#WWDC18

Creating Photo and Video Effects Using Depth

Session 503

Emmanuel Piuze-Phaneuf, Core Image
Ron Sokolovsky, Video Engineering

Portrait

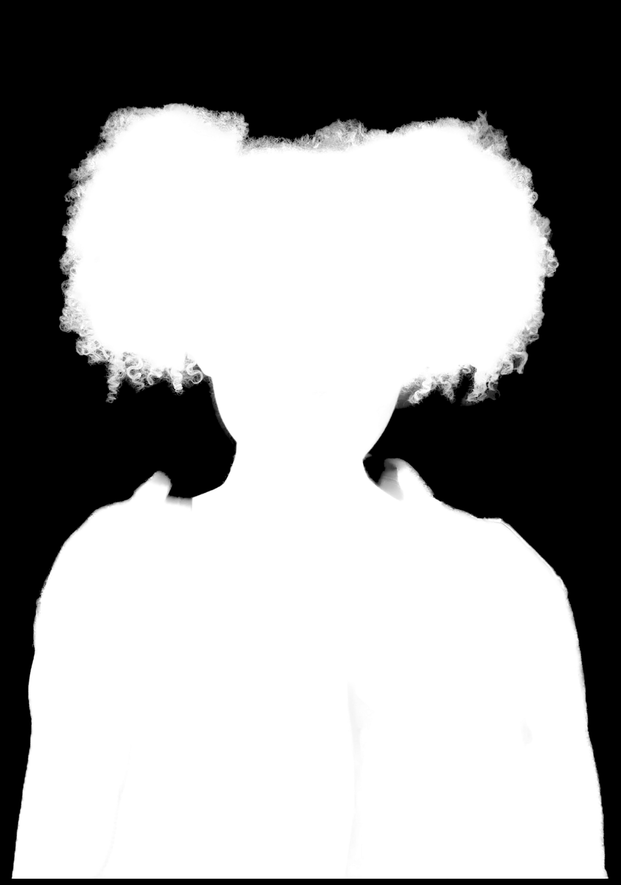
Portrait

Depth

NEW

Portrait

Depth



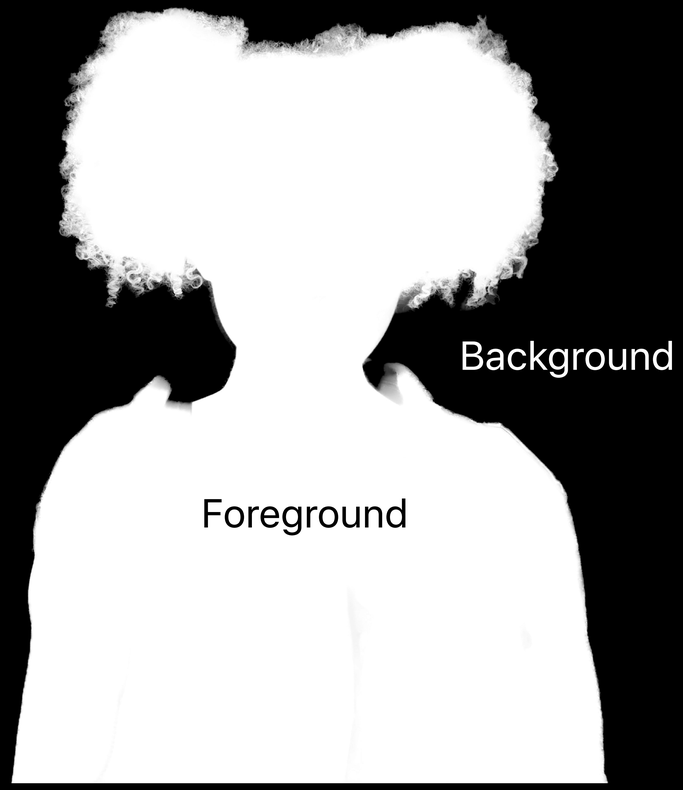
Matte

Portrait Segmentation API

Portrait Matte

Portrait Matte

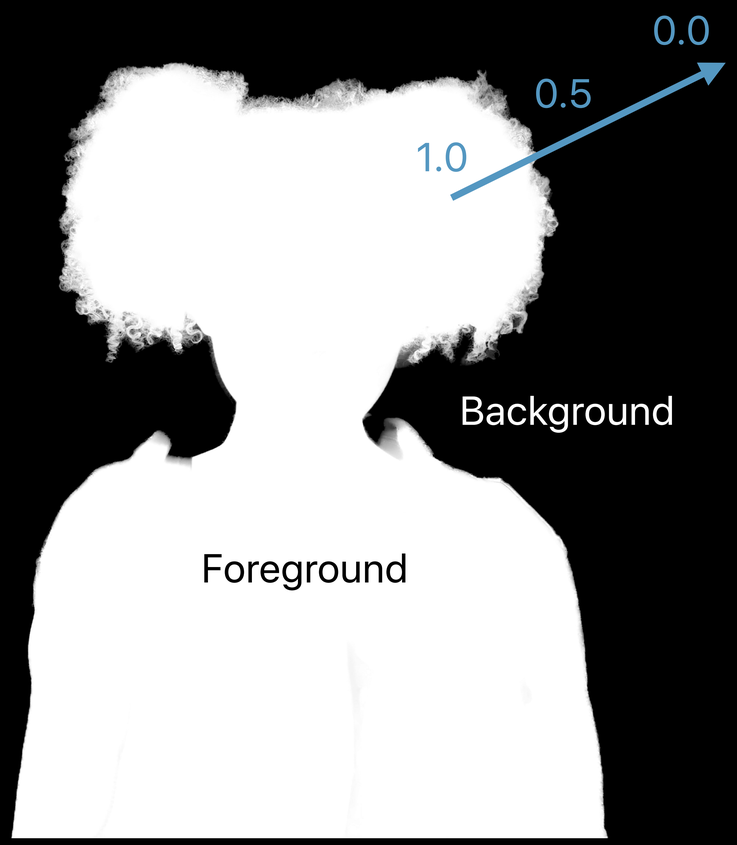
Segmentation



Portrait Matte

Segmentation

Fuzzy mask

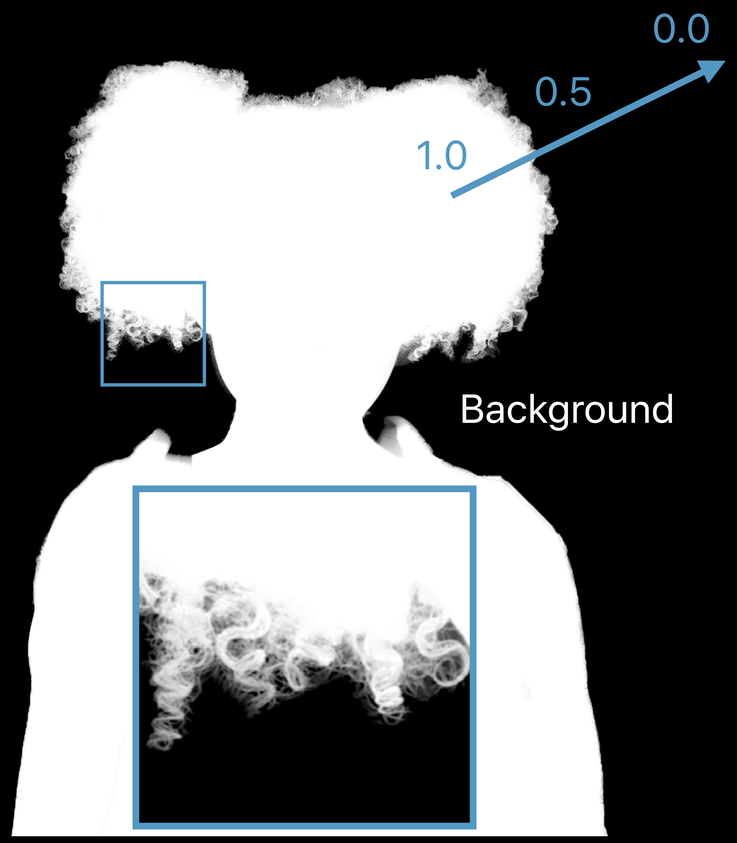


Portrait Matte

Segmentation

Fuzzy mask

Fine details



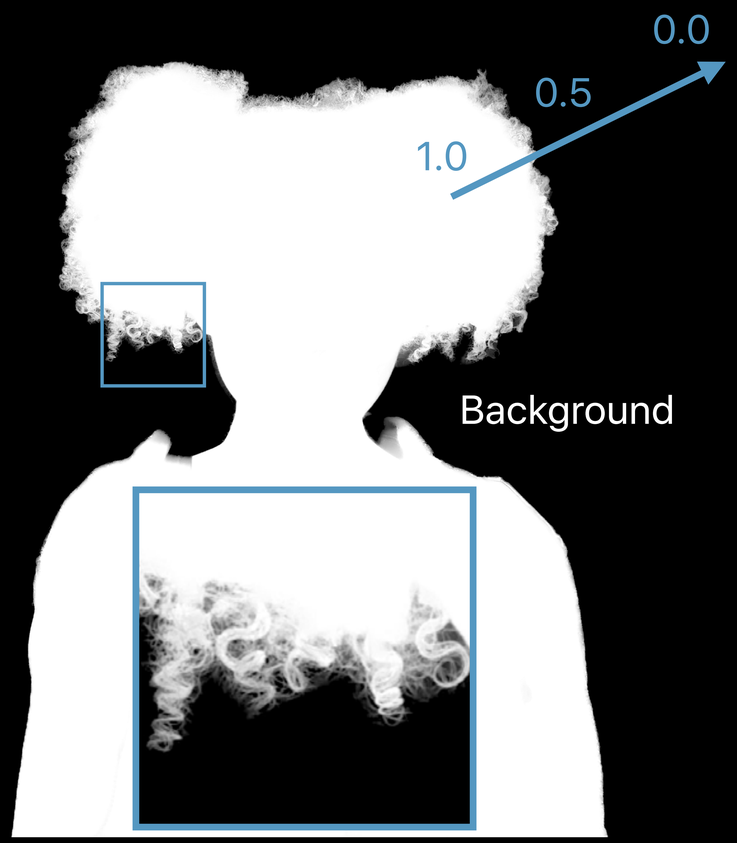
Portrait Matte

Segmentation

Fuzzy mask

Fine details

Many applications



iOS 12

iOS 12

Front and rear

iOS 12

Front and rear

Portrait

iOS 12

Front and rear

Portrait

People

iOS 12

Front and rear

Portrait

People

Linear

iOS 12

Front and rear

Portrait

People

Linear

No guarantee

Loading

ImageIO

```
let auxiliaryData = CGImageSourceCopyAuxiliaryDataInfoAtIndex(
                imageSource, // CGImageSource
                0, // Int
                kCGImageAuxiliaryDataTypePortraitEffectsMatte) // New ImageIO flag
```

Loading

ImageIO

```
let auxiliaryData = CGImageSourceCopyAuxiliaryDataInfoAtIndex(
                imageSource, // CGImageSource
                0, // Int
                kCGImageAuxiliaryDataTypePortraitEffectsMatte) // New ImageIO flag
```

Loading

ImageIO

```
let auxiliaryData = CGImageSourceCopyAuxiliaryDataInfoAtIndex(
                imageSource, // CGImageSource
                0, // Int
                kCGImageAuxiliaryDataTypePortraitEffectsMatte) // New ImageIO flag
```

```
{
  kCGImageAuxiliaryDataInfoData: CFDataRef, // Image data
  kCGImageAuxiliaryDataInfoDataDescription: CFDictionary, // Pixel buffer attributes
  kCGImageAuxiliaryDataInfoMetadata: CGImageMetadataRef // Image metadata
}
```

Loading

AVPortraitEffectsMatte

```
let auxiliaryData = CGImageSourceCopyAuxiliaryDataInfoAtIndex(...)
let matteData = AVPortraitEffectsMatte(fromDictionaryRepresentation: auxiliaryData)
```

Loading

AVPortraitEffectsMatte

```
let auxiliaryData = CGImageSourceCopyAuxiliaryDataInfoAtIndex(...)
let matteData = AVPortraitEffectsMatte(fromDictionaryRepresentation: auxiliaryData)
```

Loading

AVPortraitEffectsMatte

```
let auxiliaryData = CGImageSourceCopyAuxiliaryDataInfoAtIndex(...)
let matteData = AVPortraitEffectsMatte(fromDictionaryRepresentation: auxiliaryData)
```

Loading

AVPortraitEffectsMatte

```
let auxiliaryData = CGImageSourceCopyAuxiliaryDataInfoAtIndex(...)
let matteData = AVPortraitEffectsMatte(fromDictionaryRepresentation: auxiliaryData)
```

```
let buf = matteData.mattingImage; // CVPixelBuffer
let format = matteData.pixelFormatType; // OSType kCVPixelFormatType_OneComponent8
```

Loading

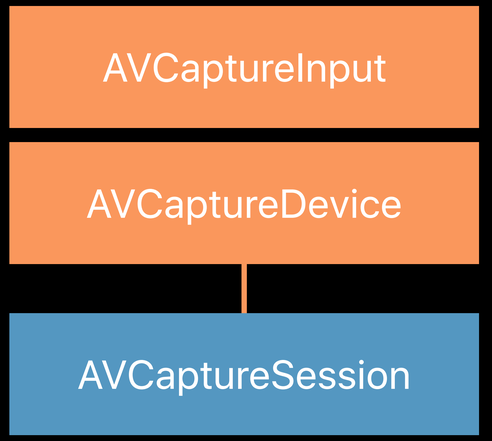
AVPortraitEffectsMatte

```
let auxiliaryData = CGImageSourceCopyAuxiliaryDataInfoAtIndex(...)
let matteData = AVPortraitEffectsMatte(fromDictionaryRepresentation: auxiliaryData)
```

```
let buf = matteData.mattingImage; // CVPixelBuffer
let format = matteData.pixelFormatType; // OSType kCVPixelFormatType_OneComponent8
```

