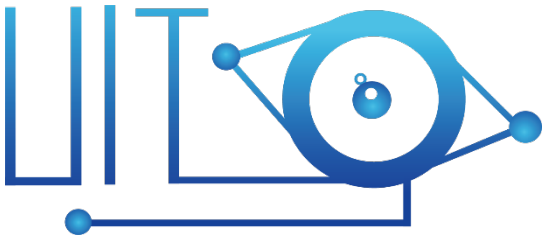


Hong Kong Univisual Intelligent Technology Limited

HKUITLTD PoseEstim Framework Documentation and Tutorial

Date: 09/10/2020

Table of Contents

Part 1: Documentation

Class *RunthisModel*

This is a helper class for analysing the image captured by the camera in real time. You can obtain human body skeleton predicted by AI, and use this result for rendering the skeleton on UI. It is in “RunThisModel/RunthisModel.swift”.

Constructor:

- **init ()**

  This constructor initializes a model analyser for later images analysing purpose.

Function:

- func **Run (pb: CVPixelBuffer, olv: CGRect, pv: CGRect) -> (Result, Times)?**

  This function passes the image buffer and the corresponding UI View information to the model analyser. A tuple of (result, times) which are two custom objects will be returned.

  Parameters:

  1. **pb**: CVPixelBuffer

     The image needs to be processed.

  2. **olv**: CGRect

     The size of the view which will be captured by the neural networks**.**

  3. **pv**: CGRect

     The size of the whole screen.

Class *GiveFeedBack*

This is a helper class for calculating score and generating comments for each selected posture. It is in "PoseAnalyzer/GiveFeedBack.swift".

Constructor:

- **init (user_input_result: Result, user_input_pose: Pose)**

  This class requires 2 arguments (i.e. Result, Pose) to be initialised. The score and comments generated are based on the arguments.

Parameters:

1. **user_input_result**: Result

   The result generated by the function *RunThisModel.Run*(…).

2. **user_input_pose**: Pose

   Pose standard to be compared with the user_input_result.

Functions:

1. func **getScore () -> Double**

   This method returns a double value of the score calculated based on the parameter "user_input_pose" of the public constructor. The value ranged from 0 to 100.

2. func **getComments () -> [String]**

   This method gives a string array that stores all comments regarding the comparison between result and selected pose standard.

3. func **getDetailedScore () -> [Double]**

   This method provides a double array which contains the detailed score.

Class *OverlayView*: UIView

This class is inherited from *UIView* and some methods are added for rendering skeleton on a view conveniently. The two public methods will call ***setNeedsDisplay ()*** in *UIView* for you, so you do not need to call it again. All you need to do for rendering the result is to give a *Result* to it. It is in "Views/OverlayView.swift".

Functions:

1. func **drawResult (result: Result, bounds: CGRect, position: AVCaptureDevice.Position)**

   This method will draw all the dots and lines contained in the *Result* on the view. It will call ***setNeedsDisplay()*** and notify the system to call ***override func draw()*** in this class.

   Parameter:

   1. **result**: Result

      The result that contain user's performance, usually generated by the function *RunThisModel.Run*(…).

   2. **bounds**: CGRect

      The location and size of the overlayView.

   3. **position**: AVCaptureDevice.Position

      The position of camera using currently, either back, front or unspecified.

2. func **clear ()**

   This method will clear all the things rendered on the view. It will call ***setNeedsDisplay ()*** and notify the system to call ***override func draw ()*** in this class.

Class *CameraFeedManager*: NSObject, AVCaptureVideoDataOuputSampleBufferDelegate

---

This class manages the camera usage of the device and provides a delegate to pass the real-time video frame to the AI model. It is in “CameraFeed/CameraFeedManager.swift”.

Constructor:

- **init (previewView: PreviewView)**

  This creates and configures an AVCaptureSession to handle video data I/O.

Parameter:

- **previewView**: PreviewView

  The previewView that display camera frame.

Functions:

1. func **checkCameraConfigurationAndStartSession ()**

   This method enables the application to start using the camera as a video data input and sending the data to the neural networks if the camera is configured successfully during the initialisation of this class. Otherwise, alert will be raised.

2. func **stopSession ()**

   Ends the running session and remove the corresponding observers.

3. func **showCurrentInput () -> AVCaptureDevice.Position**

   This function returns the position of the current camera input, in AVCaptureDevice.Position type.

4. func **switchCamera ()**

   Call this function to switch front and back cameras.

