Sample Code

Selecting a selfie based on capture quality

Compare face-capture quality in a set of images by using Vision.

Download

iOS 13.0+ | iPadOS 13.0+ | Xcode 14.2+

Overview

New in iOS 13, the Vision framework adds the Face Capture Quality metric to represent the capture quality of a given face in a photo. The sample app shows you how to use this metric to evaluate a collection of images of the same person and identify which one has the best capture quality.

Face Capture Quality is a holistic measure that considers scene lighting, blur, occlusion, expression, pose, focus, and more. It provides a score you can use to rank multiple captures of the same person. The pre-trained underlying models score a capture lower if, for example, the image contains low light or bad focus, or if the person has a negative expression. These scores are floating-point numbers normalized between `0.0` and `1.0`.

First the app creates and performs a `VNDetectFaceCaptureQualityRequest` and obtains face observations from the results:

```
let faceDetectionRequest = VNDetectFaceCaptureQualityRequest()
do {
    try handler.perform([faceDetectionRequest])
    guard let faceObservations = faceDetectionRequest.results as? [VNFaceObservation
        return
    }
    displayFaceObservations(faceObservations)
    if isCapturingFaces {
        saveFaceObservations(faceObservations, in: pixelBuffer)
    }
```

```
} catch {
    print("Vision error: \(error.localizedDescription)")
}
```

Then the app passes the face observations to `saveFaceObservations(_:in:)`, where it retrieves the `faceCaptureQuality` score for each capture. When the user presses down on the capture button, the app saves each capture's image data along with its quality score:

```
let faceDetectionRequest = VNDetectFaceCaptureQualityRequest()
do {
    try handler.perform([faceDetectionRequest])
    guard let faceObservations = faceDetectionRequest.results as? [VNFaceObservation
        return
    }
    displayFaceObservations(faceObservations)
    if isCapturingFaces {
        saveFaceObservations(faceObservations, in: pixelBuffer)
    }
} catch {
    print("Vision error: \(error.localizedDescription)")
}
```

Next, the app sorts the captures based on quality score:

```
// Sort faces in descending quality-score order.
savedFaces.sort { $0.qualityScore < $1.qualityScore }
```

Finally, the app displays the saved faces with their quality scores:

```
let savedFace = savedFaces[indexPath.item]
let faceImage = UIImage(contentsOfFile: savedFace.url.path)
cell.imageView.image = faceImage
cell.label.text = "\(savedFace.qualityScore)"
```

Configure the sample code project

To run this sample app, you need the following:

- Xcode 11 or later
- iPhone with iOS 13 or later

Connect the iPhone to the Mac over USB. The first time you run this sample app, the system prompts you to grant the app access to the camera. You must allow the sample app to access the camera for it to function correctly.

> **Note**
>
> This sample code project is associated with WWDC19 session 222: Understanding Images in Vision Framework.

See Also

Face and body detection

`class VNDetectFaceCaptureQualityRequest`

A request that produces a floating-point number that represents the capture quality of a face in a photo.

`class VNDetectFaceLandmarksRequest`

An image-analysis request that finds facial features like eyes and mouth in an image.

`class VNDetectFaceRectanglesRequest`

A request that finds faces within an image.

`class VNDetectHumanRectanglesRequest`

A request that finds rectangular regions that contain people in an image.

`class VNHumanObservation`

An object that represents a person that the request detects.