Capture

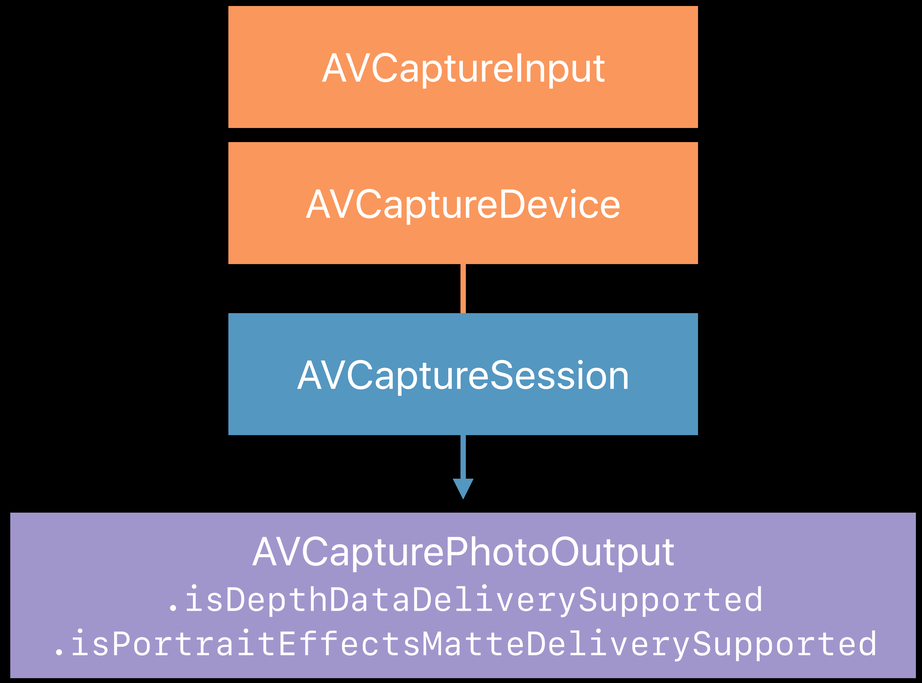
AVCapturePhoto



Capture

AVCapturePhoto

Check support

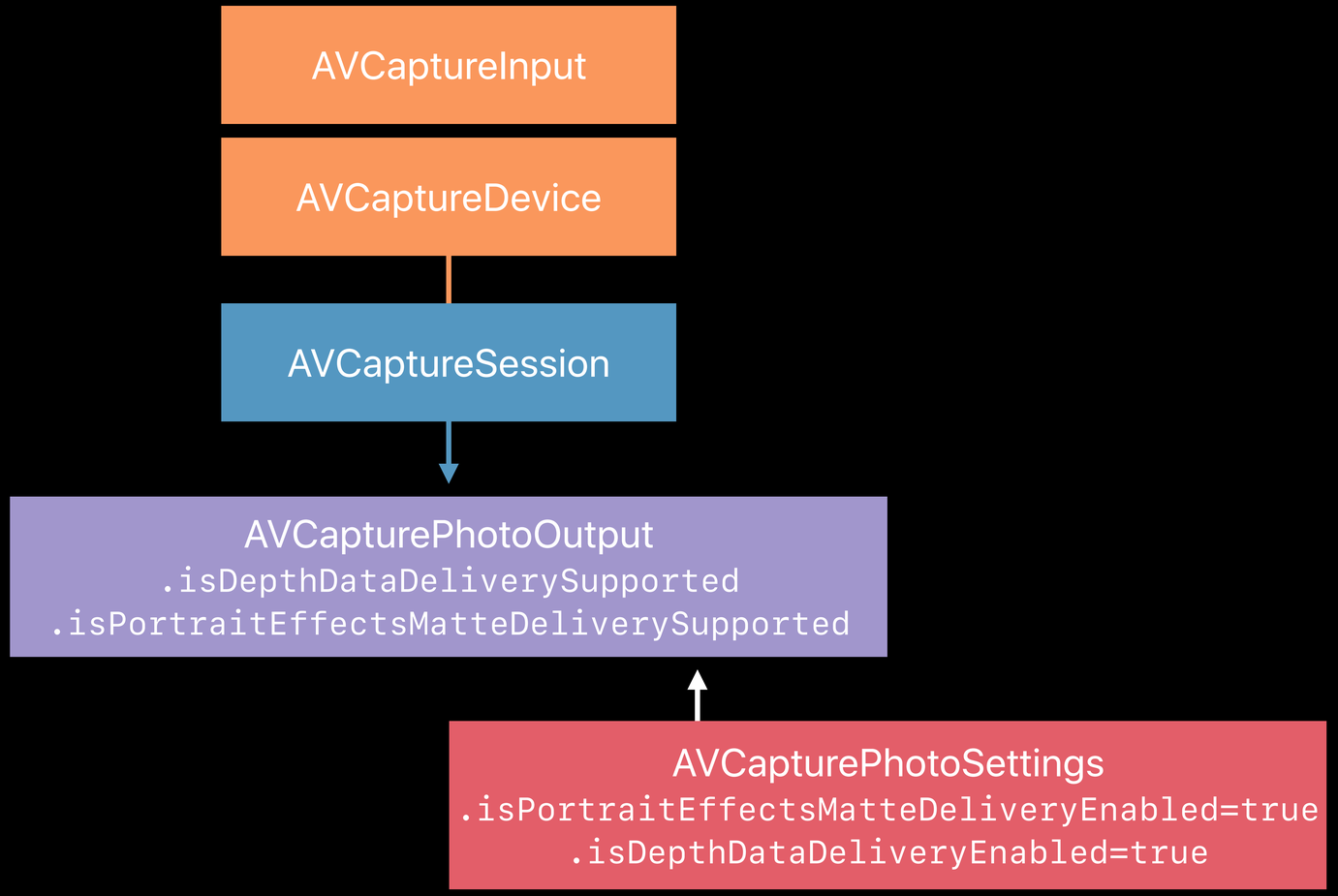


Capture

AVCapturePhoto

Check support

Enable with Depth



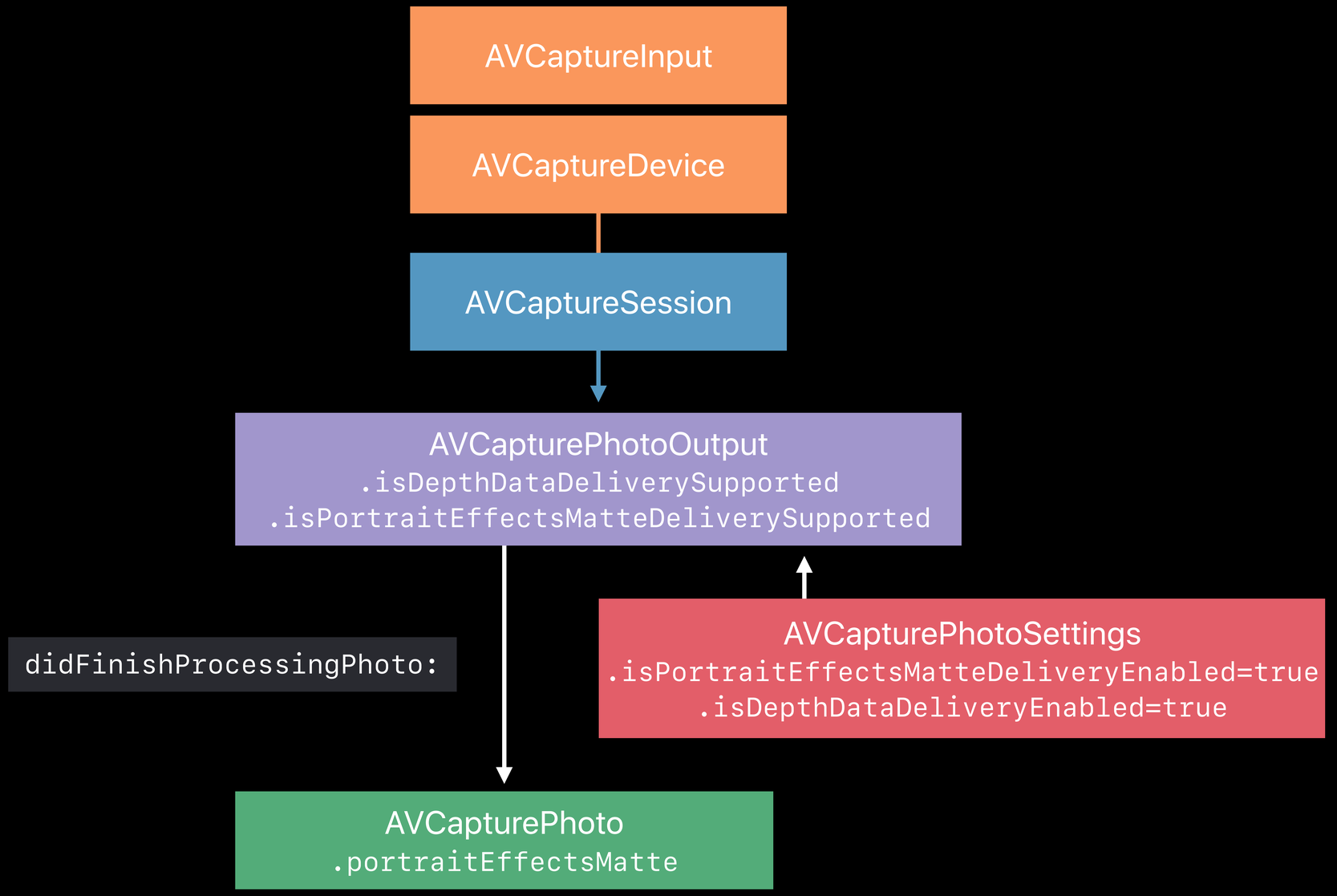
Capture

AVCapturePhoto

Check support

Enable with Depth

Get the matte



Loading and Saving

Core Image

```
let opts = [ CIImageOption.auxiliaryPortraitEffectsMatte: true ]
let matte = CIImage(imageWithContentsOfURL:url options:opts)
```

Loading and Saving

Core Image

```
let opts = [ CIImageOption.auxiliaryPortraitEffectsMatte: true ]
let matte = CIImage(imageWithContentsOfURL:url options:opts)
```

Loading and Saving

Core Image

```
let opts = [ CIImageOption.auxiliaryPortraitEffectsMatte: true ]
let matte = CIImage(imageWithContentsOfURL:url options:opts)
```

Loading and Saving

Core Image

```
let opts = [ CIImageOption.auxiliaryPortraitEffectsMatte: true ]
let matte = CIImage(imageWithContentsOfURL:url options:opts)
```

```
let opts = [ CIImageOption.portraitEffectsMatteImage : matte ]
[ctx writeHEIFRepresentationOfImage(of:image to:url format:.RGBA8
                                    colorSpace:cs options:opts error:&error]
```

Loading and Saving

Core Image

```
let opts = [ CIImageOption.auxiliaryPortraitEffectsMatte: true ]
let matte = CIImage(imageWithContentsOfURL:url options:opts)
```

```
let opts = [ CIImageOption.portraitEffectsMatteImage : matte ]
[ctx writeHEIFRepresentationOfImage(of:image to:url format:.RGBA8
                                    colorSpace:cs options:opts error:&error]
```

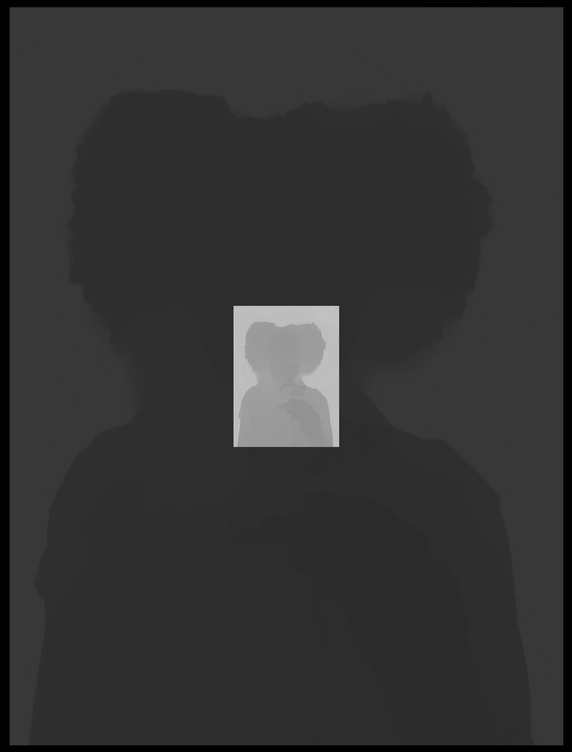
Loading and Saving

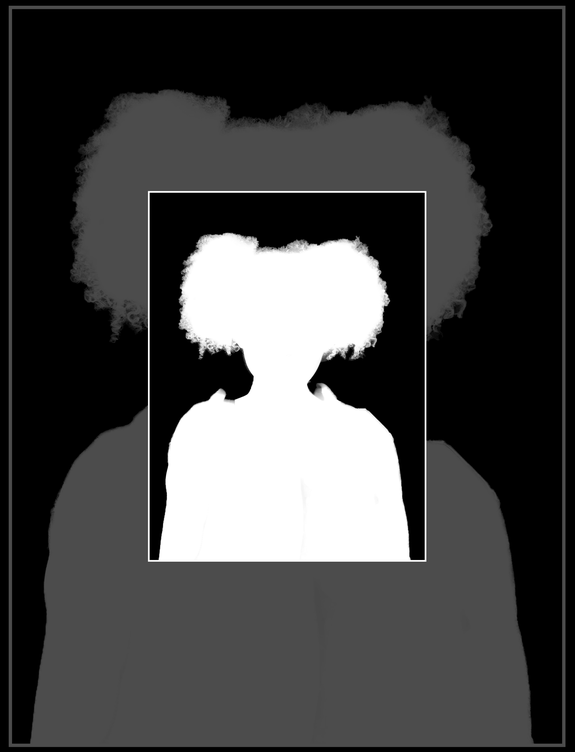
Core Image

```
let opts = [ CIImageOption.auxiliaryPortraitEffectsMatte: true ]
let matte = CIImage(imageWithContentsOfURL:url options:opts)
```

```
let opts = [ CIImageOption.portraitEffectsMatteImage : matte ]
[ctx writeHEIFRepresentationOfImage(of:image to:url format:.RGBA8
                                    colorSpace:cs options:opts error:&error]
```

	RGB	Depth	Matte
RFC	4032 x 3024	768 x 576	2016 x 1512
FFC TrueDepth	3088 x 2320	640 x 480	1544 x 1160





RGB | Depth | Matte

RFC	4032 x 3024	768 x 576	2016 x 1512
FFC TrueDepth	3088 x 2320	640 x 480	1544 x 1160

Demo

Jupyter Notebook

Related Session

Core Image: Performance, Prototyping, and Python	Executive Ballroom	Thursday 5:00PM