Protocol *CameraFeedManagerDelegate*: class

This is a delegate of class CameraFeedManager that provides function to deliver CVPixelBuffer of frame. Some functions are also provided to notify the alert and errors raised by the CameraFeedManager.

Functions:

1. func **cameraFeedManager (_ manager: CameraFeedManager, didOutput pixelBuffer: CVPixelBuffer)**

   This function delivers the pixel buffer of the current frame seen by the device's camera.

2. func **cameraFeed Manager (_ manager: CameraFeedManager, sessionWasInterrupted canResumeManually: Bool)**

   This function is called when the session was interrupted.

3. func **cameraFeedManagerDidEncounterSessionRunTimeError (_ manager: CameraFeedManager)**

   This function is called when a session runtime error occurred.

4. func **cameraFeedManagerDidEndSessionInterruption (_ manager: CameraFeedManager)**

   This function is called when the session interruption has ended.

5. func **presentVideoConfigurationErrorAlert (_ manager: CameraFeedManager)**

   This function is called when there is an error in the video configuration.

6. func **presentCameraPermissionDeniedAlert (_ manager: CameraFeedManager)**

   This function is called when the camera permissions have been denied.

enum *Pose*: String

This enum class will provide a set of unified string for different poseture. You can use this set of string to select a pose and generate a score and comments for this pose. The set of new types defined in Pose is shown below.

1. Navasana	For pose Navasana
2. ustrasana	For pose Ustrasana
3. ardha_uttanasana	For pose ArdhaUttanasana

struct *Line*

This struct stores the information of the end points of a line. The data type of the end points is CGPoint.

Variables:

1. let **from**: CGPoint

   The starting point of the line.

2. let **to**: CGPoint

   The end point of the line.

struct *Result*

This struct includes the pose result's score and the body skeleton points information. It is in "CustomObjects/TimeResult.swift".

Variables:

1. var **dots**: [CGPoint]

   A CGPoint array that contains body skeleton points.

2. var **lines**: [Line]

   A line object array that contains the joint connection lines.

3. var **score**: Float

   The score of user's performances.

---

struct *Times*

A structure that stores processing time. It is in "CustomObjects/TimeResult.swift".

Variables:

