Advances in Camera Capture and Photo Segmentation

Brad Ford, Camera Software
Jacob Schack Vestergaard, Camera Software
David Hayward, Core Image
Multi-Camera Capture

Semantic Segmentation for Photos
Multi-Camera Capture
Multi-Camera Capture Support by Platform

OS X Lion (2011)
Multi-Camera Capture Support by Platform

OS X Lion (2011)

iOS 12 (2018)
Multi-Camera Capture Support
Multi-Camera Capture Support
Multi-Camera Capture Support
Multi-Camera Capture
Session building
AV Foundation Capture Classes
AV Foundation Capture Classes

AVCaptureSession
AV Foundation Capture Classes

AVCaptureSession

AVCaptureDeviceInput

AVCaptureDevice (DualCamera)
AV Foundation Capture Classes

AVCaptureSession

AVCaptureDeviceInput

AVCaptureDevice
(DualCamera)

AVCaptureVideoDataOutput

AVCaptureDepthDataOutput
AV Foundation Capture Classes

AVCaptureDevice (DualCamera)

AVCaptureDeviceInput

AVCaptureSession

AVCaptureVideoDataOutput

AVCaptureDepthDataOutput

Video SampleBuffers

AVDepthData
Introducing AVCaptureMultiCamSession

Multiple AVCaptureDeviceInputs

Multiple AVCaptureOutputs of the same type

Multiple AVCaptureVideoPreviewLayers

Not a replacement for AVCaptureSession
AVCaptureMultiCamSession Example
AVCaptureMultiCamSession Example

AVCaptureDevice (Front Camera)

AVCaptureDeviceInput

AVCaptureDevice (Back Camera)

AVCaptureDeviceInput

AVCaptureMultiCamSession
AVCaptureMultiCamSession Example

AVCaptureDevice (Front Camera)

AVCaptureDeviceInput

AVCaptureMultiCamSession

AVCaptureVideoDataOutput (Front)

Video SampleBuffers

AVCaptureDevice (Back Camera)

AVCaptureDeviceInput

AVCaptureMultiCamSession

AVCaptureVideoDataOutput (Back)

Video SampleBuffers
AVCaptureMultiCamSession Example

AVCaptureDeviceInput (Front Camera) -> AVCaptureMultiCamSession

AVCaptureVideoDataOutput (Front) -> AVCaptureVideoPreviewLayer (Front)

AVCaptureVideoDataOutput (Back) -> AVCaptureVideoPreviewLayer (Back)
AVCaptureMultiCamSession Example

AVCaptureDeviceInput (Front Camera) -> AVCaptureMultiCamSession -> AVCaptureVideoDataOutput (Front)

AVCaptureDeviceInput (Back Camera) -> AVCaptureMultiCamSession -> AVCaptureVideoDataOutput (Back)

AVCaptureVideoPreviewLayer (Front) -> AVCaptureVideoDataOutput (Front)

AVCaptureVideoPreviewLayer (Back) -> AVCaptureVideoDataOutput (Back)

AVCaptureMetadataOutput (Front) -> AVCaptureVideoDataOutput (Front)

AVCaptureMetadataOutput (Back) -> AVCaptureVideoDataOutput (Back)
AVCaptureMultiCamSession Example

AVCaptureDevice (Front Camera)
  └── AVCaptureDeviceInput
      ├── AVCaptureVideoDataOutput (Front)
      │   ├── AVCaptureVideoPreviewLayer (Front)
      │   └── AVCaptureMetadataOutput (Front)
      └── AVCaptureMovieFileOutput (Front)

AVCaptureDevice (Back Camera)
  └── AVCaptureDeviceInput
      ├── AVCaptureVideoDataOutput (Back)
      │   └── AVCaptureVideoPreviewLayer (Back)
      └── AVCaptureMetadataOutput (Back)

AVCaptureMultiCamSession
  ├── AVCaptureVideoDataOutput (Front)
  │   ├── AVCaptureVideoPreviewLayer (Front)
  │   └── AVCaptureMetadataOutput (Front)
  └── AVCaptureVideoDataOutput (Back)
      └── AVCaptureVideoPreviewLayer (Back)

AVCaptureMovieFileOutput (Front)

AVCapturePhotoOutput (Front)

AVCapturePhotoOutput (Back)

AVCaptureMovieFileOutput (Back)
Anatomy of a Connection

AVCaptureDevice
(Dual Camera)

AVCaptureDeviceInput
Anatomy of a Connection

AVCaptureDevice
(Dual Camera)

AVCaptureDeviceInput

<table>
<thead>
<tr>
<th>Video Port</th>
<th>Depth Port</th>
<th>Metadata Objects Port</th>
<th>Face Metadata Items Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
let ports: [AVCaptureInput.Port] = dualCameraDeviceInput.ports
Anatomy of a Connection

AVCaptureDevice (Dual Camera)

AVCaptureDeviceInput

- Video Port
- Depth Port
- Metadata Objects Port
- Face Metadata Items Port
Anatomy of a Connection

```python
session.addInput(dualCameraInput)
```
Anatomy of a Connection

AVCaptureDevice (Dual Camera)

AVCaptureDeviceInput

Video Port
Depth Port
Metadata Objects Port
Face Metadata Items Port

AVCaptureSession

session.addOutput(videoDataOutput)
Anatomy of a Connection

AVCaptureDevice
(Dual Camera)

