Capturing still and Live Photos

Configure and capture single or multiple still images, Live Photos, and other forms of photography.

Overview

Video captured on the iPhone 8, iPhone 8 Plus, and iPhone X running iOS 11 or later uses the HEVC codec by default. If your app shares the captured video using a system share sheet, the video will be automatically converted to a format compatible with the destination device.

AVFoundation supports many ways to capture photos. You can simply capture still HEIF or JPEG images, capture in RAW format for custom processing, snap several images in one shot, create Live Photos with motion and sound, and much more. In iOS, all photography workflows use the `AVCapturePhotoOutput` class.

> **Note**
>
> For macOS, see `AVCaptureStillImageOutput` instead.

Prepare for photo capture

First, set up an `AVCaptureSession` containing a supported camera device as one of its inputs and an `AVCapturePhotoOutput` as one of its outputs. (For details, see Choosing a capture device and Setting up a capture session.) Each camera device supports a wide range of resolution and frame rate settings. To easily get the best photo quality for the user's device, you can use the `photo` session preset instead of directly choosing individual settings.

Some capture options affect the internal configuration of the media capture pipeline. Because changing those options causes the pipeline to reconfigure itself, which takes time, enable them before offering the user the ability to shoot photos with those settings. Otherwise, the configuration delay could prevent the user from capturing a photo at the right moment.

For example, to configure the capture pipleline to support Live Photos, enable that property on the photo output, as shown below. After you've enabled Live Photo capture, you can choose for each individual shot whether to use still or Live Photo capture for each shot (see Capturing and saving Live Photos).

```
self.captureSession.beginConfiguration()

let photoOutput = AVCapturePhotoOutput()
photoOutput.isHighResolutionCaptureEnabled = true
photoOutput.isLivePhotoCaptureEnabled = photoOutput.isLivePhotoCaptureSupported

guard self.captureSession.canAddOutput(photoOutput) else { return }
self.captureSession.sessionPreset = .photo
self.captureSession.addOutput(photoOutput)

self.previewView.session = captureSession

self.captureSession.commitConfiguration()
self.captureSession.startRunning()
```

Choose settings

To capture a photo, first create an `AVCapturePhotoSettings` object describing the settings you want to use for that shot and the data format for the resulting still photo. For example:

- On supported devices, you can use the HEIF/HEVC format for improved image quality at smaller file sizes: use `init(format:)` and choose `hevc` for the video codec. On devices without HEVC support, use the default initializer `init()` to fall back to JPEG format.
- To shoot in RAW format, use `init(rawPixelFormatType:)` with one of the `availableRawPhotoPixelFormatTypes` supported by the photo output.

After creating a photo settings object, you can choose other settings for the photo. For example, the code below creates a settings object for HEIF/HEVC shooting, with automatic flash and image stabilization.

```
let photoSettings: AVCapturePhotoSettings
if self.photoOutput.availablePhotoCodecTypes.contains(.hevc) {
    photoSettings = AVCapturePhotoSettings(format:
        [AVVideoCodecKey: AVVideoCodecType.hevc])
} else {
    photoSettings = AVCapturePhotoSettings()
}
photoSettings.flashMode = .auto
photoSettings.isAutoStillImageStabilizationEnabled =
    self.photoOutput.isStillImageStabilizationSupported
```

Other possible photo settings include Live Photos, depth data capture, and multi-image (bracketed) capture, as well as options for embedding preview or thumbnail images in output image files. For more information, see Next Steps and More Capture Options below.

Capture the photo

Pass your photo settings object to the `capturePhoto(with:delegate:)` method to trigger photo capture with the settings you've chosen.

Important

Calling `capturePhoto(with:delegate:)` is analogous to pressing a camera's shutter button—each call starts a new photo capture. Each call requires a unique `AVCapturePhotoSettings` object with the settings for that capture. To reuse settings for multiple captures, see `init(from:)`.

Handle capture results

The `delegate` you pass to the `capturePhoto(with:delegate:)` method is an object to track the progress of and handle results from that photo capture. Capturing a photo is an asynchronous process with multiple steps that unfold over time. Because your app can trigger additional captures while earlier captures are still processing, your delegate implementation should be able to handle multiple captures at once. An easy way to handle concurrent captures is to define a class adopting the `AVCapturePhotoCaptureDelegate` protocol and create a separate instance of that class for each capture:

```
class PhotoCaptureProcessor: NSObject, AVCapturePhotoCaptureDelegate {
    // ...
}

let captureProcessor = PhotoCaptureProcessor()
self.photoOutput.capturePhoto(with: photoSettings, delegate: captureProcessor)
```

When your captured image data is ready for use, the photo output calls your delegate's `photoOutput(_:didFinishProcessingPhoto:error:)` method. You can use the resulting

`AVCapturePhoto` object there to display, process or save the image.

Topics

Next steps

Saving captured photos

Add an image and other data from a photo capture to the photo library.

Tracking photo capture progress

Monitor key events during capture to provide feedback in your camera UI.

Capturing and saving Live Photos

Capture Live Photos like those created in the system Camera app and save them to the Photos library.

More capture options

Capturing photos with depth

Get a depth map with a photo to create effects like the system camera’s Portrait mode (on compatible devices).

Capturing a bracketed photo sequence

Capture several photos at once, varying parameters like exposure duration or light sensitivity.

Capturing uncompressed image data

Get processed image data without compression to use for filtering or lossless output.

Capturing thumbnail and preview images

Enable delivery of reduced-size images with the main image in a photo capture.

See Also

Photo capture

{} Capturing consistent color images

Add the power of a photography studio and lighting rig to your app with the new Constant Color API.

Capturing photos in RAW and Apple ProRAW formats

Support professional photography workflows by enabling minimally processed image capture in your camera app.

Supporting Continuity Camera in Your Mac App

Incorporate scanned documents and pictures from a user's iPhone, iPad, or iPod touch into your Mac app using Continuity Camera.

`class AVCapturePhoto`

A container for image data from a photo capture output.

`class AVCaptureDeferredPhotoProxy`

A lightly-processed photo with data that the system may use to process and fetch a higher-resolution asset at a later time.

`class AVCapturePhotoOutput`

A capture output for still image, Live Photos, and other photography workflows.

`protocol AVCapturePhotoCaptureDelegate`

Methods for monitoring progress and receiving results from a photo capture output.

`class AVCapturePhotoOutputReadinessCoordinator`

An object that monitors changes to a photo output's capture readiness.

`protocol AVCapturePhotoOutputReadinessCoordinatorDelegate`

A delegate protocol to receive updates about a photo output's capture readiness.

~~`class AVCaptureStillImageOutput`~~

A capture output for capturing still photos.

Deprecated