1. var **preprocessing**: Double

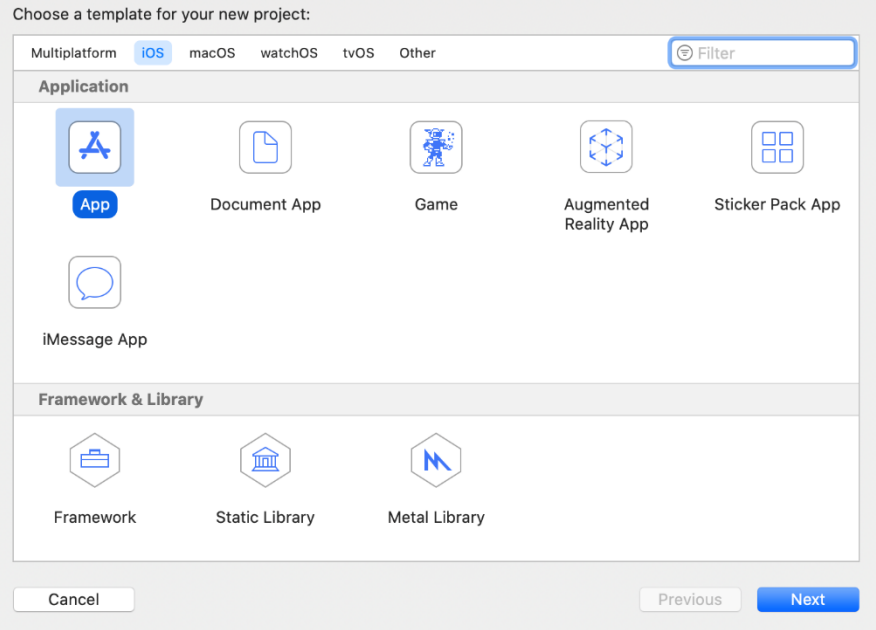
2. var **inference**: Double

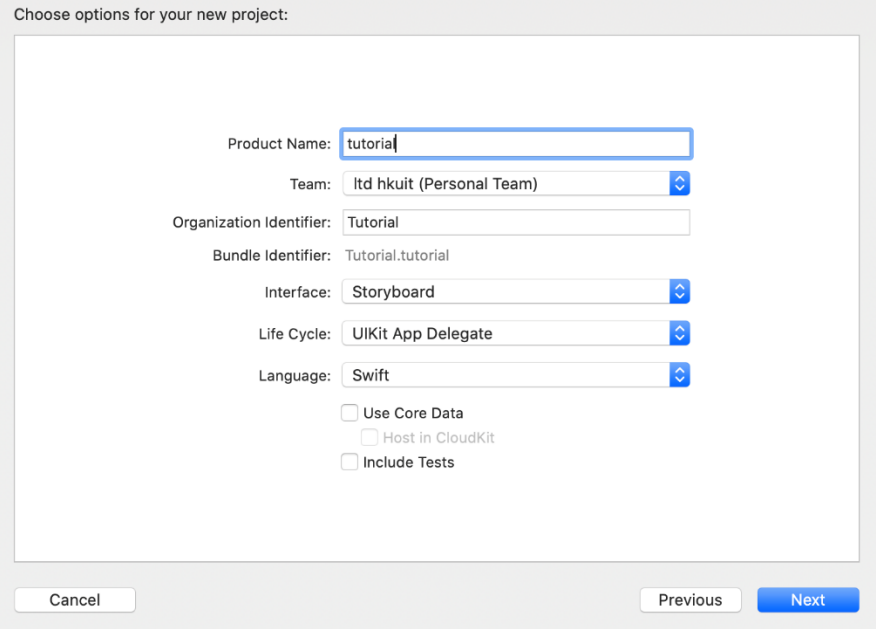
3. var **postprocessing**: Double