AVCaptureDeviceInput

Video Port
Depth Port
Metadata Objects Port
Face Metadata Items Port

AVCaptureSession

AVCaptureVideoDataOutput
MultiCam Session Building Best Practices

When adding inputs and outputs
MultiCam Session Building Best Practices

When adding inputs and outputs

• Use `addInputWithNoConnections` or `addOutputWithNoConnections`
MultiCam Session Building Best Practices

When adding inputs and outputs
• Use `addInputWithNoConnections` or `addOutputWithNoConnections`

When adding video preview layers
MultiCam Session Building Best Practices

When adding inputs and outputs

• Use `addInputWithNoConnections` or `addOutputWithNoConnections`

When adding video preview layers

• Use `AVCaptureVideoPreviewLayer.setSessionWithNoConnections()`
Then create and add explicit connections

```swift
let backCameraVideoConnection = AVCaptureConnection(inputPorts: [backCameraVideoPort],
                                                     output: backVideoDataOutput)

session.addConnection(backCameraVideoConnection)
```
AVMultiCamPiP Graph Topology

AVCaptureDevice (Front Camera)
  - AVCaptureDeviceInput

AVCaptureDevice (Back Camera)
  - AVCaptureDeviceInput

AVCaptureMultiCamSession
  - AVCaptureVideoDataOutput (Front)
  - AVCaptureVideoDataOutput (Back)

AVCaptureVideoPreviewLayer (Front)
  - (Metal Shader Compositor)

AVCaptureVideoPreviewLayer (Back)

AVAssetWriter
Multi-Camera Capture

Limitations
No Camera Cloning at the Input

AVCaptureDevice (Back Camera)

AVCaptureDeviceInput 1

AVCaptureDeviceInput 2

AVCaptureMultiCamSession
No Split Personality Camera Output

AVCaptureDevice (Back Camera)

AVCaptureDeviceInput

AVCaptureMultiCamSession

AVCaptureVideoDataOutput 1

AVCaptureVideoDataOutput 2
No Split Personality Camera Output

AVCaptureDevice (Back Camera)

AVCaptureDeviceInput

AVCaptureMultiCamSession

AVCaptureVideoDataOutput 1

AVCaptureVideoDataOutput 2
No Multi-Camera Output Stuffing

AVCaptureDevice (Back Camera)
AVCaptureDeviceInput

AVCaptureDevice (Back Camera)
AVCaptureDeviceInput

AVCaptureMultiCamSession

AVCaptureVideoDataOutput
No Multi-Camera Output Stuffing

AVCaptureDeviceInput (Back Camera)

AVCaptureDeviceInput (Back Camera)

AVCaptureMultiCamSession

AVCaptureVideoDataOutput
Presets

session.sessionPreset = .high
Presets? We Don’t Need No Stinkin’ Presets
Presets? We Don’t Need No Stinkin’ Presets

AVCaptureMultiCamSession

- Front Camera
  - AVCaptureDeviceInput
  - AVCaptureVideoPreviewLayer (Front)
  - AVCapturePhotoOutput (Front)
  - AVCaptureMetadataOutput (Front Faces)
  - AVCaptureMovieFileOutput (Front)

- Microphone
  - AVCaptureDeviceInput
  - AVCaptureVideoPreviewLayer

- Back Camera
  - AVCaptureDeviceInput
  - AVCaptureVideoPreviewLayer (Back)
  - AVCapturePhotoOutput (Back)
  - AVCaptureMetadataOutput (Back Faces)
  - AVCaptureMovieFileOutput (Back)
Presets? We Don’t Need No Stinkin’ Presets
Preset? We Don’t Need No Stinkin’ Presets

Front Camera

- AVCaptureDeviceInput
- AVCaptureMovieFileOutput (Front)
- AVCaptureMetadataOutput (Front Faces)
- AVCapturePhotoOutput (Front)

Microphone

- AVCaptureDeviceInput

Back Camera

- AVCaptureDeviceInput
- AVCaptureMovieFileOutput (Back)
- AVCaptureMetadataOutput (Back Faces)
- AVCapturePhotoOutput (Back)

AVCaptureMultiCamSession

640x480 @ 30 fps
Presets? We Don’t Need No Stinkin’ Presets
Presets? We Don’t Need No Stinkin’ Presets

AVCaptureDeviceInput

AVCaptureVideoPreviewLayer

AVCapturePhotoOutput

AVCaptureMetadataOutput

AVCaptureMovieFileOutput

1920x1080 @ 60 fps

AVCaptureDeviceInput

AVCaptureVideoPreviewLayer

AVCapturePhotoOutput

AVCaptureMetadataOutput

AVCaptureMovieFileOutput

Front Camera

Microphone

Back Camera

AVCaptureMultiCamSession
Multi-Camera Capture

Cost functions
“There’s no such thing as a free lunch.”