Real-Time Video Effects with TrueDepth

Ron Sokolovsky, Video Engineering

TrueDepth stream

Point clouds

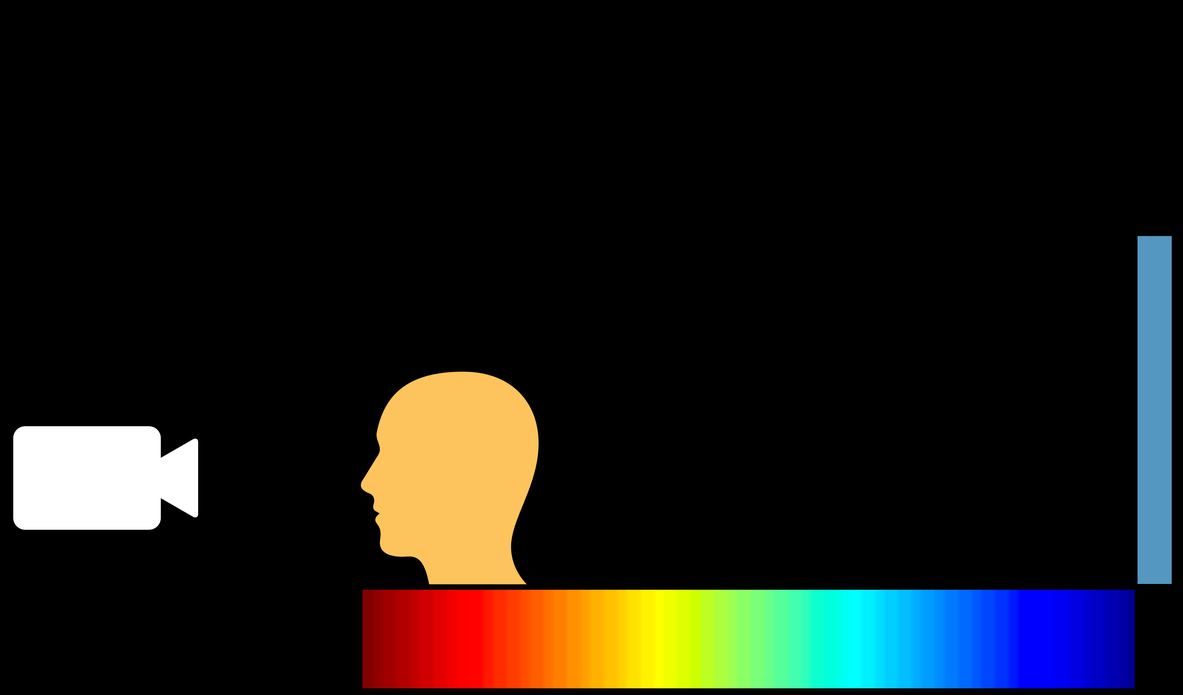
Backdrop

TrueDepth stream

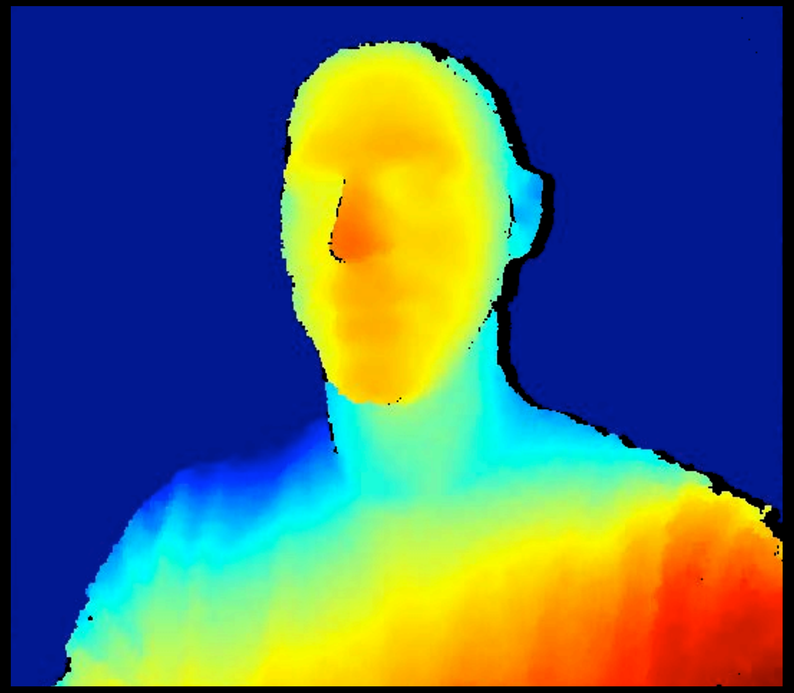
Point clouds

Backdrop

Depth Map



Minimum to maximum depth in frame

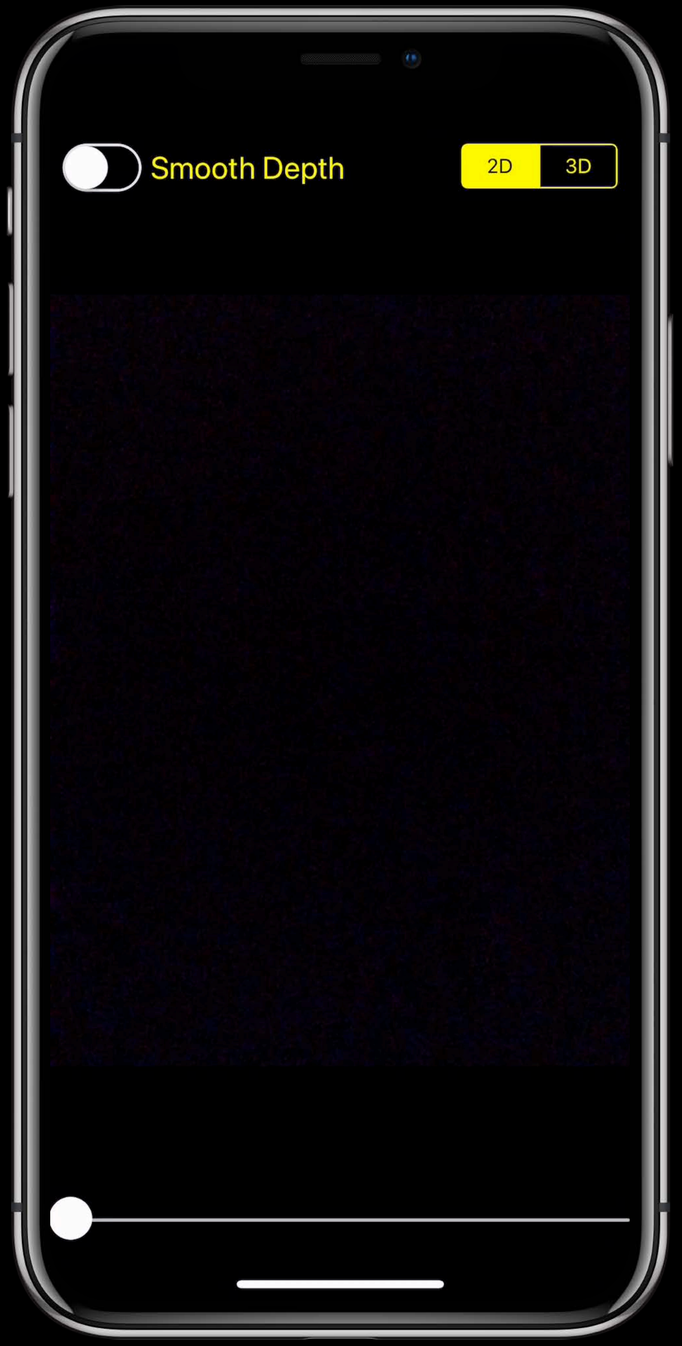


Black pixels—no depth


Smooth Depth
2D
3D

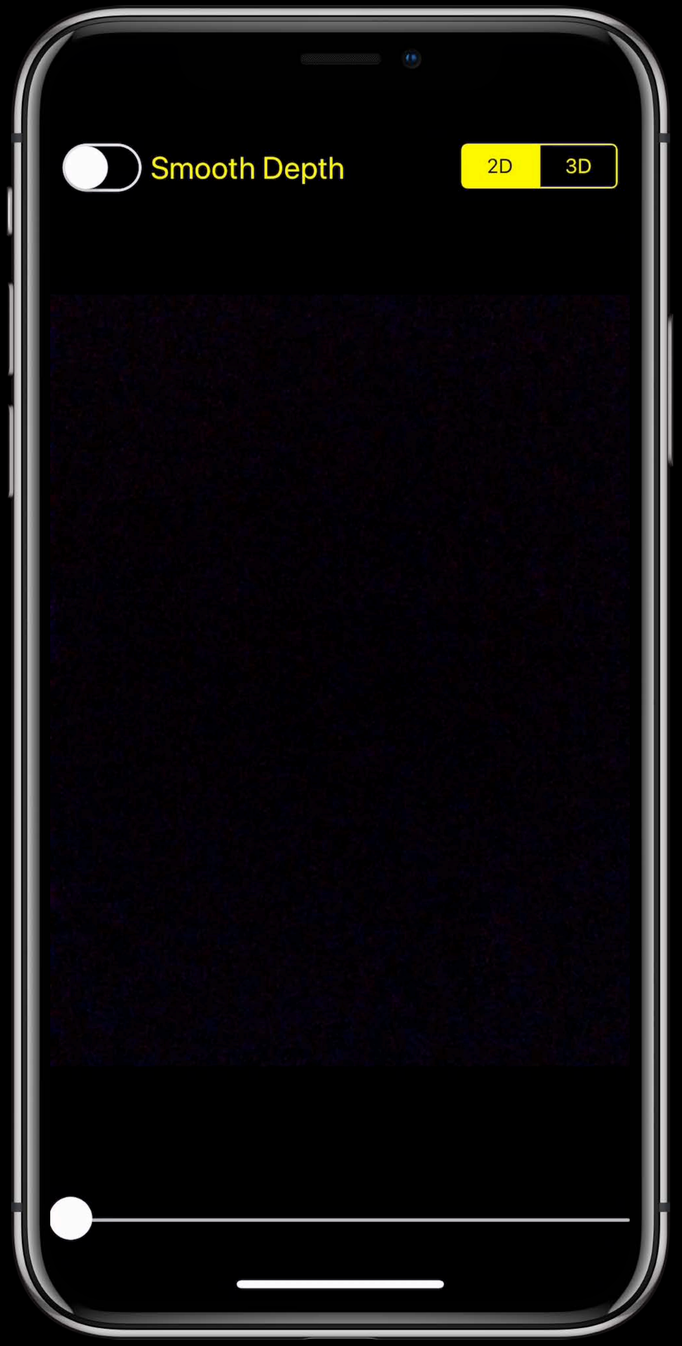
Video

TrueDepth stream



The dark side

TrueDepth stream



The dark side

TrueDepth stream

How do you add a TrueDepth stream?

```
// Discover the TrueDepth camera
let videoDeviceDiscoverySession = AVCaptureDevice.DiscoverySession(deviceTypes:
        [.builtInTrueDepthCamera], mediaType: .video, position: .front)

// Initialize capture device input
let videoDeviceInput = try AVCaptureDeviceInput(device:
        videoDeviceDiscoverySession.devices.first!)

// Add a depth data output to the session
let depthDataOutput = AVCaptureDepthDataOutput()
session.addOutput(depthDataOutput)
depthDataOutput.setDelegate(self, callbackQueue: dataOutputQueue)
```

```
// Discover the TrueDepth camera
let videoDeviceDiscoverySession = AVCaptureDevice.DiscoverySession(deviceTypes:
        [.builtInTrueDepthCamera], mediaType: .video, position: .front)

// Initialize capture device input
let videoDeviceInput = try AVCaptureDeviceInput(device:
        videoDeviceDiscoverySession.devices.first!)

// Add a depth data output to the session
let depthDataOutput = AVCaptureDepthDataOutput()
session.addOutput(depthDataOutput)
depthDataOutput.setDelegate(self, callbackQueue: dataOutputQueue)
```

```
// Discover the TrueDepth camera
let videoDeviceDiscoverySession = AVCaptureDevice.DiscoverySession(deviceTypes:
        [.builtInTrueDepthCamera], mediaType: .video, position: .front)

// Initialize capture device input
let videoDeviceInput = try AVCaptureDeviceInput(device:
        videoDeviceDiscoverySession.devices.first!)

// Add a depth data output to the session
let depthDataOutput = AVCaptureDepthDataOutput()
session.addOutput(depthDataOutput)
depthDataOutput.setDelegate(self, callbackQueue: dataOutputQueue)
```

```
// Discover the TrueDepth camera
let videoDeviceDiscoverySession = AVCaptureDevice.DiscoverySession(deviceTypes:
        [.builtInTrueDepthCamera], mediaType: .video, position: .front)

// Initialize capture device input
let videoDeviceInput = try AVCaptureDeviceInput(device:
        videoDeviceDiscoverySession.devices.first!)

// Add a depth data output to the session
let depthDataOutput = AVCaptureDepthDataOutput()
session.addOutput(depthDataOutput)
depthDataOutput.setDelegate(self, callbackQueue: dataOutputQueue)
```

```
// Discover the TrueDepth camera
let videoDeviceDiscoverySession = AVCaptureDevice.DiscoverySession(deviceTypes:
        [.builtInTrueDepthCamera], mediaType: .video, position: .front)

// Initialize capture device input
let videoDeviceInput = try AVCaptureDeviceInput(device:
        videoDeviceDiscoverySession.devices.first!)

// Add a depth data output to the session
let depthDataOutput = AVCaptureDepthDataOutput()
session.addOutput(depthDataOutput)
depthDataOutput.setDelegate(self, callbackQueue: dataOutputQueue)
```

Disparity or Depth?

Disparity usually yields better results

Depth values are intuitive, but the error goes with Z^2

```
let disparitymap = depthmap.applyingFilter("CIDepthToDisparity", parameters: nil)
```

Resolution

sessionPreset	Aspect Ratio	RGB	TrueDepth
.photo	4:3	1440x1080	640x480
.hd1920x1080	16:9	1920x1080	640x360
.hd1280x720	16:9	1280x720	640x360
.vga640x480	4:3	640x480	640x480

Resolution

sessionPreset	Aspect Ratio	RGB	TrueDepth
.photo	4:3	1440x1080	640x480
.hd1920x1080	16:9	1920x1080	640x360
.hd1280x720	16:9	1280x720	640x360
.vga640x480	4:3	640x480	640x480

Test System Pressure

High resolutions + fast frame rate + processing + duration

```
AVCaptureDevice.SystemPressureState.Level
```

Nominal

Fair

Serious

Critical

Shutdown

Monitor System Pressure and Adapt

Prevent AVCaptureDevice shutdown

Reduce frame rate when level is increasing:

Nominal (30 fps) ⟶ Serious (15 fps)

Nominal (30 fps) ⟶ Fair (24 fps) ⟶ Serious (20 fps) ⟶ Critical (15 fps)

```
captureDevice.lockForConfiguration()
captureDevice.activeVideoMinFrameDuration = CMTimeMake(1, 15) // 15 fps
captureDevice.unlockForConfiguration()
```

Holey depth map, Batman!