Part 2: Tutorial

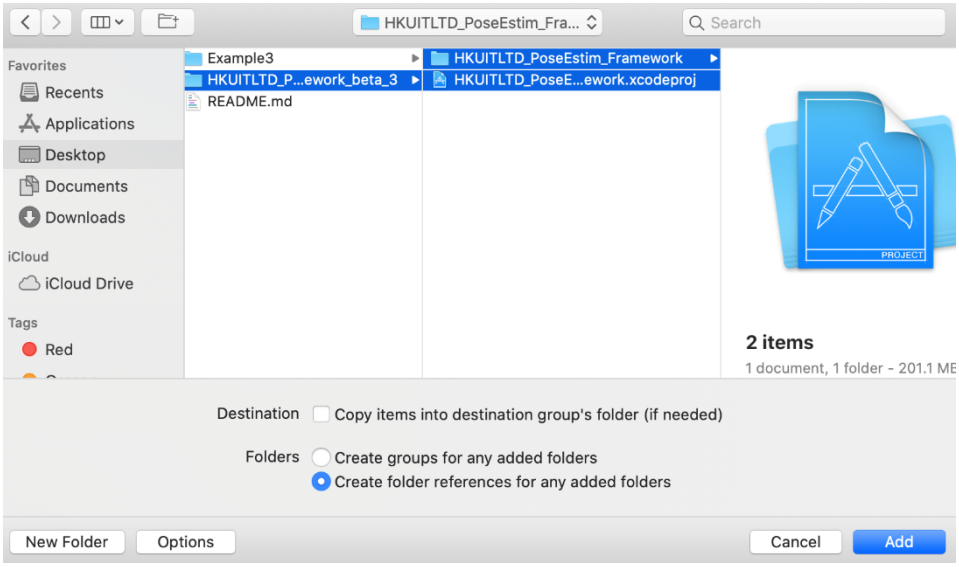
Part I: Project setup

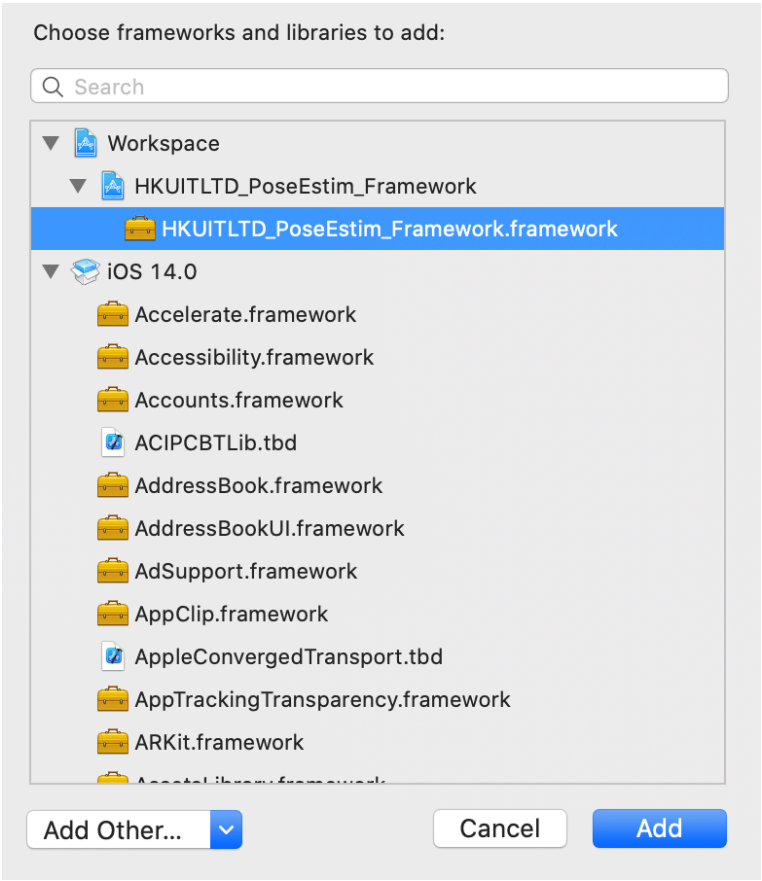
1. Create an iOS App project and choose “Storyboard” as user interface





