Class

VideoProcessor

An object that performs offline analysis of video content.

iOS 18.0+ | iPadOS 18.0+ | macOS 15.0+ | tvOS 18.0+ | visionOS 2.0+

```
final class VideoProcessor
```

Overview

A video processor streamlines video content analysis through frame-by-frame processing. Instead of manually extracting frames, the video processor manages the processing pipeline and delivers results through convenient async streams. You can attach multiple different analysis requests to the same `VideoProcessor` instance and they'll all operate on the same frames simultaneously. For example, you can provide a video and perform aesthetic scoring, face detection, and object recognition all at once without processing the video multiple times.

```
let videoURL = // A local `URL` path to the video you want to process.
let videoProcessor = VideoProcessor(videoURL)

// Calculate the aesthetics score for each frame.
let aestheticsScoresRequest = CalculateImageAestheticsScoresRequest()

// Perform face detection on each frame.
let faceDetectionRequest = DetectFaceRectanglesRequest()
```

Processing every single video frame provides the most accuracy, but can be computationally expensive and time-consuming. Before you begin video analysis, determine how many frames to process by using `VideoProcessor.Cadence`. `VideoProcessor.Cadence.time`

`Interval(_:)` processes frames at regular intervals, which provides consistent sampling throughout the video's duration.

```
do {
    let asset = AVURLAsset(url: videoURL)
    let totalDuration = try await asset.load(.duration).seconds
    let framesToEvaluate: Double = 100

    // Create a time interval that processes 100 frames.
    let interval = CMTime(
        seconds: totalDuration / framesToEvaluate,
        preferredTimescale: 600
    )
    let cadence = VideoProcessor.Cadence.timeInterval(interval)

    // Add the requests to get an `AsyncSequence` stream that provides access
    // to the observation results.
    let aestheticsScoreStream = try await videoProcessor.addRequest(aestheticsScores
                                                                     cadence: cadence
    let faceDetectionStream = try await videoProcessor.addRequest(faceDetectionReque
                                                                   cadence: cadence)


    // Start the analysis.
    videoProcessor.startAnalysis()
} catch {
    print("Error processing the video: \(error.localizedDescription)")
}
```

After you start processing a video, access the observations that the framework provides through an `AsyncSequence` stream. For example, the following code stores the timestamp and the aesthetics score:

```
var aestheticsResults: [CMTime: Float] = [:]
for try await observation in aestheticsScoreStream {
    if let timeRange = observation.timeRange {
        aestheticsResults[timeRange.start] = observation.overallScore
    }
}
```

Topics

Creating a video processor

`init(URL)`

Creates a video processor to perform framework requests against the video asset you specify.

Adding and removing a request

```
func addRequest<T>(T, cadence: VideoProcessor.Cadence?) async throws ->
some AsyncSequence<T.Result, any Error>
```

Adds a request to the video processor.

`enum Cadence`

A type that describes the video processing cadence.

`func removeRequest(any VisionRequest) async -> Bool`

Stops performing a request on future frames.

Starting the analysis

`func startAnalysis(of: CMTimeRange?)`

Begins analyzing video frames.

Cancelling the analysis

`func cancel() async`

Stops the video processor.

Relationships

Conforms To

`Sendable`, `SendableMetatype`

See Also

Utilities

`enum ComputeStage`

Types that represent the compute stage.