Filtered Depth Map

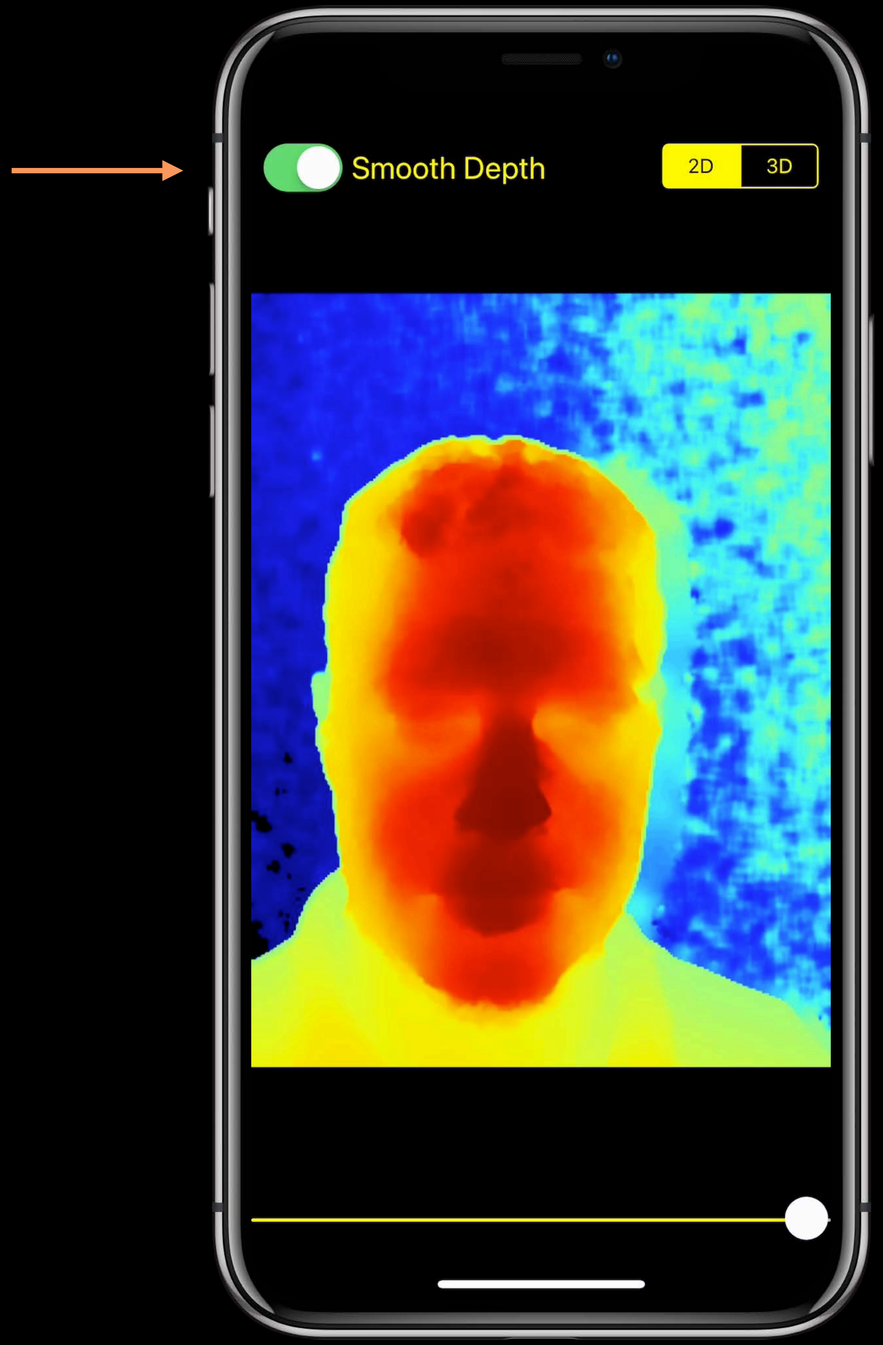
Smoothed, spatially and temporally

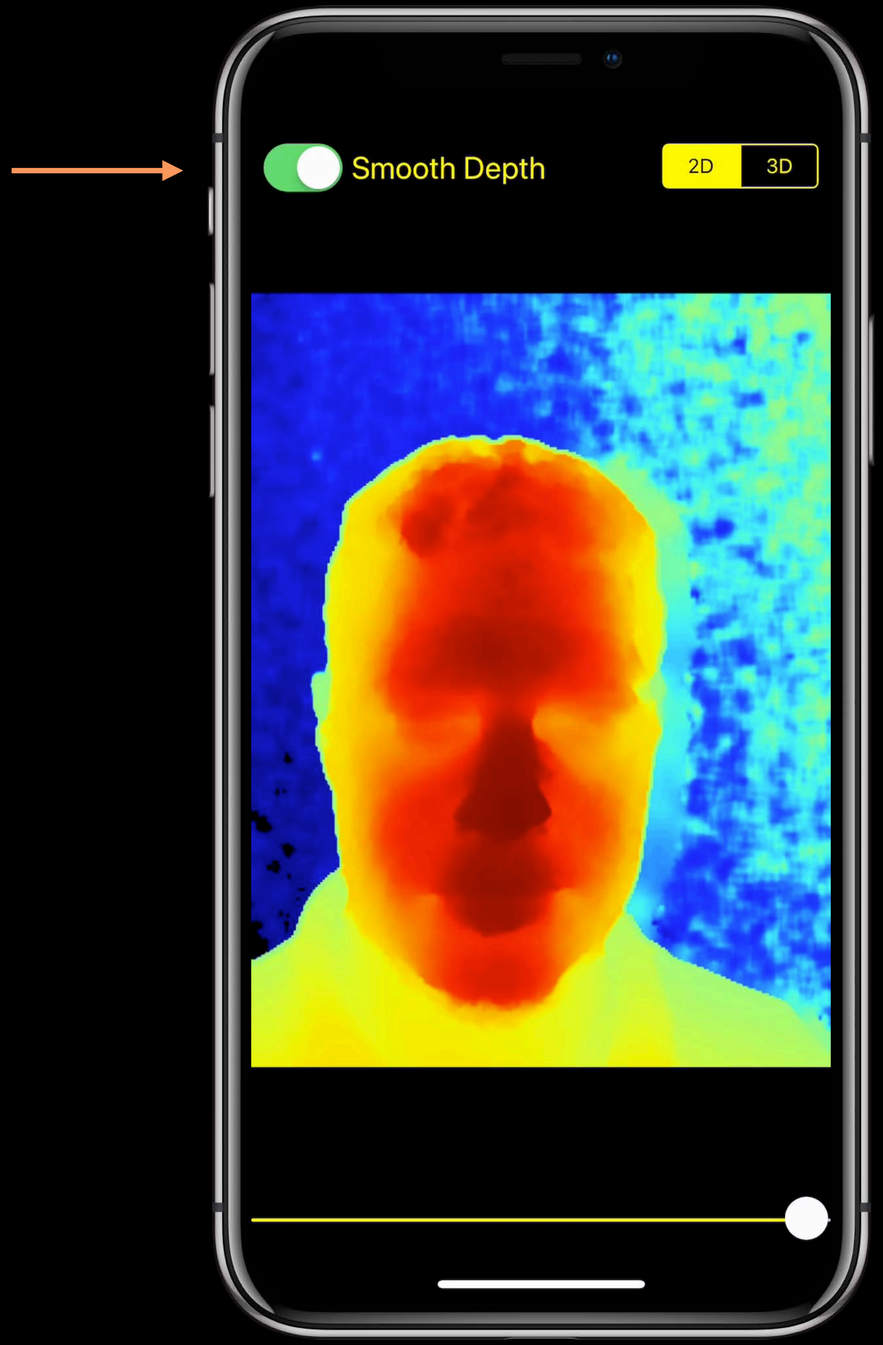
Holes filled, using RGB

For photography/segmentation applications

iOS 12: spatial resolution of filtering improved from iOS 11

```
depthDataOutput.isFilteringEnabled = true // default
```


Smooth Depth
2D
3D


Smooth Depth
2D
3D

Using Raw Depth Data

Use for point clouds, or real world measurements

No depth value == 0

Watch out for averaging/downsampling

```
depthDataOutput.isFilteringEnabled = false
```

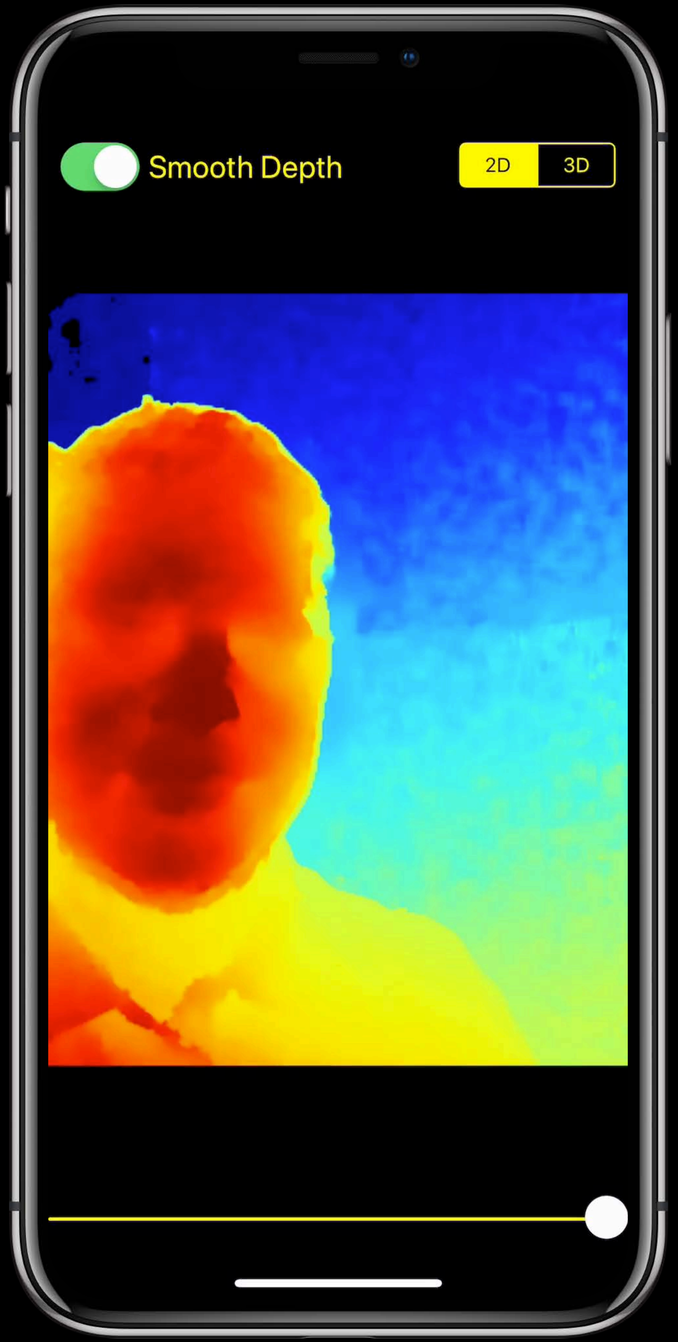
Depth Map Holes

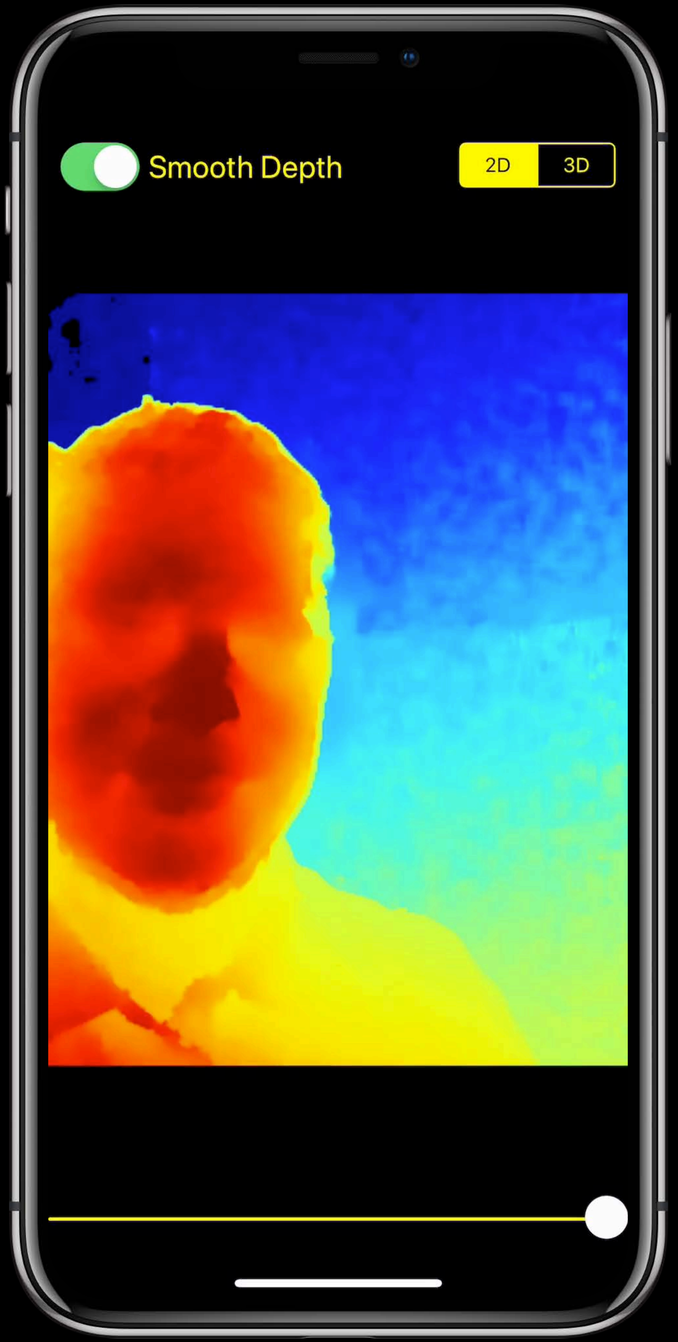
The TrueDepth camera detects most objects up to about five meters