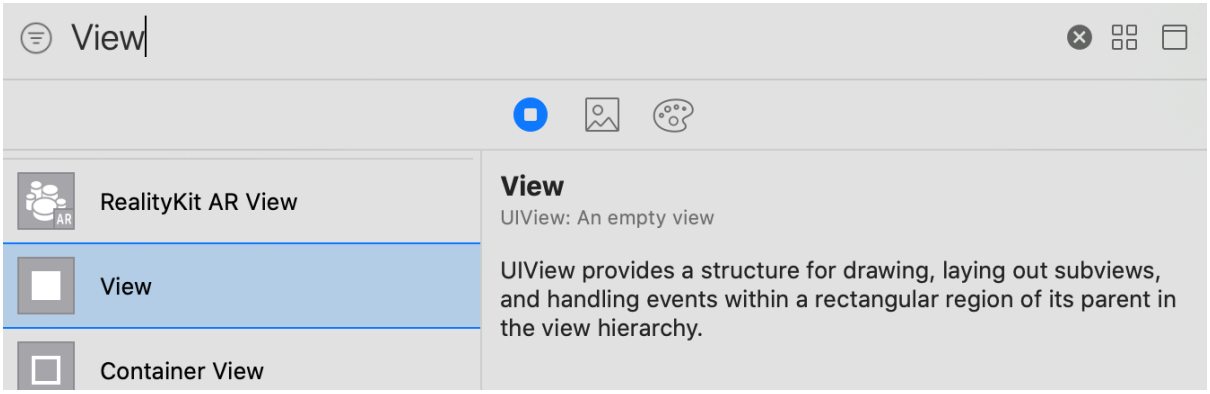
2. Add and include framework files

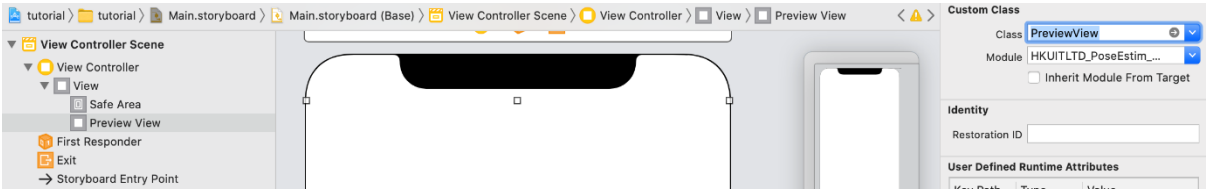




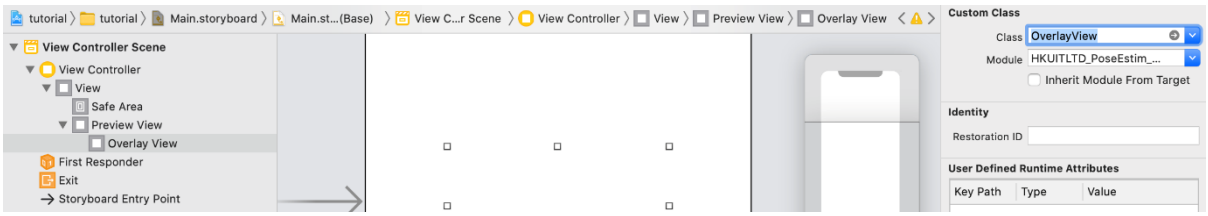
Part II: Main Storyboard Components

1. Open the main storyboard, add a View object and morph it to previewView class



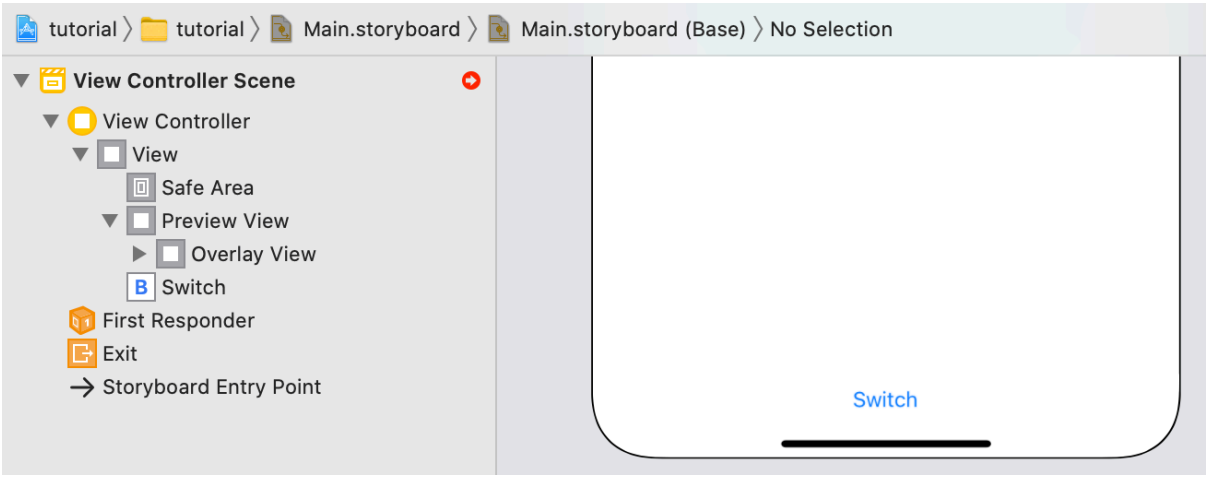


2. Add another View object inside the previewView object and morph it to overlayView class

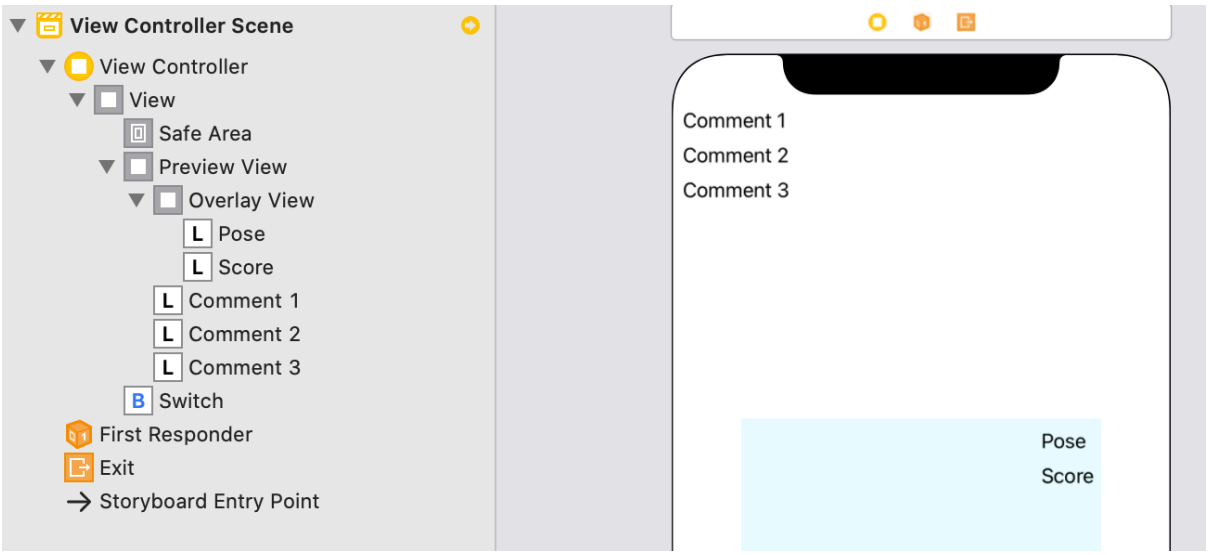


The colour of the overlayView can be customised for better distinguishment

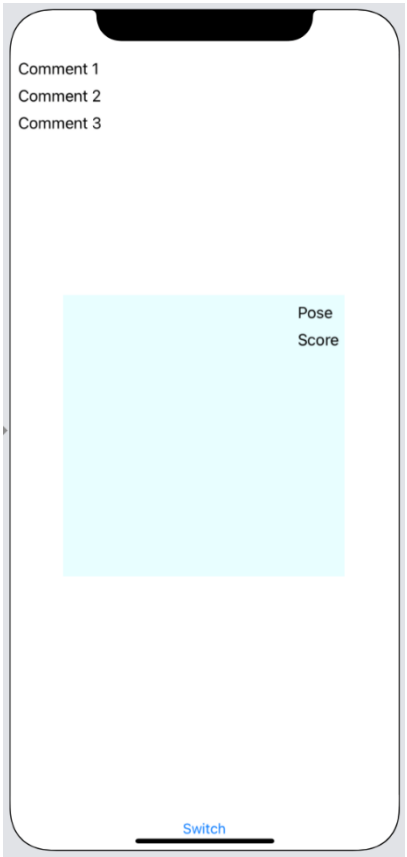
3. Add a button object for switching camera



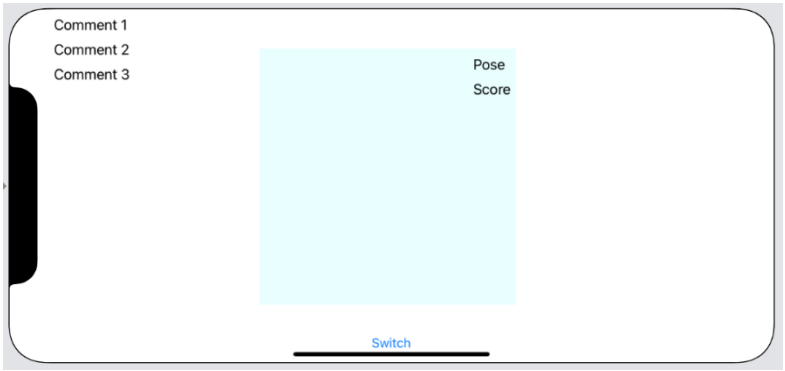
4. Add 3 labels on the previewView layer to show comments and 2 labels on the overlayView layer to display the selected pose and the user's performance score



5. Customize the objects' constraints for the display consistency



Portrait View



Landscape View

The overlayView's width to height ratio must be 1:1 and its size must be smaller than the previewLayer's size

Part III: View Controller Configuration

1. Open the ViewController.swift and import the required library

```
import UIKit
import AVFoundation
import HKUITLTD_PoseEstim_Framework

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }


}
```