- Somebody who wants you to pay for lunch
Hardware Costs

Multiple cameras = multiple sensors
Hardware Costs

Multiple cameras = multiple sensors

One ISP, limited to n pixels per clock
Hardware Costs

Multiple cameras = multiple sensors

One ISP, limited to n pixels per clock

Contributors to hardware cost
Hardware Costs

Multiple cameras = multiple sensors

One ISP, limited to n pixels per clock

Contributors to hardware cost
• Video resolution
Hardware Costs

Multiple cameras = multiple sensors

One ISP, limited to n pixels per clock

Contributors to hardware cost
  • Video resolution
  • Max frame rate
Hardware Costs

Multiple cameras = multiple sensors

One ISP, limited to n pixels per clock

Contributors to hardware cost

- Video resolution
- Max frame rate
- Sensor “binning”
Sensor Binning
Sensor Binning
Sensor Binning
Hardware Cost Reporting
Hardware Cost Reporting

multicamSession.hardwareCost
Hardware Cost Reporting

multicamSession.hardwareCost

≥ 0.0 && ≤ 1.0 : Runnable
Hardware Cost Reporting

```
multicamSession.hardwareCost

≥ 0.0 && ≤ 1.0 : Runnable

≥ 1.0 : No Bueno```
Hardware Cost Reporting

\[
\text{multicamSession.hardwareCost} \geq 0.0 \&\& \leq 1.0 \text{ : Runnable}
\]

\[
\text{\geq 1.0} \text{ : No Bueno}
\]

\[
\text{AVCaptureSessionRuntimeError} = \text{hardwareCostOverage}
\]
How to Reduce Your Hardware Cost

Refer to AVMultiCamPiP Sample Code
How to Reduce Your Hardware Cost

Lower the camera resolution

Choose a binned format

Refer to AVMultiCamPiP Sample Code
How to Reduce Your Hardware Cost

Lower the camera resolution
Choose a binned format
Lower the camera max frame rate

Refer to AVMultiCamPiP Sample Code
How to Reduce Your Hardware Cost

Lower the camera resolution

Choose a binned format

Lower the camera max frame rate

Refer to AVMultiCamPiP Sample Code
How to Reduce Your Hardware Cost

Lower the camera resolution

Choose a binned format

Lower the camera max frame rate

Set a max frame rate override at the device input

Refer to AVMultiCamPiP Sample Code
How to Reduce Your Hardware Cost

Lower the camera resolution

Choose a binned format

Lower the camera max frame rate

Set a max frame rate override at the device input

```
CMTime thirtyFPS = CMTimeMake( 1, 30 )
deviceInput.videoMinFrameDurationOverride = thirtyFPS
```

Refer to AVMultiCamPiP Sample Code
System Pressure States
System Pressure States

Camera System Pressure States introduced in iOS 11
System Pressure States

Camera System Pressure States introduced in iOS 11

- System Temperature
Camera System Pressure States introduced in iOS 11

- System Temperature
- Peak Power Demands
System Pressure States

Camera System Pressure States introduced in iOS 11

- System Temperature
- Peak Power Demands
- Infrared Projector Temperature
System Pressure States

Nominal
Fair
Serious
Critical
Shutdown
System Pressure Cost

Camera system cost excluding all other factors

Contributors
• Same as hardware cost contributors
• VIS, OIS
• Smart HDR
• Infrared sensor and projector power for TrueDepth
• Microphone power
System Pressure Cost Reporting

MultiCamSession tallies a system pressure cost
Independent of current system pressure state
Accounts only for factors the session knows about
System Pressure Cost Reporting
System Pressure Cost Reporting

multicamSession.systemPressureCost
System Pressure Cost Reporting

multicamSession.systemPressureCost

< 1.0 : Runnable indefinitely
multicamSession.systemPressureCost

- < 1.0: Runnable indefinitely
- ≥ 1.0 && ≤ 2.0: Runnable for 15 minutes
System Pressure Cost Reporting

- `< 1.0` : Runnable indefinitely
- `≥ 1.0 && ≤ 2.0` : Runnable for 15 minutes
- `≥ 2.0 && ≤ 3.0` : Runnable for 10 minutes
System Pressure Cost Reporting

```
multicamSession.systemPressureCost

< 1.0 : Runnable indefinitely
≥ 1.0 && ≤ 2.0 : Runnable for 15 minutes
≥ 2.0 && ≤ 3.0 : Runnable for 10 minutes
> 3.0 : ¿Cómo se dice en fuego?
```
System Pressure Cost Reporting

- `< 1.0 : Runnable indefinitely`
- `≥ 1.0 && ≤ 2.0 : Runnable for 15 minutes`
- `≥ 2.0 && ≤ 3.0 : Runnable for 10 minutes`
- `> 3.0 : ¿Cómo se dice en fuego?`

MulticamSession will interrupt your session when system pressure = Shutdown
How to Reduce System Pressure While Running

Lower one or more capture device frame rates

Throttle any GPU or CPU processing in your app code

Disable a camera (session keeps running!)
How to Reduce System Pressure While Running

Lower one or more capture device frame rates

Throttle any GPU or CPU processing in your app code

Disable a camera (session keeps running!)