Low reflectivity materials absorb light


Smooth Depth
2D
3D


Smooth Depth
2D
3D


Smooth Depth
2D
3D


Smooth Depth
2D
3D

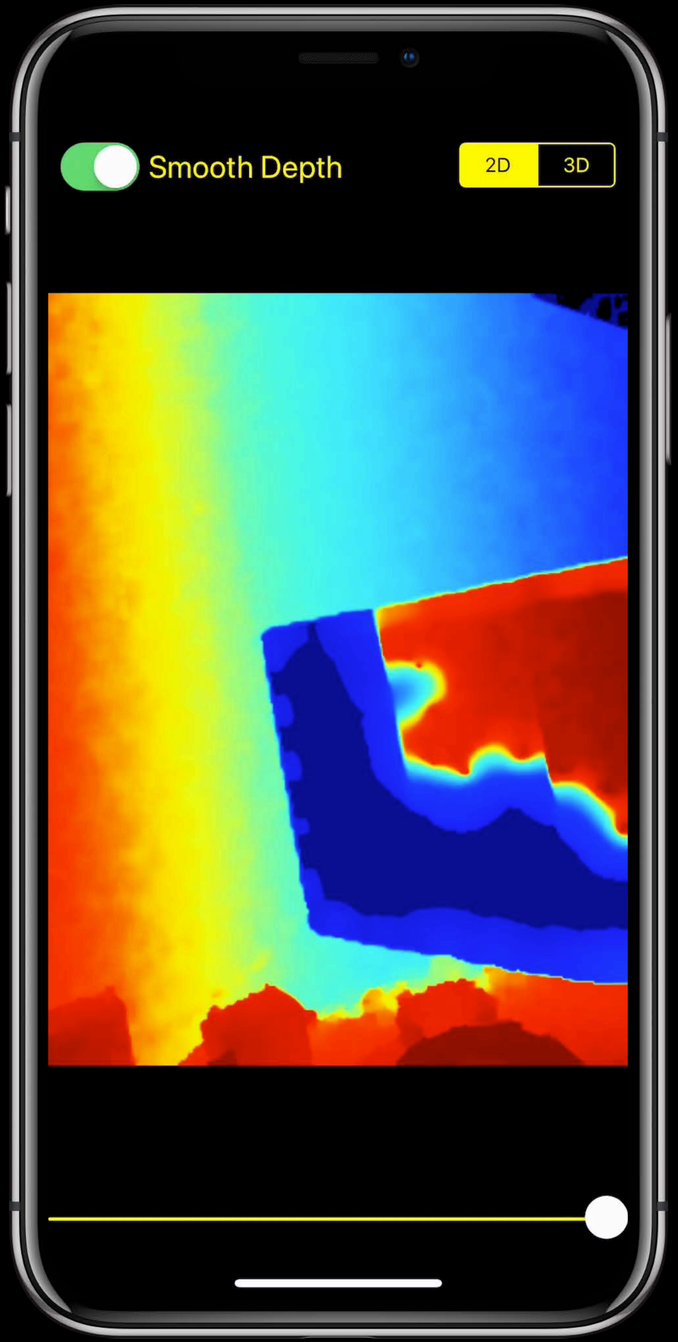
Depth Map Holes

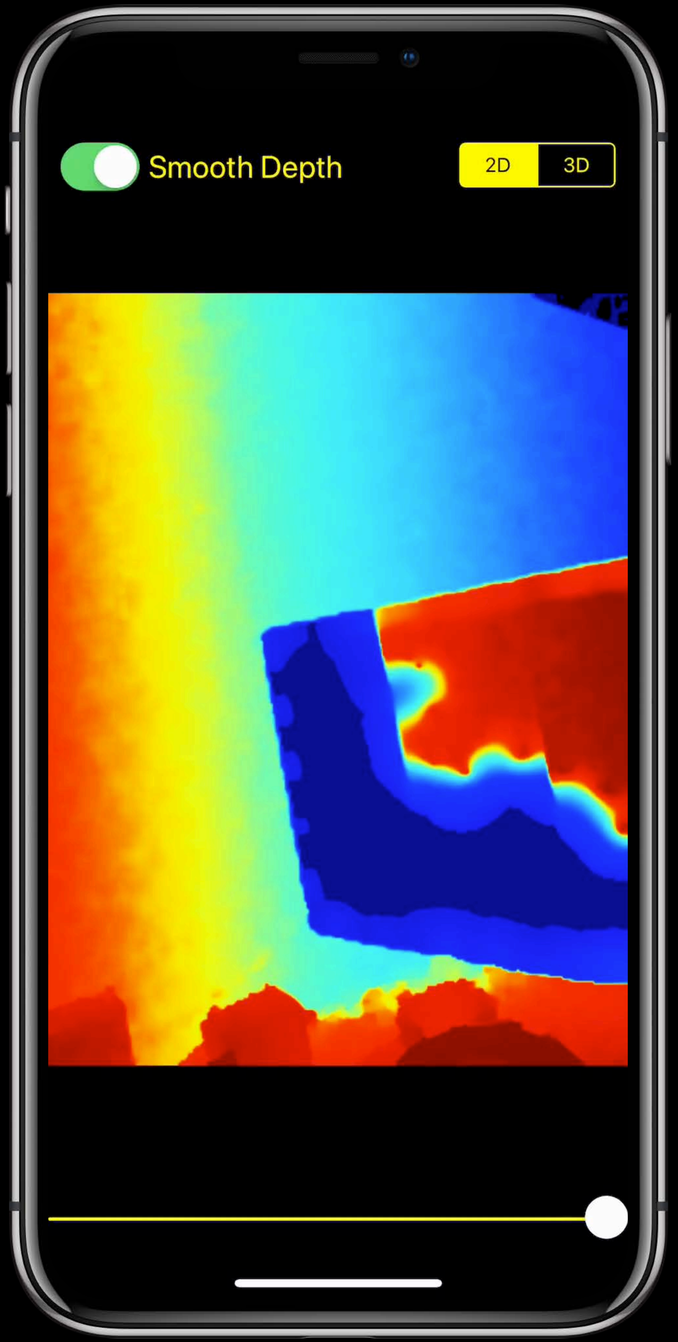
Reflectivity

Specular materials reflect light only from certain directions


Smooth Depth
2D
3D


Smooth Depth
2D
3D


Smooth Depth
2D
3D


Smooth Depth
2D
3D

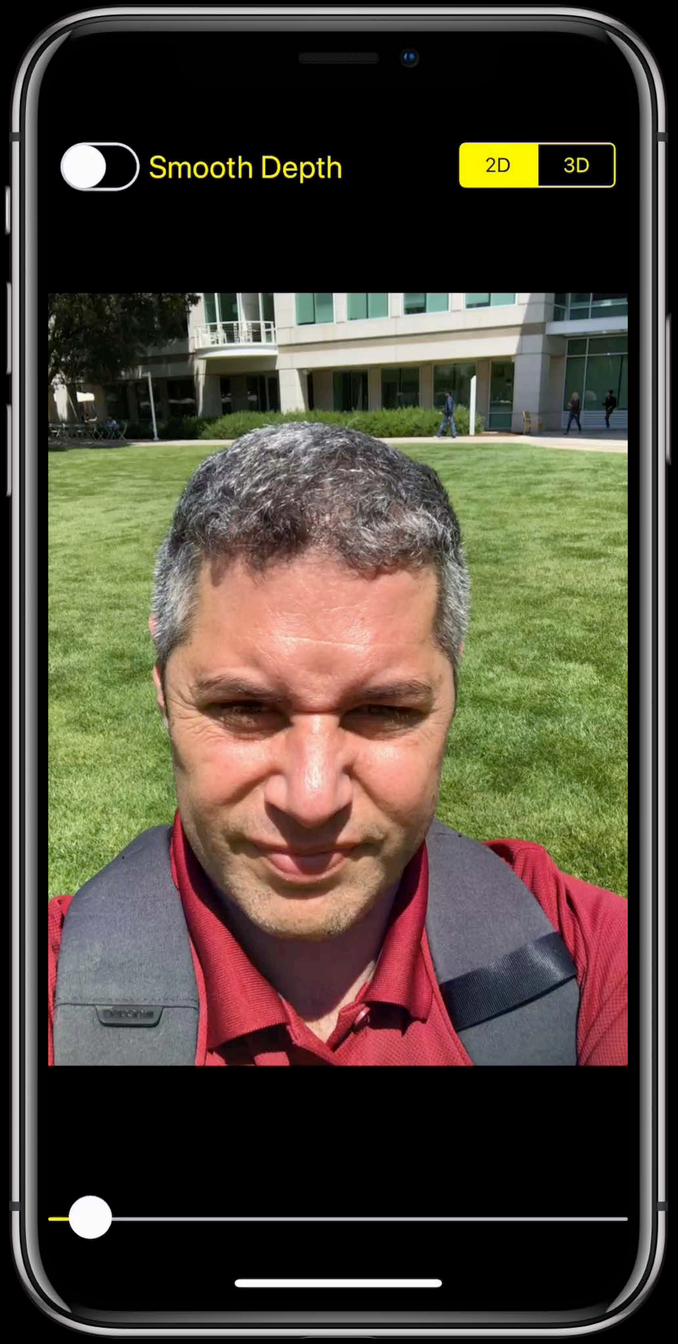
Depth Map Holes

Reflectivity

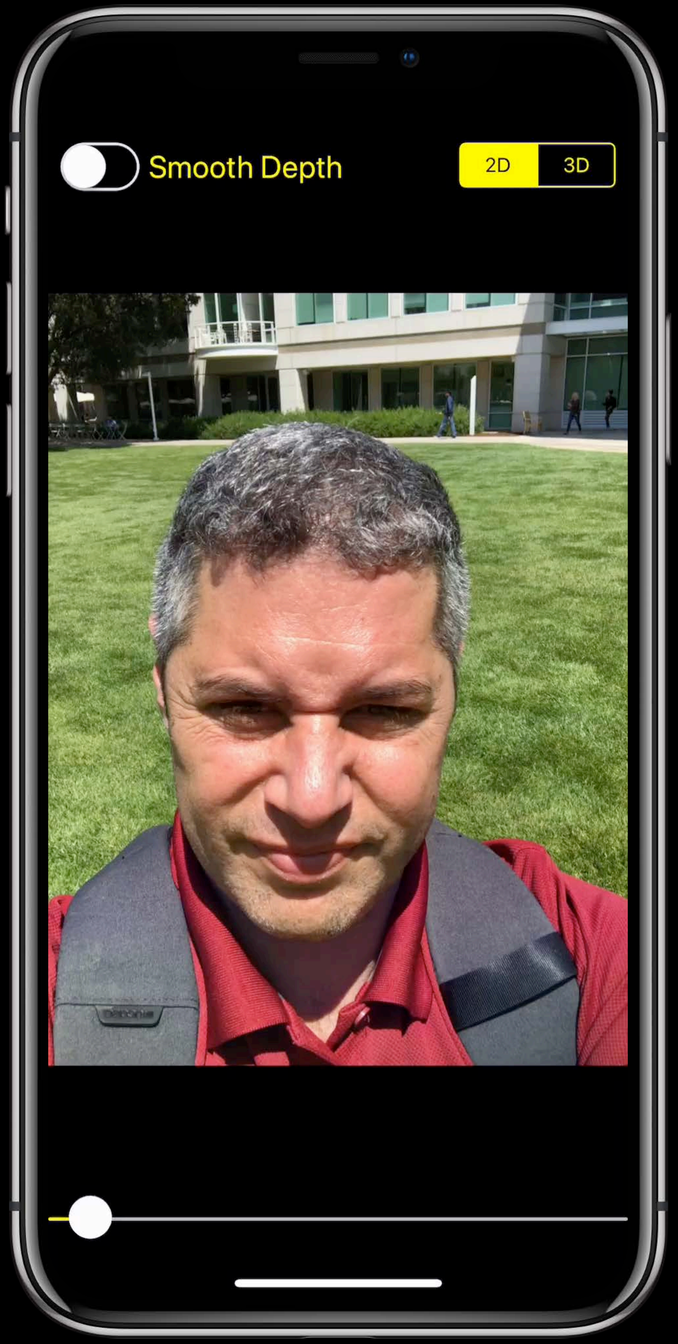
Specular

Outdoor conditions are more challenging

Outdoor



Outdoor



Depth Map Holes

Reflectivity

Specular

Outdoor

Shadow from parallax between projector and camera

Shadows

Shadows

Shadows

Demo

TrueDepthStreamer

TrueDepth stream

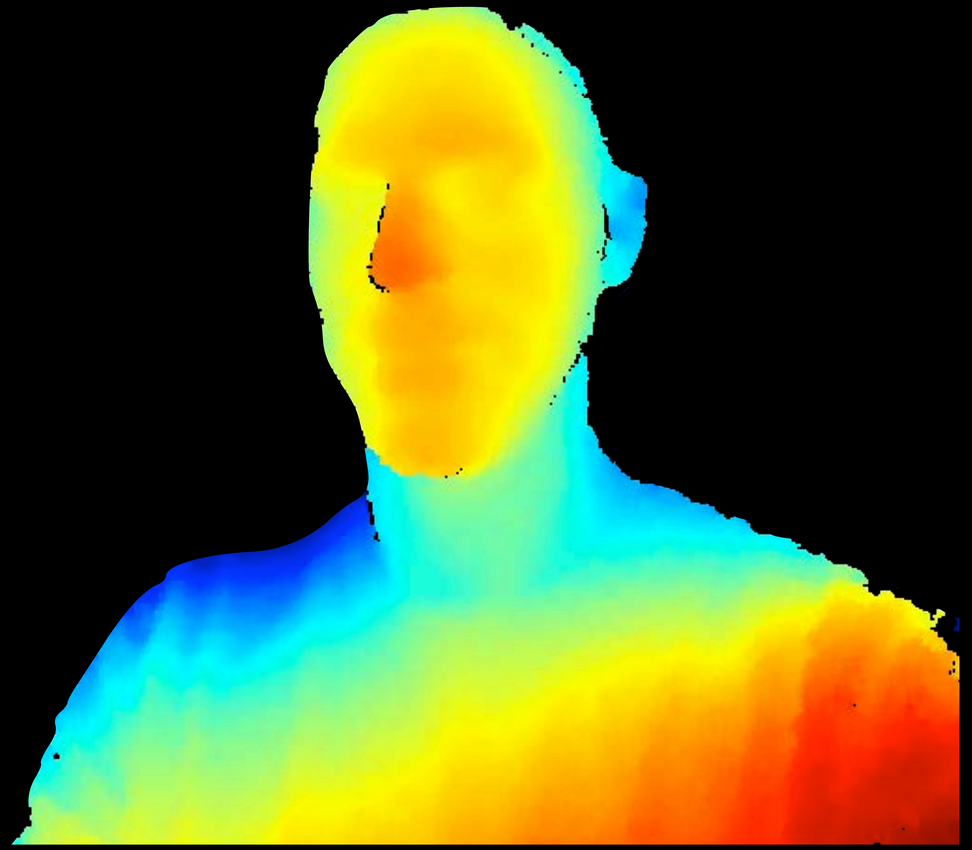
Point clouds

Backdrop

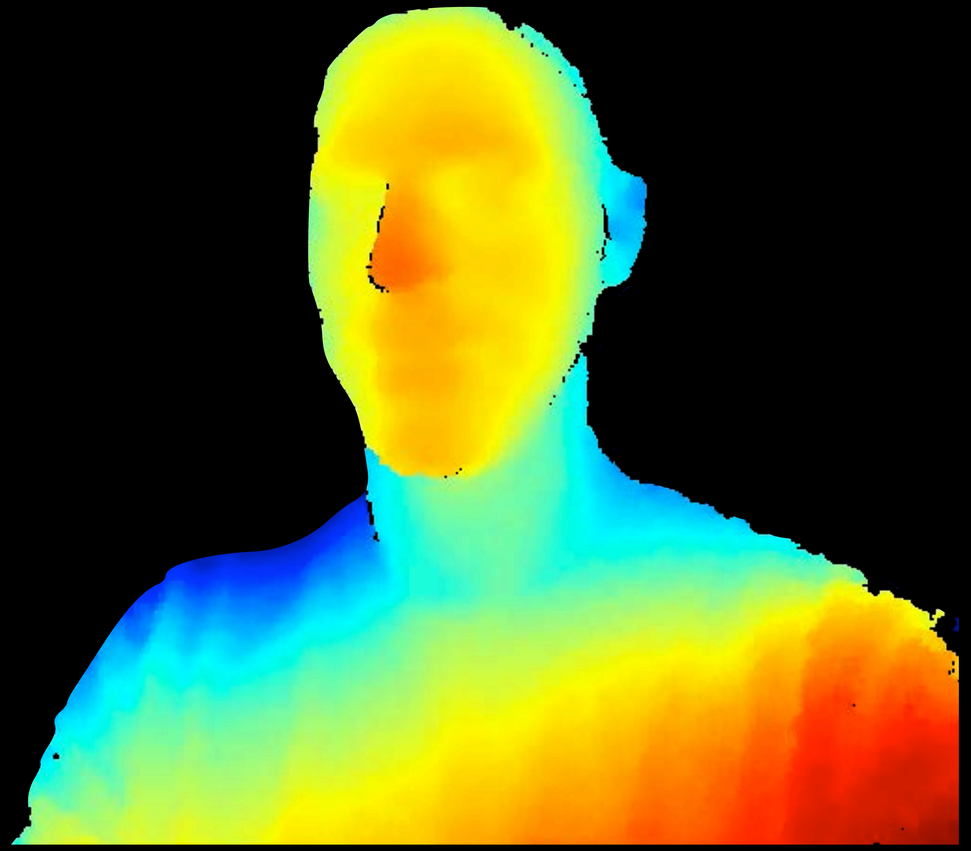
TrueDepth stream

Point clouds

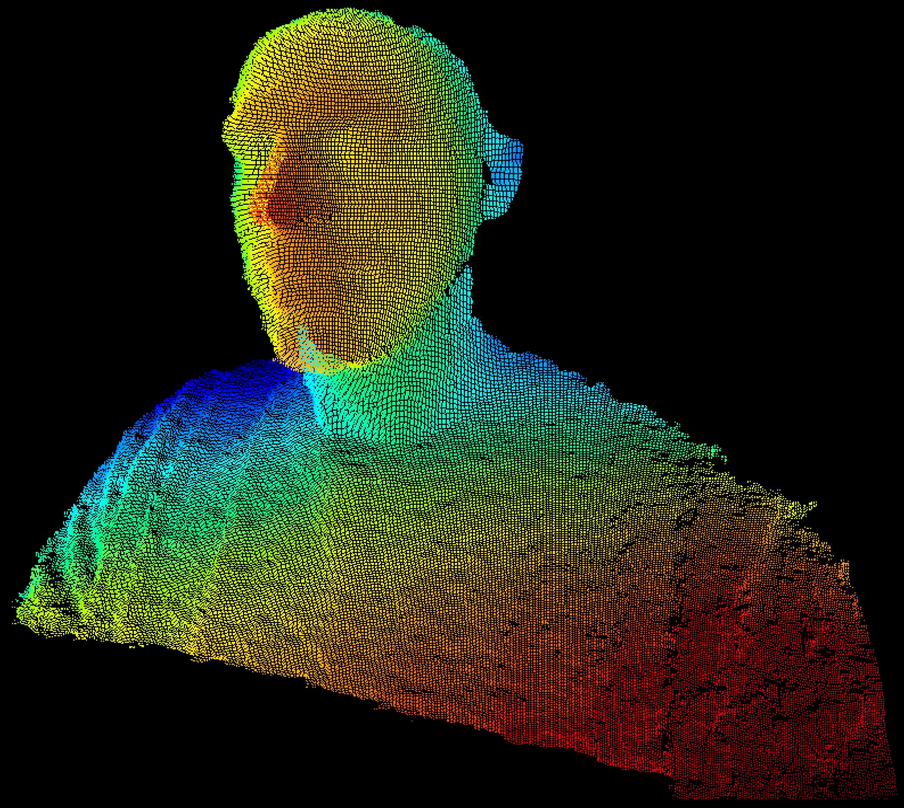
Backdrop



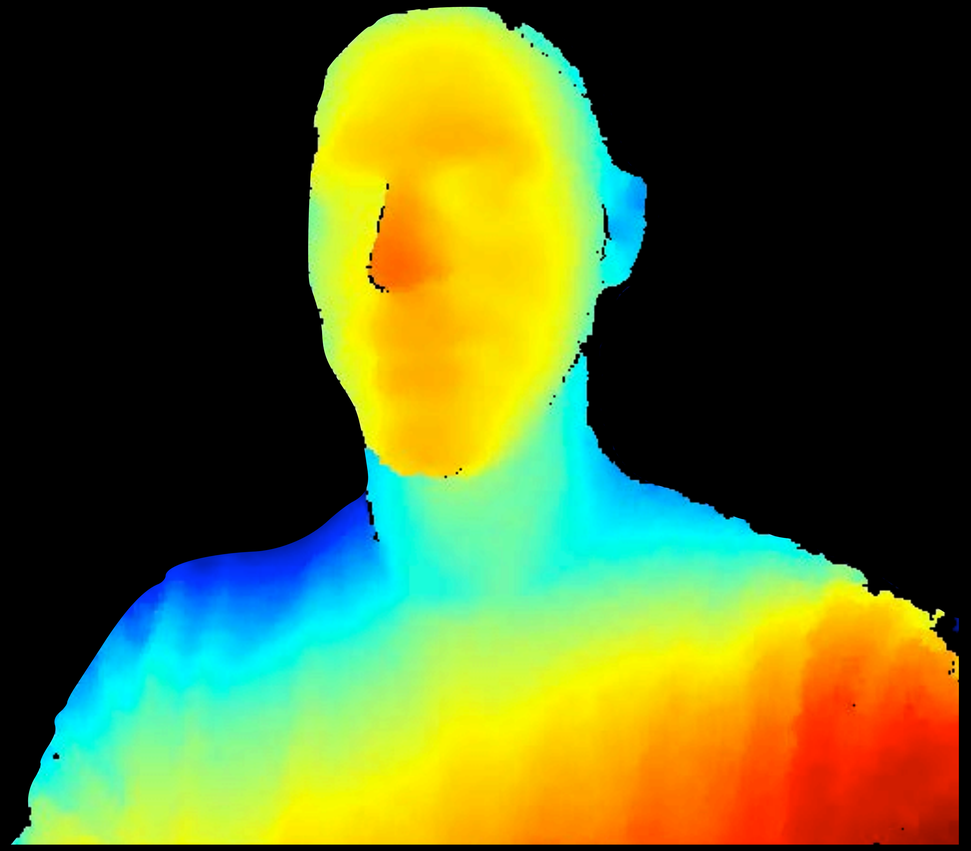
Depth Map
Z = func(U, V)



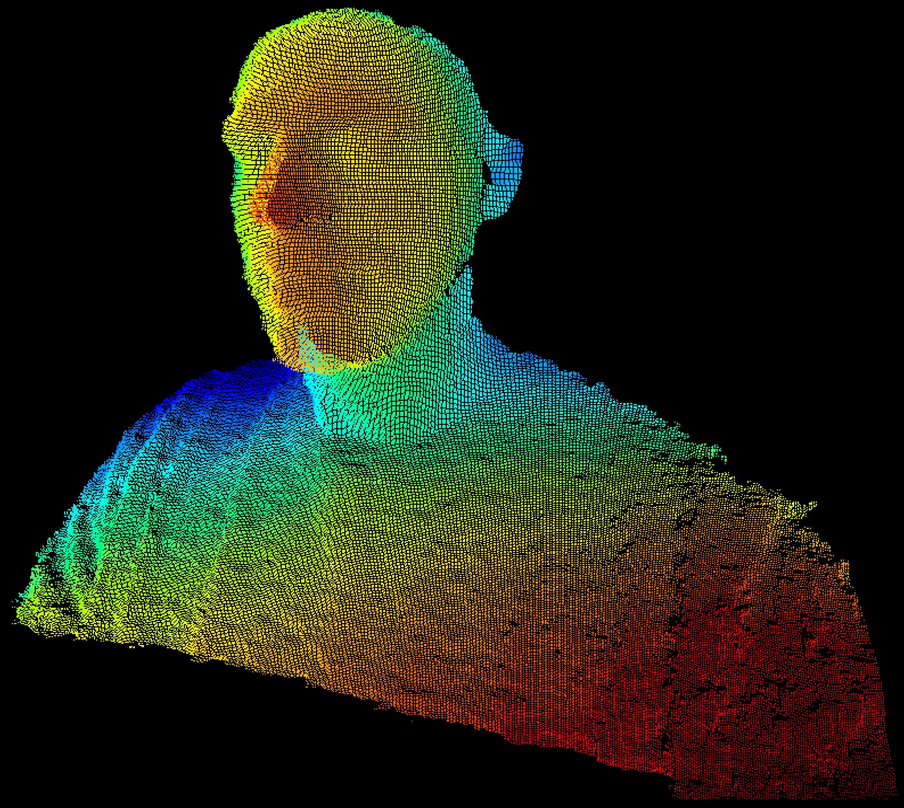
Depth Map
Z = func(U, V)



Point Cloud
(X, Y, Z)



Depth Map
Z = func(U, V)



Point Cloud
(X, Y, Z)

From Depth Map to Point Cloud

$$X = (U - X_0) \times Z / f_x$$

$$Y = (V - Y_0) \times Z / f_y$$

Intrinsics Matrix

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

