Sample Code

Building a feature-rich app for sports analysis

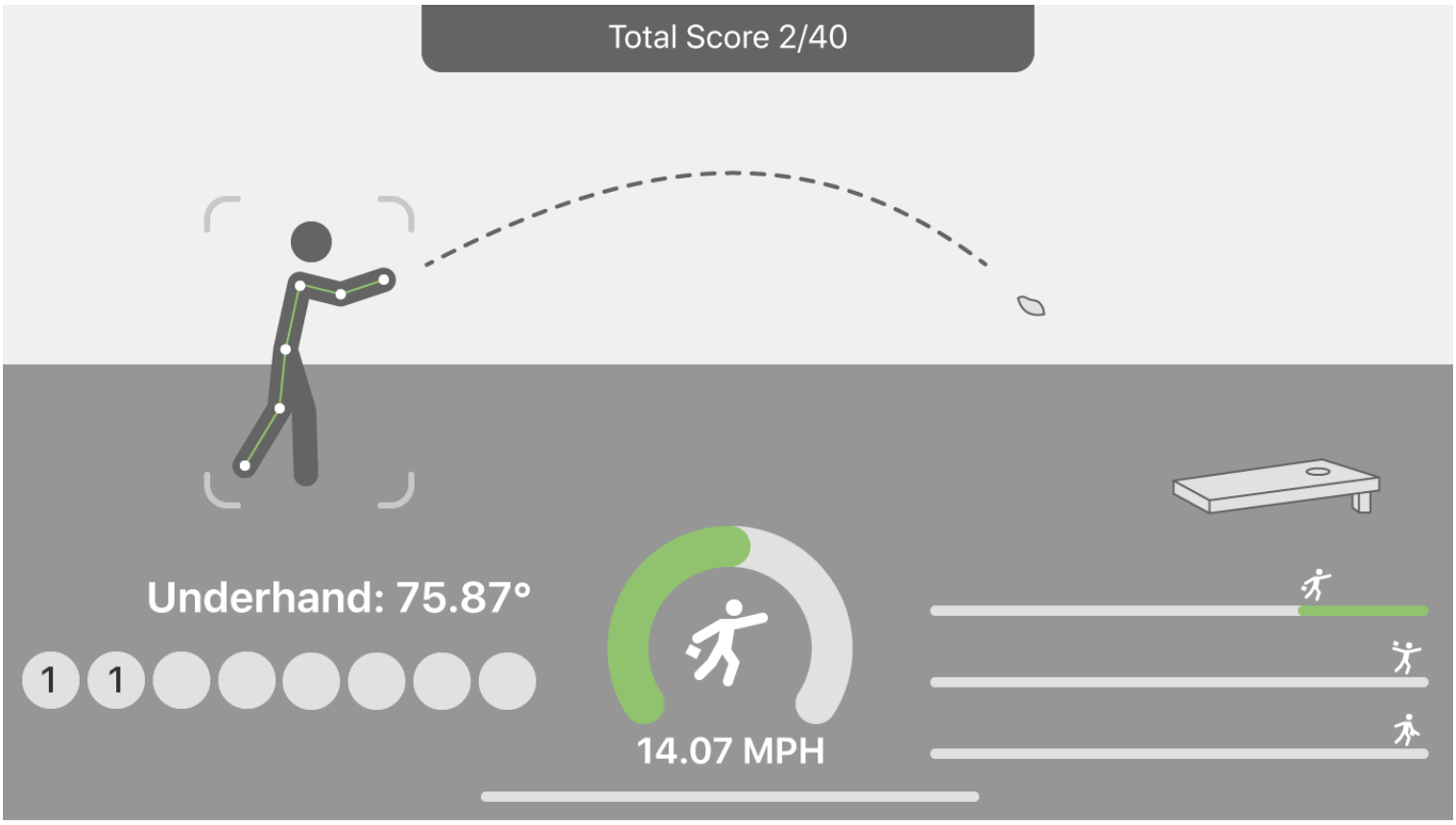
Detect and classify human activity in real time using computer vision and machine learning.

Download

iOS 14.0+ | iPadOS 14.0+ | Xcode 14.2+

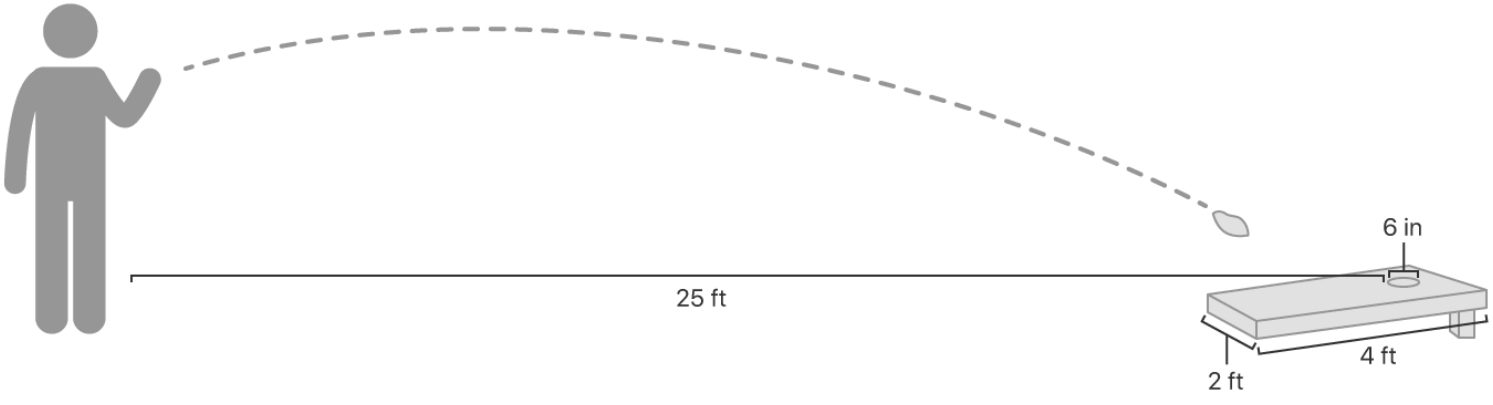
Overview

The Action & Vision sample app leverages several capabilities available in Vision and Core ML in iOS 14 and later. The app provides an example of how you can use these technologies together to help players improve their bean-bag tossing performance.