```javascript
frontCameraInputVideoPort.enabled = false
```
Unreported Costs

Unreported costs are unreported.

Artificial constraints on devices and formats.
<table>
<thead>
<tr>
<th></th>
<th>Back Wide</th>
<th>Back Telephoto</th>
<th>Back Dual Camera</th>
<th>Front</th>
<th>Front TrueDepth</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>3</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>5</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>6</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>
guard AVCaptureMultiCamSession.isMultiCamSupported else {
    print("MultiCam not supported on this device")
    setupResult = .multiCamNotSupported
    return
}

// Find the supported multicam device combinations

let deviceTypes = [AVCaptureDevice.DeviceType.builtInDualCamera,
                   AVCaptureDevice.DeviceType.builtInWideAngleCamera,
                   AVCaptureDevice.DeviceType.builtInTelephotoCamera]

let session = AVCaptureDevice.DiscoverySession(deviceTypes: deviceTypes,
                                                mediaType: .video,
                                                position: .unspecified)

let multicamSupportedDeviceSets = session.supportedMultiCamDeviceSets
guard AVCaptureMultiCamSession.isMultiCamSupported else {
    print("MultiCam not supported on this device")
    setupResult = .multiCamNotSupported
    return
}

// Find the supported multicam device combinations

let deviceTypes = [AVCaptureDevice.DeviceType.builtInDualCamera,
    AVCaptureDevice.DeviceType.builtInWideAngleCamera,
    AVCaptureDevice.DeviceType.builtInTelephotoCamera]

let session = AVCaptureDevice.DiscoverySession(deviceTypes: deviceTypes,
    mediaType: .video,
    position: .unspecified)

let multicamSupportedDeviceSets = session.supportedMultiCamDeviceSets
guard AVCaptureMultiCamSession.isMultiCamSupported else {
    print("MultiCam not supported on this device")
    setupResult = .multiCamNotSupported
    return
}

// Find the supported multicam device combinations

let deviceTypes = [AVCaptureDevice.DeviceType.builtInDualCamera,
                   AVCaptureDevice.DeviceType.builtInWideAngleCamera,
                   AVCaptureDevice.DeviceType.builtInTelephotoCamera]

let session = AVCaptureDevice.DiscoverySession(deviceTypes: deviceTypes,
                                              mediaType: .video,
                                              position: .unspecified)

let multicamSupportedDeviceSets = session.supportedMultiCamDeviceSets
## Supported MultiCam Formats (iPhone XS)

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Max FPS</th>
<th>Binned?</th>
<th>Hi-Res Stills</th>
</tr>
</thead>
<tbody>
<tr>
<td>640x480</td>
<td>60</td>
<td>Yes</td>
<td>2016x1512</td>
</tr>
<tr>
<td>1280x720</td>
<td>60</td>
<td>Yes</td>
<td>2112x1188</td>
</tr>
<tr>
<td>1440x1080</td>
<td>60</td>
<td>Yes</td>
<td>2016x1512</td>
</tr>
<tr>
<td>1920x1080</td>
<td>30</td>
<td>No</td>
<td>4224x2376</td>
</tr>
<tr>
<td>1920x1080</td>
<td>60</td>
<td>Yes</td>
<td>2112x1188</td>
</tr>
<tr>
<td>1920x1440</td>
<td>30</td>
<td>No</td>
<td>4032x3024</td>
</tr>
<tr>
<td>1920x1440</td>
<td>60</td>
<td>Yes</td>
<td>2016x1512</td>
</tr>
</tbody>
</table>
## Supported MultiCam Formats (iPhone XS)

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Max FPS</th>
<th>Binned?</th>
<th>Hi-Res Stills</th>
</tr>
</thead>
<tbody>
<tr>
<td>640x480</td>
<td>60</td>
<td>Yes</td>
<td>2016x1512</td>
</tr>
<tr>
<td>1280x720</td>
<td>60</td>
<td>Yes</td>
<td>2112x1188</td>
</tr>
<tr>
<td>1440x1080</td>
<td>60</td>
<td>Yes</td>
<td>2016x1512</td>
</tr>
<tr>
<td>1920x1080</td>
<td>30</td>
<td>No</td>
<td>4224x2376</td>
</tr>
<tr>
<td>1920x1080</td>
<td>60</td>
<td>Yes</td>
<td>2112x1188</td>
</tr>
<tr>
<td>1920x1440</td>
<td>30</td>
<td>No</td>
<td>4032x3024</td>
</tr>
<tr>
<td>1920x1440</td>
<td>60</td>
<td>Yes</td>
<td>2016x1512</td>
</tr>
</tbody>
</table>
## Supported MultiCam Formats (iPhone XS)

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Max FPS</th>
<th>Binned?</th>
<th>Hi-Res Stills</th>
</tr>
</thead>
<tbody>
<tr>
<td>640x480</td>
<td>60</td>
<td>Yes</td>
<td>2016x1512</td>
</tr>
<tr>
<td>1280x720</td>
<td>60</td>
<td>Yes</td>
<td>2112x1188</td>
</tr>
<tr>
<td>1440x1080</td>
<td>60</td>
<td>Yes</td>
<td>2016x1512</td>
</tr>
<tr>
<td>1920x1080</td>
<td>30</td>
<td>No</td>
<td>4224x2376</td>
</tr>
<tr>
<td>1920x1080</td>
<td>60</td>
<td>Yes</td>
<td>2112x1188</td>
</tr>
<tr>
<td>1920x1440</td>
<td>30</td>
<td>No</td>
<td>4032x3024</td>
</tr>
<tr>
<td>1920x1440</td>
<td>60</td>
<td>Yes</td>
<td>2016x1512</td>
</tr>
</tbody>
</table>
## Supported MultiCam Formats (iPhone XS)

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Max FPS</th>
<th>Binned?</th>
<th>Hi-Res Stills</th>
</tr>
</thead>
<tbody>
<tr>
<td>640x480</td>
<td>60</td>
<td>Yes</td>
<td>2016x1512</td>
</tr>
<tr>
<td>1280x720</td>
<td>60</td>
<td>Yes</td>
<td>2112x1188</td>
</tr>
<tr>
<td>1440x1080</td>
<td>60</td>
<td>Yes</td>
<td>2016x1512</td>
</tr>
<tr>
<td>1920x1080</td>
<td>30</td>
<td>No</td>
<td>4224x2376</td>
</tr>
<tr>
<td>1920x1080</td>
<td>60</td>
<td>Yes</td>
<td>2112x1188</td>
</tr>
<tr>
<td>1920x1440</td>
<td>30</td>
<td>No</td>
<td>4032x3024</td>
</tr>
<tr>
<td>1920x1440</td>
<td>60</td>
<td>Yes</td>
<td>2016x1512</td>
</tr>
</tbody>
</table>
// Find and activate the next smaller format that supports multicam

let formats = videoDeviceInput.device.formats
for format in formats.reversed() {
    if format.isMultiCamSupported {
        dims = CMVideoFormatDescriptionGetDimensions(format.formatDescription)
        if dims.width < activeWidth || dims.height < activeHeight {