From Depth Map to Point Cloud

$$X = (U - X_0) \times Z / f_x$$

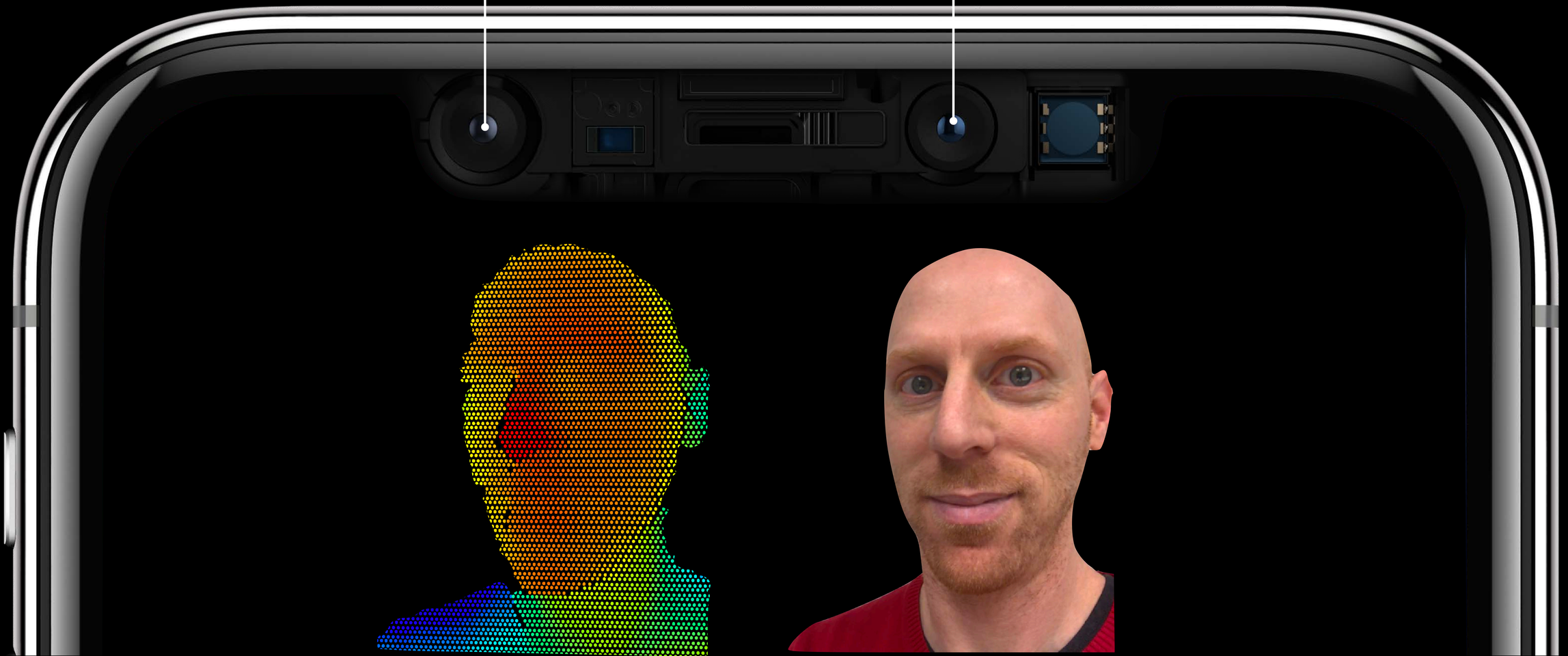
$$Y = (V - Y_0) \times Z / f_y$$

Intrinsics Matrix

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

```
matrix_float3x3 intrinsicsdepthData.cameraCalibrationData.intrinsicMatrix;
```

RGB-D



RGB-D

Metal Graphic Shaders

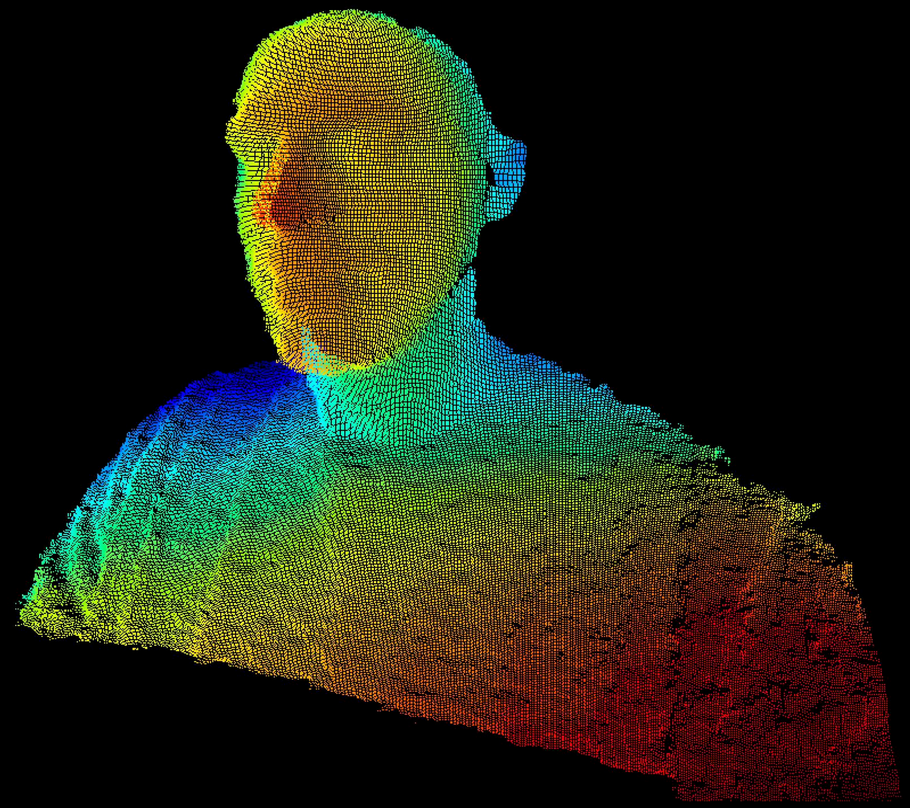
Vertex Shader

Point location

Z = func(U, V)

Transform to (X, Y, Z)

Reproject with view matrix



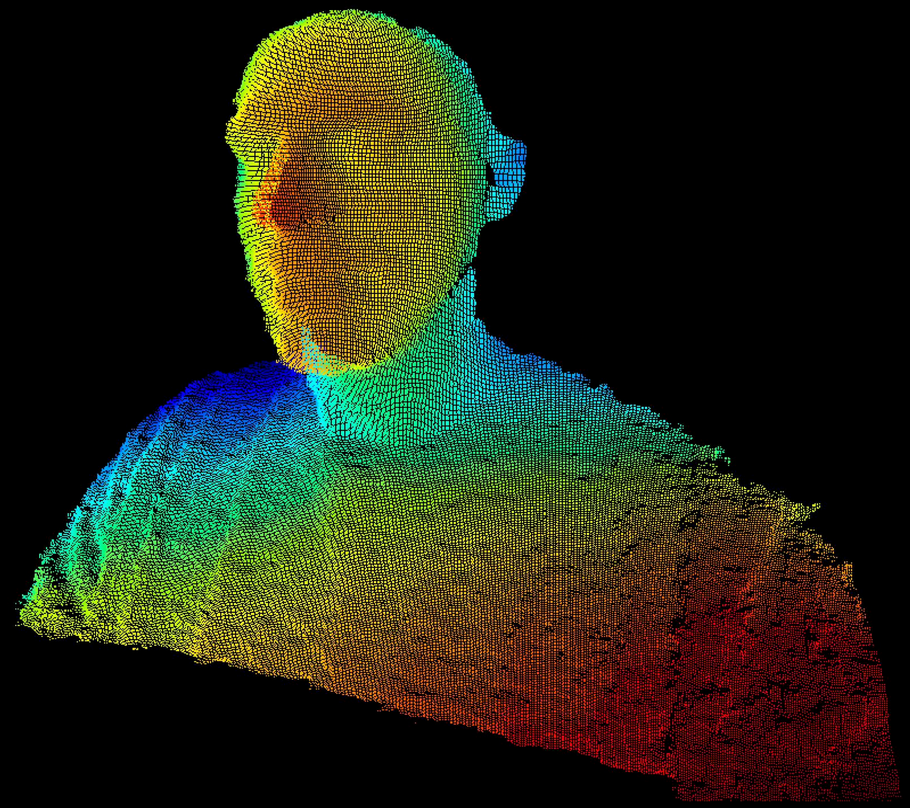
Vertex Shader

Point location

Z = func(U, V)

Transform to (X, Y, Z)

Reproject with view matrix



Fragment Shader

Point color

Get vertex info

Discard holes (Z == 0)

Apply color from RGB frame at vertex coordinates

Fragment Shader

Point color

Get vertex info

Discard holes (Z == 0)

Apply color from RGB frame at vertex coordinates

TrueDepth stream

Point clouds

Backdrop

TrueDepth stream

Point clouds

Backdrop

Demo

Backdrop

Per-Frame, Real-Time Processing

Detect a face

Create binary foreground mask, smooth, and upscale

Upscale foreground to background and blend

Per-Frame, Real-Time Processing

Detect a face

Create binary foreground mask, smooth, and upscale

Upscale foreground to background and blend

Resize background to video resolution once:

- Not upscaling foreground
- Blending at lower resolution

Getting Face Metadata

Until a face is detected, use default face depth of 0.5 meters

Transform center of face to depth map coordinates, and get its depth

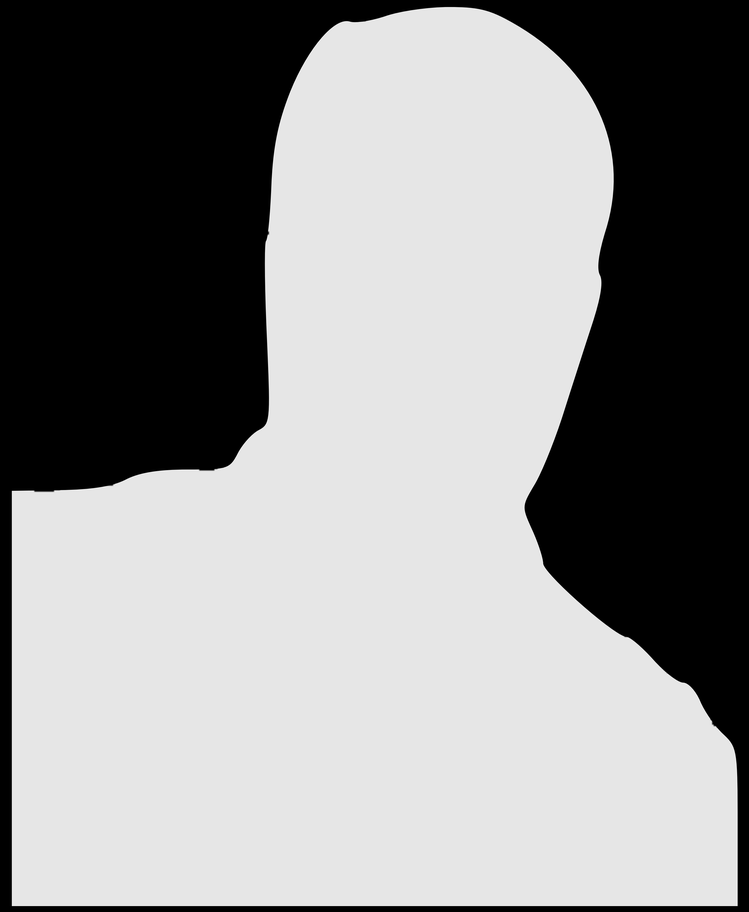
```
metadataOutput.metadataObjectTypes = [AVMetadataObject.ObjectType.face]
```

Binary Foreground Mask

Threshold = face depth + 0.25 meters

Binary mask:

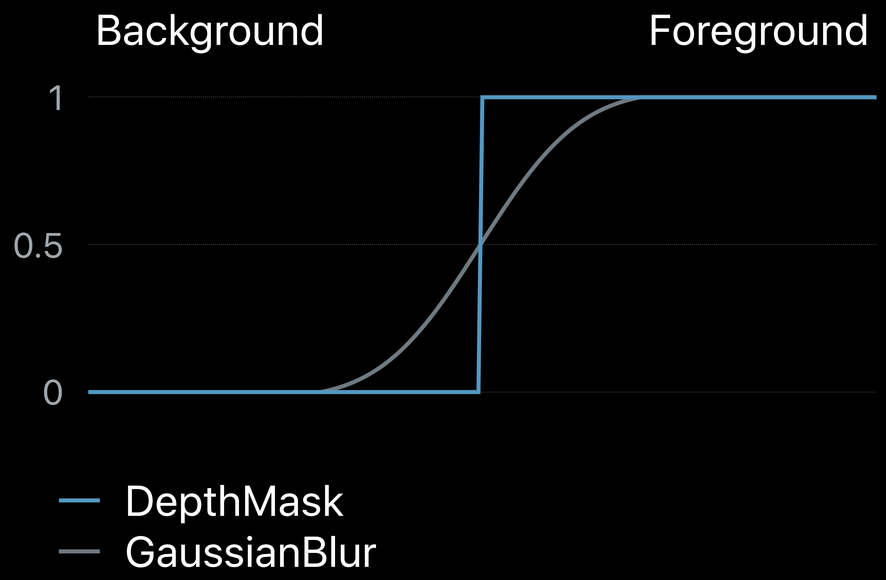
- Foreground = 1
- Background = 0

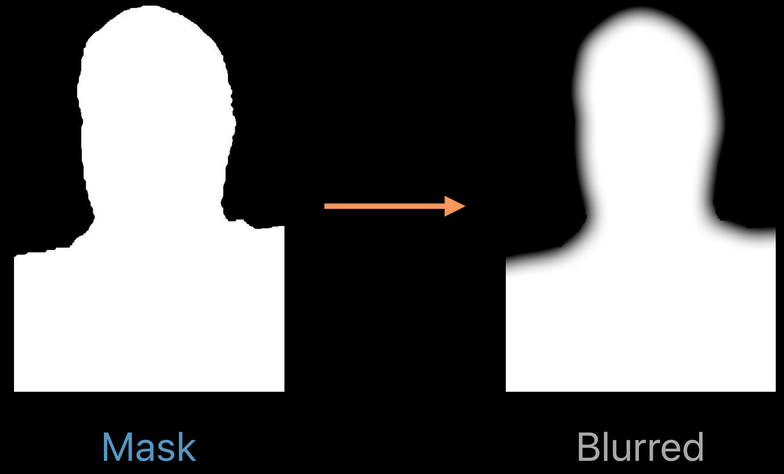


Binary Mask

Binary Mask

Gaussian Blur

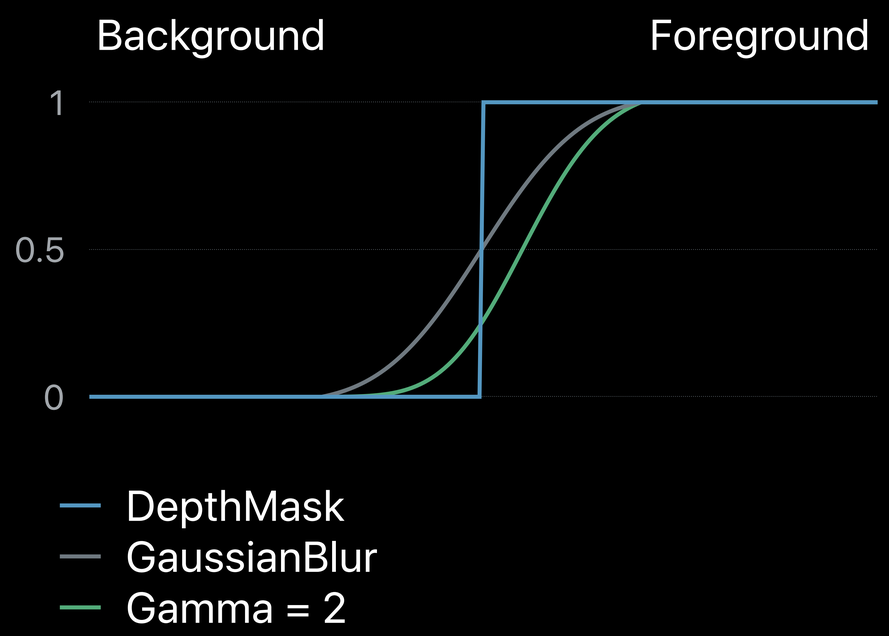


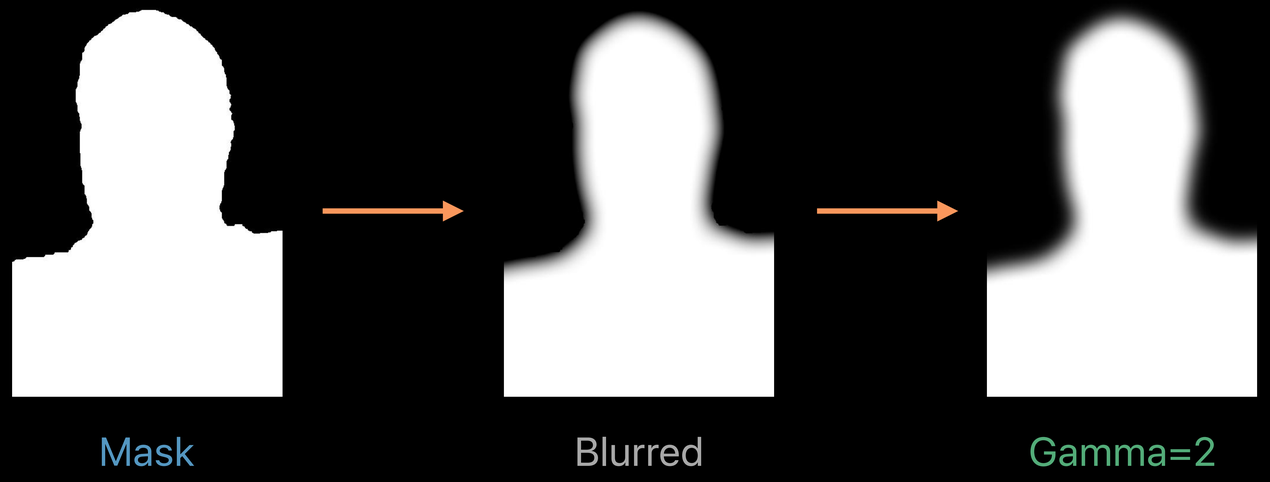


Blur Radius

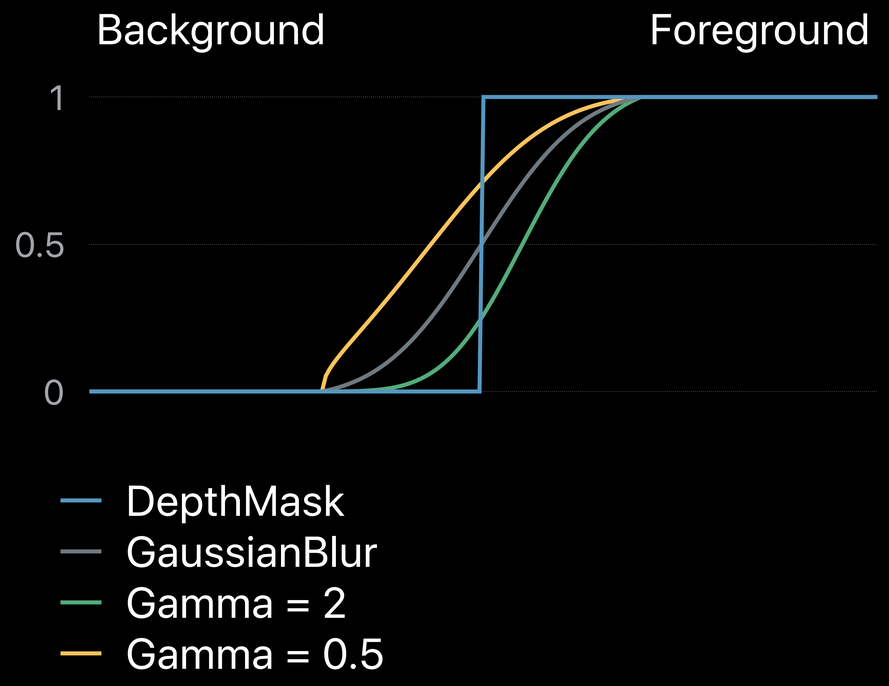
Blur Radius

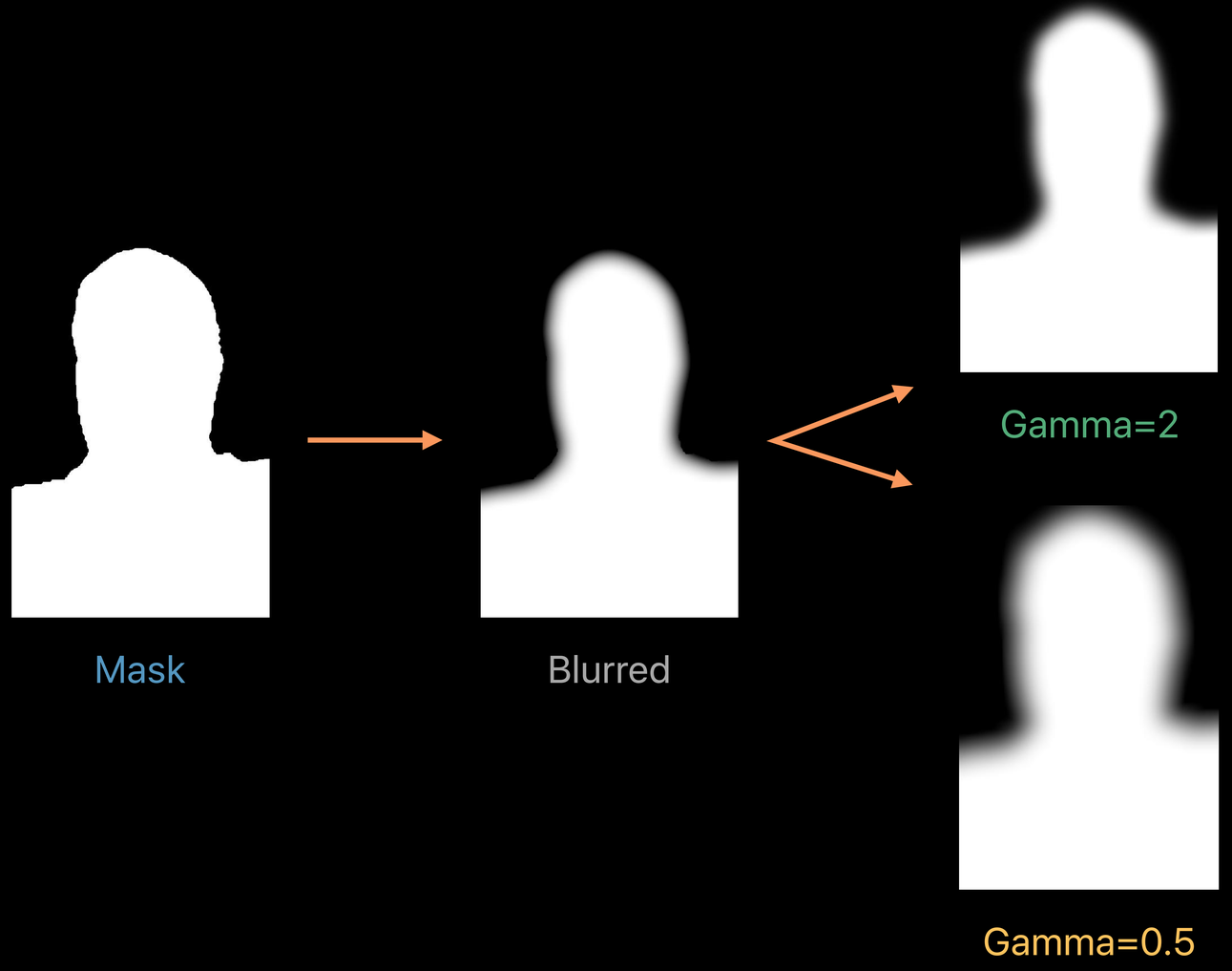
Gamma Adjust





Gamma Adjust





Adjusting Gamma

Adjusting Gamma

Binary Mask

Alpha matte

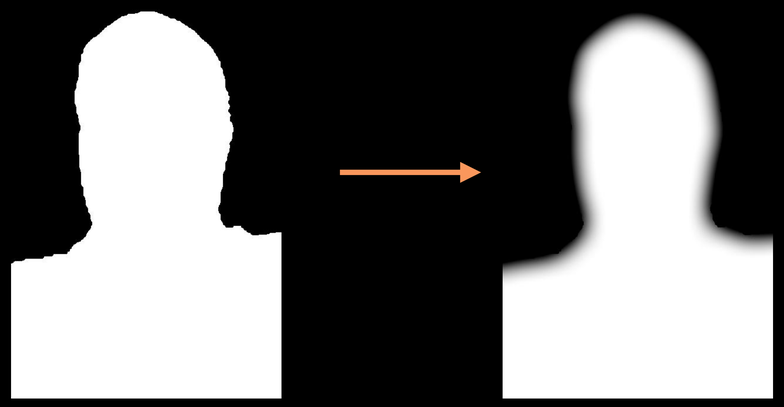
```
let matte = depthMask.applyingFilter("CIGaussianBlur", parameters: ["inputRadius": 20])
    .applyingFilter("CIGammaAdjust", parameters: ["inputPower": 2])
    .applyingFilter(“CIBicubicScaleTransform”, parameters:“inputScale”: RGBwidth/depthWidth])
```

