

[Vision](#) / [Original Objective-C and Swift API](#) / Detecting moving objects in a video

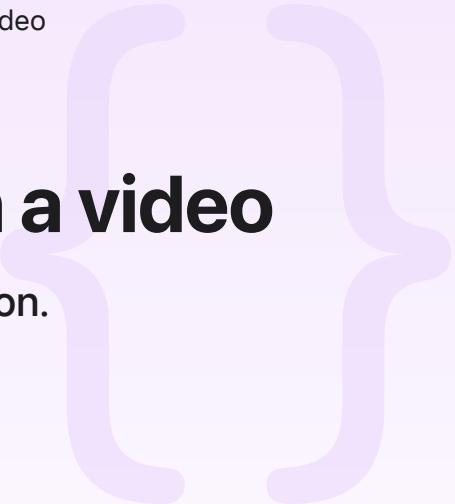
Sample Code

Detecting moving objects in a video

Identify the trajectory of a thrown object by using Vision.

[Download](#)

iOS 15.0+ | iPadOS 15.0+ | Xcode 14.1+



Overview

The Vision framework provides the ability to detect trajectories of objects in a video. The algorithm looks at the differences between frames of a video and identifies objects that travel along a parabolic path. A single object — like a bouncing ball — may produce multiple trajectories.

The sample code project shows you how to configure a trajectory detection request to analyze sample buffers. It explores how to set up a capture session to analyze a live-capture feed, and an asset reader to analyze a prerecorded video. When the sample app detects a trajectory, it illustrates it by displaying the detected path on the screen.

For more information about identifying trajectories, see [Identifying Trajectories in Video](#).

Configure the project and prepare your environment

You must run the sample code project on a physical device with an A12 processor or later.

The sample's current configuration looks for a small object moving left to right in a video. Try the sample app by downloading and analyzing a [prerecorded video](#) of a person tossing a bean bag. For live capture, the sample app requires a stable scene with a fixed camera position and stationary background, so mount your iOS device to a tripod and keep it fixed on the field of view. You need to modify the trajectory request's configuration based on your conditions when you use your own video.

Create a trajectory request

Before the sample app creates a trajectory request, it gets sample buffers based on the selected source — live capture or a prerecorded video — in `CameraViewController`. After the `AVCaptureSession` or `AVAssetReader` retrieves a sample buffer, it passes to the `ContentAnalysisViewController`. The sample app creates a `VNImageRequestHandler` to perform the trajectory request with.

```
let visionHandler = VNImageRequestHandler(cmSampleBuffer: buffer,  
                                         orientation: orientation,  
                                         options: [:])
```

The sample app sets up a `VNDetectTrajectoriesRequest` object to define what the sample app looks for. It looks for trajectories after 1/60 second of video, and returns trajectories with a length of 6 or greater. Generally, developers use a shorter length for real-time apps, and longer lengths to observe finer and longer curves.

```
detectTrajectoryRequest = VNDetectTrajectoriesRequest(frameAnalysisSpacing: CMTime(\  
                                         trajectoryLength: 6) { [weak self]  
  
    guard let results = request.results as? [VNTrajectoryObservation] else {  
        return  
    }  
  
    DispatchQueue.main.async {  
        self?.processTrajectoryObservation(results: results)  
    }  
}
```

After the sample app creates the `VNDetectTrajectoriesRequest`, it configures additional options that describe the radius of the object it's looking for. To improve the efficiency of the analysis, it also specifies the region of interest.

```
// Following optional bounds by checking for the moving average radius  
// of the trajectories the app is looking for.  
detectTrajectoryRequest.objectMinimumNormalizedRadius = 10.0 / Float(1920.0)  
detectTrajectoryRequest.objectMaximumNormalizedRadius = 30.0 / Float(1920.0)  
  
// Help manage the real-time use case to improve the precision versus delay tradeoff  
detectTrajectoryRequest.targetFrameTime = CMTimeMake(value: 1, timescale: 60)
```

```
// The region of interest where the object is moving in the normalized image space.  
detectTrajectoryRequest.regionOfInterest = normalizedFrame
```

After the sample app configures the trajectory request for the buffer, it processes the list of VNTrajectoryObservation results.

Process the trajectory observation results

The sample app configuration targets the prerecorded video from the configuration section. The VNDetectTrajectoriesRequest follows objects moving on a parabolic path, and requires more than a single data point (trajectory length). After a request gathers enough data points to recognize the trajectory — a length of at least 5 — it passes the observation results that contain the trajectory information.

The first step in processing the results involves filtering the VNTrajectoryObservation based on the following conditions:

- The trajectory moves from left to right.
- The trajectory starts in the first half of the region of interest.
- The trajectory length increases to 8, which indicates a throw instead of smaller movements.
- The trajectory contains a parabolic equation constant a , less than or equal to 0, and implies there are either straight lines or downward-facing lines.
- The trajectory confidence is greater than 0.9.

When the results meet the above conditions, the sample app deems the observation a valid trajectory. The sample app confirms the trajectory and makes any necessary correction to the path. If a left-to-right moving trajectory begins too far from a fixed region, the sample extrapolates it back to the region by using the available quadratic equation coefficients.

```
for trajectory in results {  
    // Filter the trajectory.  
    if filterParabola(trajectory: trajectory) {  
        // Verify and correct an incomplete path.  
        trajectoryView.points = correctTrajectoryPath(trajec  
  
        // Display a transition.  
        trajectoryView.performTransition(.fadeIn, duration: 0.05)  
  
        // Determine the size of the moving object that the app tracks.  
        print("The object's moving average radius: \(trajectory.movingAverageRadius)
```

}

}

The sample app displays valid trajectories on the screen with particle effects by using [Sprite Kit](#).

See Also

Trajectory detection

 Identifying Trajectories in Video

Gain new insights into your video data by using Vision to detect trajectories.

`class VNDetectTrajectoriesRequest`

A request that detects the trajectories of shapes moving along a parabolic path.