            do {
                try videoDeviceInput.device.lockForConfiguration()
                videoDeviceInput.device.activeFormat = format
                videoDeviceInput.device.unlockForConfiguration()
                return true
            } catch {
                return false
            }
        }
    }
}
## Supported Session / App Configurations

<table>
<thead>
<tr>
<th>Configuration</th>
<th>macOS</th>
<th>iOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>One app, one capture session, multiple cameras</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>One app, multiple sessions, one or more cameras</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Multiple apps, multiple sessions, one or more cameras</td>
<td>✔</td>
<td>✗</td>
</tr>
</tbody>
</table>
Be Responsible
Multi-Camera Capture
Synchronized Streaming
Virtual Cameras

- Ambient light sensor
- Infrared camera
- Flood illuminator
- Dot projector
- 7MP camera
- Proximity sensor
- Speaker
- Microphone
- Front camera
- Dot projector
Virtual Cameras

DualCam presents one video stream at a time*
Switches between cameras at different zoom levels
Generates disparity (depth) from wide and tele

*AVCapturePhotoOutput dualCameraDualPhotoDelivery is the exception
// Virtual devices are made up of constituent devices

if aCameraDevice.isVirtualDevice == true
{
  let subCameras = aCameraDevice.constituentDevices

  for let subCamera in subCameras {
    print("Sub cameras: \(subCamera.localizedName)")
  }
}
Synchronized Streaming
Synchronised Streaming

When running a virtual device, its constituent devices share the same
Synchronized Streaming

When running a virtual device, its constituent devices share the same
• Active resolution
Synchronized Streaming

When running a virtual device, its constituent devices share the same
• Active resolution
• Frame rate
Synchronized Streaming

When running a virtual device, its constituent devices share the same
• Active resolution
• Frame rate

They are synchronized
Synchronized Streaming

When running a virtual device, its constituent devices share the same

• Active resolution

• Frame rate

They are synchronized

• Sensor read out matches frame centers
Synchronized Streaming

When running a virtual device, its constituent devices share the same
• Active resolution
• Frame rate

They are synchronized
• Sensor read out matches frame centers
• Exposure / White Balance / Focus
Demo
AVDualCam
AVDualCam Graph Topology

AVCaptureDevice (DualCamera)

AVCaptureDeviceInput

AVCaptureMultiCamSession

AVCaptureVideoDataOutput (Wide)

AVCaptureDepthDataOutput (Tele)

AVCaptureDataOutputSynchronizer

(Metal Shader Filter / Compositor)

Preview

AVAssetWriter
Any Port in a Storm?

let ports: [AVCaptureInput.Port] = dualCameraDeviceInput.ports
Virtual devices have secret ports.
// Constituent Device Port Discovery

    guard let widePort = dualCameraInput.ports(for: .video,
        sourceDeviceType: .builtInWideAngleCamera,
        sourceDevicePosition: dualCamera.position).first,
    let telePort = dualCameraInput.ports(for: .video,
        sourceDeviceType: .builtInTelephotoCamera,
        sourceDevicePosition: dualCamera.position).first

else {
    print("Could not obtain wide and telephoto camera input ports")
    return false
}
// Constituent Device Port Discovery

guard let widePort = dualCameraInput.ports(for: .video,
    sourceDeviceType: .builtInWideAngleCamera,
    sourceDevicePosition: dualCamera.position).first,
let telePort = dualCameraInput.ports(for: .video,
    sourceDeviceType: .builtInTelephotoCamera,
    sourceDevicePosition: dualCamera.position).first

else {
    print("Could not obtain wide and telephoto camera input ports")
    return false
}
// Constituent Device Port Connections

let wideAngleCameraConnection = AVCaptureConnection(inputPorts: [widePort],
    output: wideVideoDataOutput)

guard session.canAddConnection(wideAngleCameraConnection) else {
    print("Could not connect wide-angle video input to output")
    return false
}

session.addConnection(wideAngleCameraConnection)
Dual Camera Homography Aids

Camera Intrinsics

Camera Extrinsics
Camera Intrinsics

\[
K = \begin{bmatrix}
  f_x & s & x_0 \\
  0 & f_y & y_0 \\
  0 & 0 & 1 \\
\end{bmatrix}
\]
// Get extrinsics
if let wide = AVCaptureDevice.default(.builtInWideAngleCamera, for: nil, position: .back),
let tele = AVCaptureDevice.default(.builtInTelephotoCamera, for: nil, position: .back) {
    self.extrinsics = AVCaptureDevice.extrinsicMatrix(from: tele, to: wide)
}

\[
\begin{bmatrix}
R_1,1 & R_1,2 & R_1,3 & t_1 \\
R_2,1 & R_2,2 & R_2,3 & t_2 \\
R_3,1 & R_3,2 & R_3,3 & t_3 \\
\end{bmatrix}
\]

Rotation  Translation
Removing Despair from Disparity

\[ \frac{b}{z} = \frac{d}{fl} \]
Multi-Camera Capture
Multi-Microphone Capture
Review: AVCaptureSession Mic Selection

Default behaviors
- Front camera in use = selects front microphone
- Back camera in use = selects back microphone
- Audio only session = selects omnidirectional mic (usually at the bottom)

Power feature — app can use AVAudioSession to configure mics
There’s no such thing as a “front mic”.
iPhone Microphone Arrays

Landscape view:
- Mic 2
- Mic 4
- Mic 3
- Mic 1

Side view:
- Mic 3
- Mic 2
- Mics 1 and 4
Microphone Beam Forming

Diagram showing two microphones connected to an iPhone.
Microphone Beam Forming

- iPhone
- 2x
- 3x
AVCaptureSession Mic Input Port Behavior

AVCaptureDeviceInput (Microphone)

Primary Port
(Front / Back / Omni)
MultiCam Behavior: The Return of the Secret Ports

AVCaptureDeviceInput (Microphone)

Primary Port
(Omni)

Back Beam-Formed

Back Beam-Formed
Positional Audio Connections

