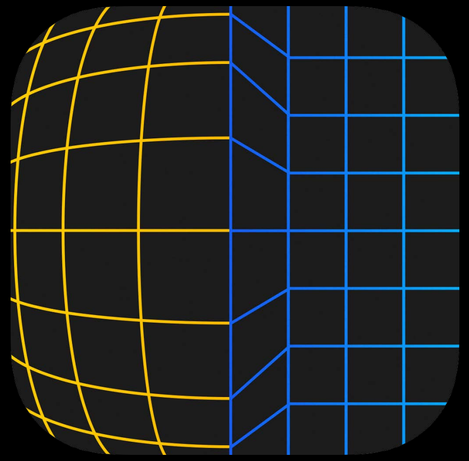
# AVCameraCalibrationData

```
open var inverseLensDistortionLookupTable: Data { get }
```

# AVCameraCalibrationData

```
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

https://developer.apple.com/wwdc17/507

# Related Sessions

| | | |
|---|---|---|
| Photography Get Together | Technology Lab J | Wednesday 6:30PM |
| Image Editing with Depth | Hall 2 | Thursday 11:00AM |
| Working with HEIF and HEVC | Hall 2 | Friday 11:00AM |

# Labs

| | | |
|---|---|---|
| Photo Extensions & PhotoKit Lab | Technology Lab A | Thur 9:00AM–12:00PM |
| AVFoundation Lab | Technology Lab F | Thur 12:00PM–3:00PM |
| Photos Editing & Core Image Lab | Technology Lab F | Thur 3:10PM–6:00PM |
| Photos Depth & Capture Lab | Technology Lab A | Thur 3:10PM–6:00PM |
| HEIF and HEVC Lab | Technology Lab F | Fri 12:00PM–1:50PM |
| Vision Lab | Technology Lab A | Fri 1:50PM–4:00PM |
| Photos Depth & Capture Lab | Technology Lab F | Fri 1:50PM–4:00PM |

# Labs

| | | |
|---|---|---|
| Photo Extensions & PhotoKit Lab | Technology Lab A | Thur 9:00AM–12:00PM |
| AVFoundation Lab | Technology Lab F | Thur 12:00PM–3:00PM |
| Photos Editing & Core Image Lab | Technology Lab F | Thur 3:10PM–6:00PM |
| Photos Depth & Capture Lab | Technology Lab A | Thur 3:10PM–6:00PM |
| HEIF and HEVC Lab | Technology Lab F | Fri 12:00PM–1:50PM |
| Vision Lab | Technology Lab A | Fri 1:50PM–4:00PM |
| Photos Depth & Capture Lab | Technology Lab F | Fri 1:50PM–4:00PM |


WWDC17