Mask

Binary Mask

Alpha matte

```
let matte = depthMask.applyingFilter("CIGaussianBlur", parameters: ["inputRadius": 20])
    .applyingFilter("CIGammaAdjust", parameters: ["inputPower": 2])
    .applyingFilter(“CIBicubicScaleTransform”, parameters:“inputScale”: RGBwidth/depthWidth])
```



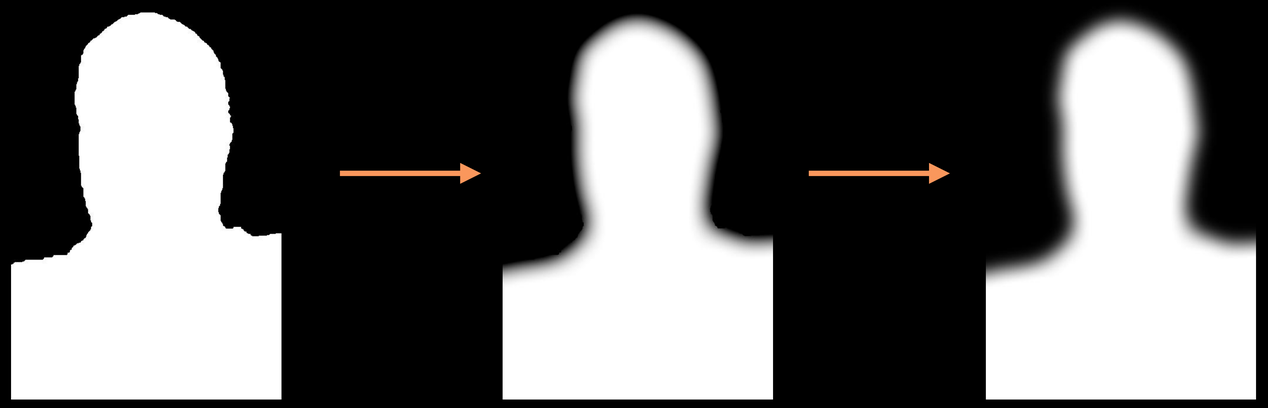
Mask

Blurred

Binary Mask

Alpha matte

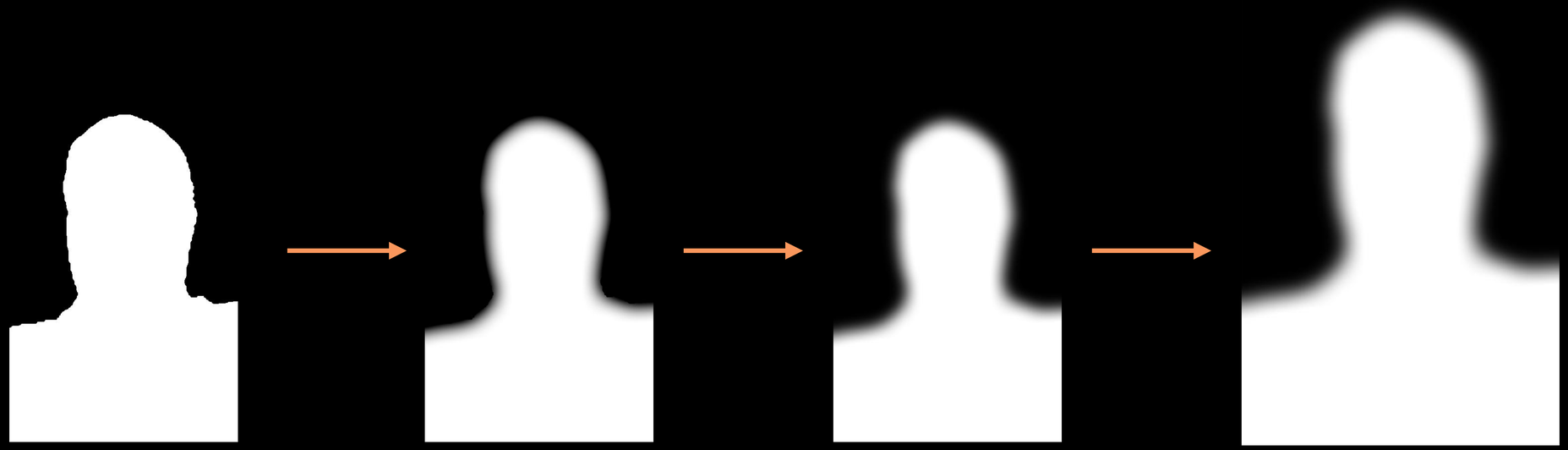
```
let matte = depthMask.applyingFilter("CIGaussianBlur", parameters: ["inputRadius": 20])
    .applyingFilter("CIGammaAdjust", parameters: ["inputPower": 2])
    .applyingFilter(“CIBicubicScaleTransform”, parameters:“inputScale”: RGBwidth/depthWidth])
```



Binary Mask

Alpha matte

```
let matte = depthMask.applyingFilter("CIGaussianBlur", parameters: ["inputRadius": 20])
    .applyingFilter("CIGammaAdjust", parameters: ["inputPower": 2])
    .applyingFilter(“CIBicubicScaleTransform”, parameters:“inputScale”: RGBwidth/depthWidth])
```



Clamp Before Filtering

Avoiding edge softening

```
let matte = depthMask.clampedToExtent()
    .applyingFilter("CIGaussianBlur", parameters: ["inputRadius": 20])
    .applyingFilter("CIGammaAdjust", parameters: ["inputPower": 2])
    .applyingFilter(“CIBicubicScaleTransform”, parameters: [“inputScale”: RGBwidth/depthWidth])
```

Crop After Filtering

Keeping the original extent

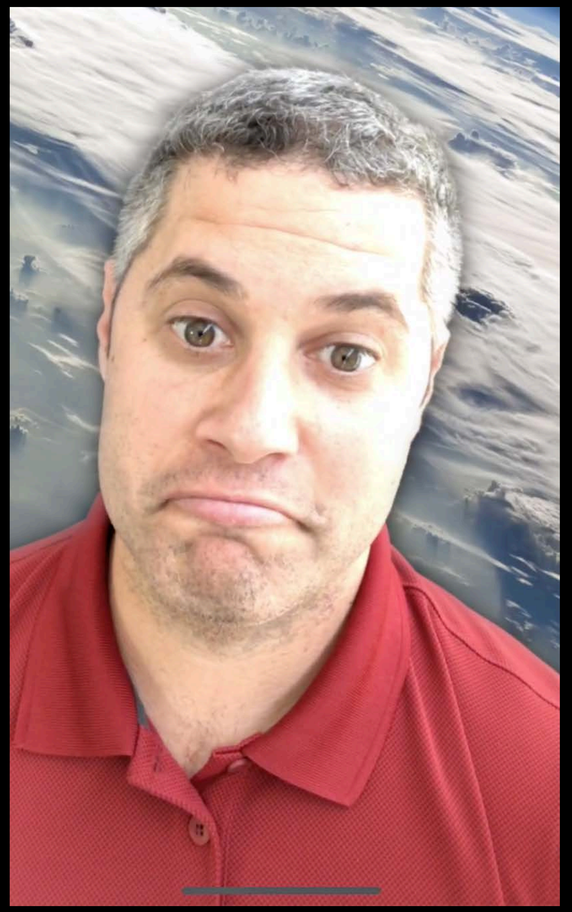
```
let matte = depthMask.clampedToExtent()
    .applyingFilter("CIGaussianBlur", parameters: ["inputRadius": 20])
    .applyingFilter("CIGammaAdjust", parameters: ["inputPower": 2])
    .cropped(to: depthMask.extent)
    .applyingFilter(“CIBicubicScaleTransform”, parameters: [“inputScale”: RGBwidth/depthWidth])
```

Blending Foreground with Background

```
let output = rgbImage.applyingFilter(“CIBlendWithMask",
    parameters: ["inputMaskImage": matte,
    “inputBackgroundImage”: background])
```

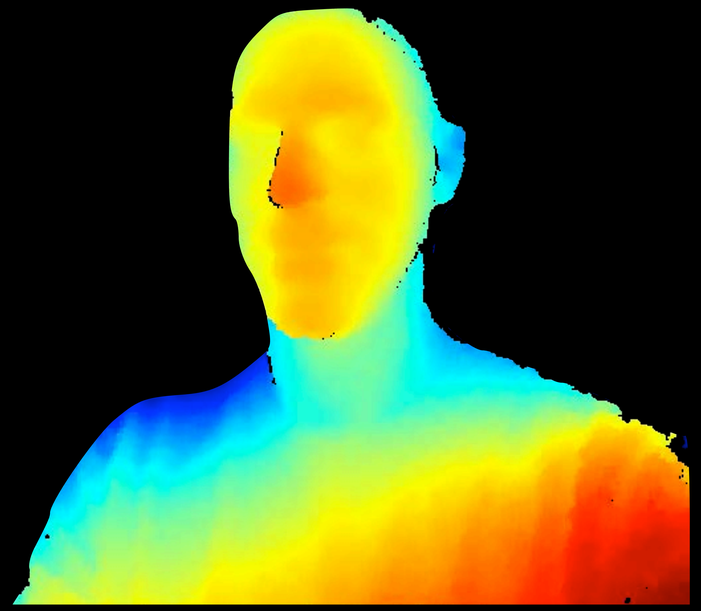
Blending Foreground with Background

```
let output = rgbImage.applyingFilter(“CIBlendWithMask",
    parameters: ["inputMaskImage": matte,
    “inputBackgroundImage“: background])
```



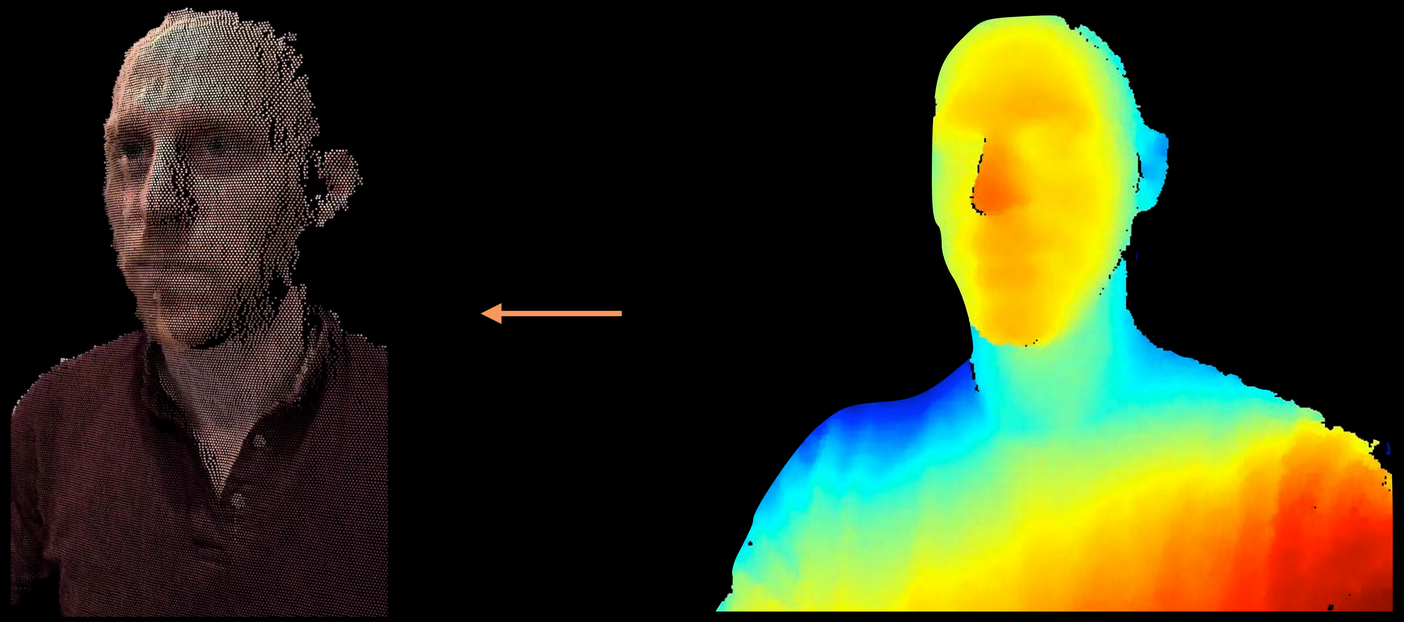
TrueDepth Stream

640x480 depth map registered to video at 30 fps



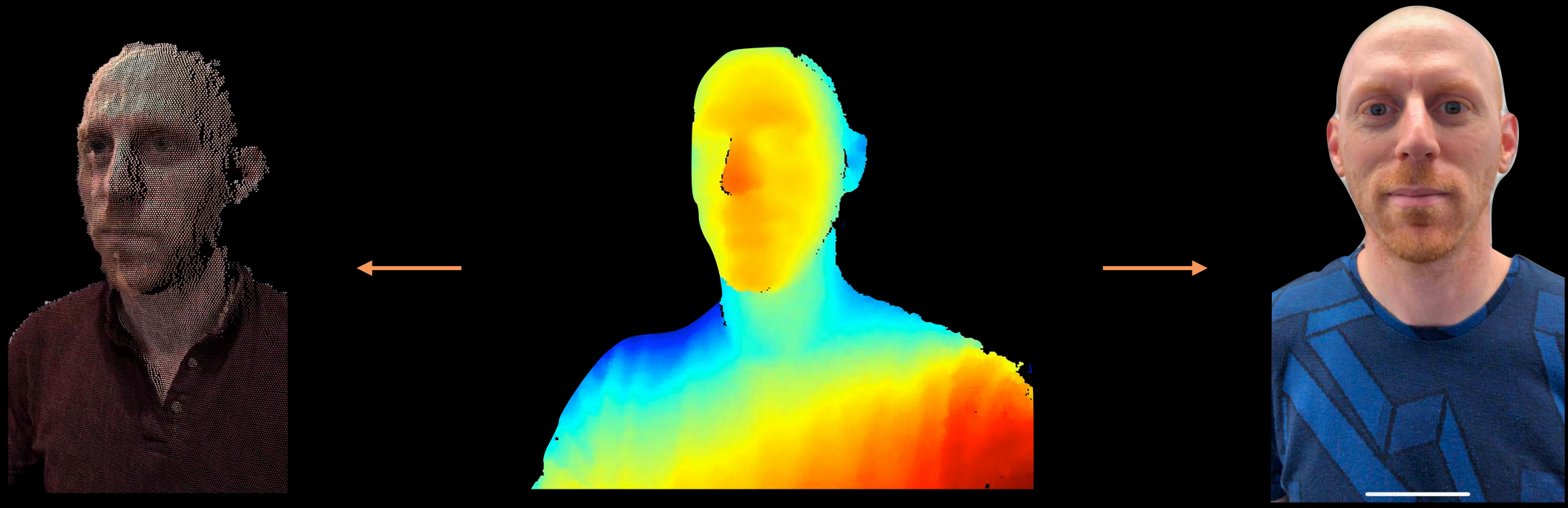
TrueDepth Stream

640x480 depth map registered to video at 30 fps



TrueDepth Stream

640x480 depth map registered to video at 30 fps



More Information

https://developer.apple.com/wwdc18/503

AVCapture Lab	Technology Lab 4	Thursday 3:00PM


WWDC18