```swift
let frontPort = micInput.ports(for: .audio,
    sourceDeviceType: micDevice.deviceType,
    sourceDevicePosition: .front).first

let backPort = micInput.ports(for: .audio,
    sourceDeviceType: micDevice.deviceType,
    sourceDevicePosition: .back).first
```

Only supported in AVCaptureMultiCamSession
let frontPort = micInput.ports(for: .audio,
    sourceDeviceType: micDevice.deviceType,
    sourceDevicePosition: .front).first

let backPort = micInput.ports(for: .audio,
    sourceDeviceType: micDevice.deviceType,
    sourceDevicePosition: .back).first
Positional Audio Connections

```swift
let frontPort = micInput.ports(for: .audio,
    sourceDeviceType: micDevice.deviceType,
    sourceDevicePosition: .front).first

let backPort = micInput.ports(for: .audio,
    sourceDeviceType: micDevice.deviceType,
    sourceDevicePosition: .back).first
```

Only supported in AVCaptureMultiCamSession
Multi-Mic Beam Forming in AVMultiCamPiP

Microphone

AVCaptureDeviceInput

AVCaptureMultiCamSession

AVCaptureVideoDataOutput (Back Beam Form)

AVCaptureDepthDataOutput (Front Beam Form)

AVAssetWriter
Non Built-In Mic Behaviors

Beam-forming only works with built-in mics

External audio signal is duplicated to all audio input ports (omni, front, and back)
Multi-Camera Capture Summary
Multi-Camera Capture Summary

Use AVCaptureMultiCamSession
Multi-Camera Capture Summary

Use AVCaptureMultiCamSession

Know its limitations
Multi-Camera Capture Summary

Use AVCaptureMultiCamSession

Know its limitations

Thoughtfully handle hardware and system pressure costs
Multi-Camera Capture Summary

Use AVCaptureMultiCamSession

Know its limitations

Thoughtfully handle hardware and system pressure costs

Use constituent device ports for synchronized camera streaming
Multi-Camera Capture Summary

Use AVCaptureMultiCamSession

Know its limitations

Thoughtfully handle hardware and system pressure costs

Use constituent device ports for synchronized camera streaming

Use front, back, or omni mics simultaneously
And Now For Something Completely Different

Changes to AVCapturePhotoOutput

• Deprecation of auto still image stabilization
• Addition of photo quality versus speed hinting

Video coming soon to developer.apple.com
Semantic Segmentation Mattes

Jacob Schack Vestergaard, Camera Software
David Hayward, Core Image
iOS12: PortraitEffectsMatte
iOS12: PortraitEffectsMatte
Semantic Segmentation Mattes
Semantic Segmentation Mattes
Semantic Segmentation Mattes
Semantic Segmentation Mattes

NEW
Semantic Segmentation Mattes
Semantic Segmentation Mattes

$\alpha = 0.7$
Mattes Are Half-Size
Under the Hood

Apple Neural Engine

Image

PortaitEffectsMatte  Skin  Hair  Teeth

Segmentation and matting

Image  Depth  PEM  Skin  Hair  Teeth

0.5W x 0.5H  0.5W x 0.5H  0.5W x 0.5H  0.5W x 0.5H

HEIF/JPEG
Where Do the Mattes Come From?

Embedded in Portrait Mode captures
Write your own capture app and opt-in
Capturing Segmentation Mattes
Initiate capture request

Settings resolved

Photo finished processing

Setting up AVCapturePhotoOutput

Working with HEIF and HEVC

WWDC 2017
Setting up AVCapturePhotoOutput

// begin configuration, set preset, add device input, ...

if session.canAddOutput(output) {
    session.addOutput(output)
    // what you usually do...

    output.enabledSemanticSegmentationMatteTypes = output.availableSemanticSegmentationMatteTypes
}
Initiating a Capture Request