> **Note**
>
> This sample code project is associated with the WWDC20 session 10099: Explore the Action & Vision App.

The app analyzes player performance in a game of bean bag toss. In this easy-to-learn game, you throw bean bags at a 4 x 2-foot board that has a 6-inch hole in one end. You stand 25 feet away from the center hole of the board, and score points if you toss the bean bag onto the board, and score additional points if it goes through the hole. To make gameplay more interesting, the app adds a style dimension to the scoring. It analyzes your throw and adds additional points based on your throw style, including overhand, underhand, or underleg.



After each throw, the app also provides real-time feedback that indicates your throw's type, speed, and trajectory, along with your score.

Configure the project and prepare your environment

You need to run the sample app on a physical device with an A12 processor or later, running iOS 14.

Some Vision algorithms require a stable scene with a fixed camera position and stationary background. To analyze your own gameplay, mount your iOS device to a tripod and keep it fixed on the field of play. Alternatively, try the app by downloading and analyzing a prerecorded video of a bean bag toss player in action.

Analyze the field of play

Before gameplay can begin, the app analyzes the scene to find the location of the gameboard. It detects its location by using an instance of VNCoreMLRequest to run inference on a custom object-detection model created using Create ML. The app has a model trained using photos of gameboards, taken from various angles and distances, in both indoor and outdoor environments. To further improve the model's accuracy, the training included images with people in the frame, and bean bags on and around the board. The app performs the request and retrieves its top observation to find the gameboard's bounding rectangle.

Next, the app performs a VNDetectContoursRequest to detect the contours of the gameboard's edges and uses them to determine the normalized pixel dimensions of the board, and the location of the hole. Because the app previously calculated the bounding rectangle of the detected board, it sets the bounding rectangle as the request's region of interest, which helps the request significantly reduce noise and improve performance. Because the app knows the real-world size of the gameboard, it accurately determines the pixel dimensions of the field of play.

To learn more about about contour detection, see the WWDC20 session 10673: Explore Computer Vision APIs.

Determine scene stability

Some Vision algorithms require a stable scene to produce accurate results. To determine scene stability, the app uses VNTranslationalImageRegistrationRequest, a request that compares two images to calculate the x-axis and y-axis shift between them. The app analyzes video frames until the shift reaches a minimum difference threshold, at which point the app considers the scene stable.

Identify the player

With the field of play established, the app is ready to start the game when a player enters the scene. To detect a player entering the frame, the app performs VNDetectHumanBodyPoseRequest, a request available in iOS 14, that detects key body points of

the face, torso, arms, and legs. The app performs this request on each video frame, and uses the data points it produces to draw a bounding box around the player. The app also uses the detected body points to determine the throw type.

For more information about using human body-pose detection, see Detecting Human Body Poses in Images and the WWDC20 session 10653: Detect Body and Hand Pose with Vision.

Detect the player's throw trajectory

The app uses Vision's VNDetectTrajectoriesRequest to analyze the player's throw. This request follows objects moving on a parabolic path. Detecting a parabola requires more than a single data point, so this is a *stateful* request that builds evidence of the object's movement over time. The app creates a single instance of the request and performs it multiple times over a sequence of video frames. After the request collects enough data points to recognize the trajectory, it calls its completion handler, and passes it the observations that contain the trajectory coordinates and timestamps. The app provides a data visualization by progressively drawing a line that traces the bag's trajectory. When the request stops producing observations for more than 20 frames, the app considers the throw complete, and calculates the points that the player scored.

The app knows the pixel dimensions of the gameboard, and uses them as a reference to determine the distance of the player's throw. Because the request also provides the detected trajectory's duration, the app uses the time and distance to measure the throw's speed.

To learn more about using `VNDetectTrajectoriesRequest`, see Identifying Trajectories in Video.

> **Note**
>
> `VNDetectTrajectoriesRequest` detects multiple, concurrent trajectories, so an app needs to apply its own business logic to filter the trajectories to only those of interest. For example, the sample app knows the location of the gameboard, and it discards any trajectories that move in the opposite direction.

Determine the throw type

To determine the throw type, the app uses another custom Core ML model, one trained using the Action Classification template available in Create ML. The app's model training used video clips of people performing overhand, underhand, and underleg throws, and also included a negative training set with players performing actions other than throwing.

As discussed previously, the app uses VNDetectHumanBodyPoseRequest to detect the player's location. In each of the observations the request returns, the request also provides the normalized coordinates for various detected body points. The app retrieves these points from the request as a

Core ML MLMultiArray and passes them as inputs to the Action Classifier. The model runs its predictions and returns the labeled results, which indicate the throw type, and a confidence value in the accuracy of the prediction.

To learn more about Action Classification, see Creating an Action Classifier Model.