```
let settings = AVCapturePhotoSettings()

settings.enabledSemanticSegmentation MatteTypes = output.enabledSemanticSegmentation MatteTypes
// or
// settings.enabledSemanticSegmentation MatteTypes = [.hair, .skin]

photoOutput.capturePhoto(with: settings, delegate: self)
```
func photoOutput(_ output: AVCapturePhotoOutput, 
willBeginCaptureFor settings: AVCaptureResolvedPhotoSettings) {

    let matteDimensions = settings.dimensionsForSemanticSegmentationMatte(ofType: .hair)
}
Resolved Capture Settings

```swift
func photoOutput(_ output: AVCapturePhotoOutput,
willBeginCaptureFor settings: AVCaptureResolvedPhotoSettings) {
    let matteDimensions = settings.dimensionsForSemanticSegmentationMatte(ofType: .hair)
}
```
func photoOutput(_ output: AVCapturePhotoOutput, didFinishProcessingPhoto photo: AVCapturePhoto, error: Error?) {
    if var matte = photo.semanticSegmentationMatte(forType: .teeth) {
        let teethBuffer = matte.mattingImage
    }
}
func photoOutput(_ output: AVCapturePhotoOutput, didFinishProcessingPhoto photo: AVCapturePhoto, error: Error?) {
    if var matte = photo.semanticSegmentationMatte(forType: .teeth) {
        let teethBuffer = matte.mattingImage
    }
}
Leveraging Core Image
Coulrophobia
[kool-ruh-foh-bee-uh]
An extreme or irrational fear of clowns
Using Segmentation Mattes with Core Image
Using Segmentation Mattes with Core Image

Creating matte images
Using Segmentation Mattes with Core Image

Creating matte images

Filtering matte images
Using Segmentation Mattes with Core Image

Creating matte images
Filtering matte images
Saving matte images
Creating Segmentation Mattes with Core Image

Creating a matte CIImage from AVSemanticSegmentationMatte

AVCapturePhoto → AVSemanticSegmentationMatte → CIImage
Creating a matte CIImage from AVSemanticSegmentationMatte

```swift
let matte = photo.semanticSegmentationMatte(forType: .hair)  // or .skin or .teeth
```
Creating a matte CIImage from AVSemanticSegmentationMatte

```swift
let matte = photo.semanticSegmentationMatte(forType: .hair)  // or .skin or .teeth
let img = CIImage(semanticSegmentationMatte: matte)
```
Creating Segmentation Mattes with Core Image

Loading a matte CIImage from HEIF
Creating Segmentation Mattes with Core Image

Loading a matte CIImage from HEIF

```
let main = CIImage(contentsOf: url)
```
Creating Segmentation Mattes with Core Image

Loading a matte CIImage from HEIF

```swift
let main = CIImage(contentsOf: url)
let hair = CIImage(contentsOf: url,
    options: [.auxiliarySemanticSegmentationHairMatte: true])
```
Creating Segmentation Mattes with Core Image

Loading a matte CIImage from HEIF

```swift
let main = CIImage(contentsOf: url)
let hair = CIImage(contentsOf: url,
                      options: [.auxiliarySemanticSegmentationHairMatte : true])
// or .auxiliarySemanticSegmentationSkinMatte
// or .auxiliarySemanticSegmentationTeethMatte
```
Filtering Segmentation Mattes with Core Image
Filtering Segmentation Mattes with Core Image

Base
Filtering Segmentation Mattes with Core Image
Filtering Segmentation Mattes with Core Image

Base

Adjusted

Matte
Filtering Segmentation Mattes with Core Image
// Filtering Segmentation Mattes with Core Image
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

let base = CIImage(contentsOf: url)
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

let base = CIImage(contentsOf: url)
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

let base = CIImage(contentsOf: url)

let maxcomp = CIFilter.maximumComponent
    maxcomp.inputImage = base
import CoreImage.CIFilterBuiltins

let base = CIImage( contentsOf : url )

let maxcomp = CIFilter.maximumComponent
    maxcomp.inputImage = base
var makeup = maxcomp.outputImage
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

let base = CIImage(contentsOf: url)

let maxcomp = CIFilter.maximumComponent
    maxcomp.inputImage = base
var makeup = maxcomp.outputImage
import CoreImage.CIFilterBuiltins

let base = CIImage(contentsOf: url)

let maxcomp = CIFilter.maximumComponent
    maxcomp.inputImage = base
var makeup = maxcomp.outputImage

let gamma = CIFilter.gammaAdjust
    blend.inputImage = makeup
    blend.power = 0.5
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

let base = CIImage(contentsOf: url)

let maxcomp = CIFilter.maximumComponent
    maxcomp.inputImage = base
var makeup = maxcomp.outputImage

let gamma = CIFilter.gammaAdjust
    blend.inputImage = makeup
    blend.power = 0.5
makeup = gamma.outputImage
import CoreImage.CIFilterBuiltins

let base = CIImage(contentsOf: url)

let maxcomp = CIFilter.maximumComponent
    maxcomp.inputImage = base
var makeup = maxcomp.outputImage

let gamma = CIFilter.gammaAdjust
    blend.inputImage = makeup
    blend.power = 0.5
makeup = gamma.outputImage
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

var matte = CIImage(contentsOf: url,
                    options: [.auxiliarySemanticSegmentationSkinMatte: true])
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

var matte = CIImage(contentsOf: url,
                      options: [.auxiliarySemanticSegmentationSkinMatte: true])
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

var matte = CIImage(contentsOf: url,
    options: [.auxiliarySemanticSegmentationSkinMatte: true])

let scale = CGAffineTransformMakeScale(
    base.extent.size.width / matte.extent.size.width,
    base.extent.size.height / matte.extent.size.height)
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

var matte = CIImage(contentsOf: url, options: [.auxiliarySemanticSegmentationSkinMatte: true])

let scale = CGAffineTransformMakeScale(base.extent.size.width / matte.extent.size.width, base.extent.size.height / matte.extent.size.height)

matte = matte.transformed(by: scale)
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

var matte = CIImage( contentsOf: url, 
                      options: [.auxiliarySemanticSegmentationSkinMatte: true] )

let scale = CGAffineTransformMakeScale(
                      base.extent.size.width / matte.extent.size.width,
                      base.extent.size.height / matte.extent.size.height)

matte = matte.transformed( by: scale )
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

let blend = CIFilter.blendWithMask
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

let blend = CIFilter.blendWithMask
    blend.backgroundImage = base
import CoreImage.CIFilterBuiltins

let blend = CIFilter.blendWithMask
blend.backgroundImage = base
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

let blend = CIFilter.blendWithMask
  blend.backgroundImage = base
  blend.inputImage = makeup
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

let blend = CIFilter.blendWithMask
    blend.backgroundImage = base
    blend.inputImage = makeup
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

let blend = CIFilter.blendWithMask
  blend.backgroundImage = base
  blend.inputImage = makeup
  blend.maskImage = matte
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

let blend = CIFilter.blendWithMask
    blend.backgroundImage = base
    blend.setInputImage = makeup
    blend.maskImage = matte
// Filtering Segmentation Mattes with Core Image

import CoreImage.CIFilterBuiltins

let blend = CIFilter.blendWithMask
    .backgroundImage = base
    .inputImage = makeup
    .maskImage = matte

let result = blend.outputImage
import CoreImage.CIFilterBuiltins

let blend = CIFilter.blendWithMask
    blend.backgroundImage = base
    blend.inputImage = makeup
    blend.maskImage = matte

let result = blend.outputImage
Saving Segmentation Mattes with Core Image
Saving Segmentation Mattes with Core Image

Saving to HEIF with matte CIImages

- Main CIImage
- Auxiliary matte CIImages

HEIF
Saving Segmentation Mattes with Core Image

Saving to HEIF with matte CIImages

```swift
context.writeHEIFRepresentation(of: result,
    to: url,
    format: .RGBA8,
    colorSpace: mainImage.colorSpace,
```
context.writeHEIFRepresentation(of: result,
    to: url,
    format: .RGBA8,
    colorSpace: mainImage.colorSpace,
    options: [.semanticSegmentationSkinMatteImage : skinImage,
Saving Segmentation Mattes with Core Image

Saving to HEIF with matte CIImages

```swift
context.writeHEIFRepresentation(of: result,
to: url,
format: .RGBA8,
colorSpace: mainImage.colorSpace,
options: [.semanticSegmentationSkinMatteImage : skinImage,
        .semanticSegmentationHairMatteImage : hairImage,
        .semanticSegmentationTeethMatteImage : teethImage])
```
Saving Segmentation Mattes with Core Image
Saving Segmentation Mattes with Core Image

Saving to HEIF with AVSemanticSegmentationMattes

Main CIImage

Auxiliary AVSemanticSegmentationMattes

HEIF
Saving Segmentation Mattes with Core Image

Saving to HEIF with AVSemanticSegmentationMattes

```swift
context.writeHEIFRepresentation(of: result,
    to: url,
    format: .RGBA8,
    colorSpace: mainImage.colorSpace,
```
Saving to HEIF with AVSemanticSegmentationMattes

```swift
context.writeHEIFRepresentation(of: result,
    to: url,
    format: .RGBA8,
    colorSpace: mainImage.colorSpace,
    options: [.AVSemanticSegmentationMattes :
        [ skinMatte, hairMatte, teethMatte]]
```
Summary
Summary

Creating matte images
Filtering matte images
Saving matte images
More Information

developer.apple.com/wwdc19/225

Capturing Depth in iPhone Photography  WWDC 2017

Introducing the Photos Frameworks  WWDC 2014