2. Add IBOutlets and IBAction of the view components

```
import UIKit
import AVFoundation
import HKUITLTD_PoseEstim_Framework

class ViewController: UIViewController {

    @IBOutlet weak var previewView: PreviewView!
    @IBOutlet weak var overlayview: OverlayView!
    @IBOutlet weak var comment1: UILabel!
    @IBOutlet weak var comment2: UILabel!
    @IBOutlet weak var comment3: UILabel!
    @IBOutlet weak var pose: UILabel!
    @IBOutlet weak var score: UILabel!
    @IBAction func switchCamera(_ sender: UIButton) {
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }


}
```

3. Add variables for setting up camera feed manager

```
@IBAction func switchCamera(_ sender: UIButton) {
}

private var overlayViewFrame:CGRect?
private var previewViewFrame:CGRect?

private lazy var cameraCapture = CameraFeedManager(previewView: previewView)
private var thisModel: RunthisModel?

var threadCount: Int = Constants.defaultThreadCount
var delegate: Delegates = Constants.defaultDelegate

private let minScore: Float = 0.5

override func viewDidLoad() {
```

Developer can adjust the minScore from 0 to 1. If the value minScore is too small, the model will draw the resultant lines even there is no human body detected.

4. Modify the viewDidLoad function and implement CameraFeedManagerDelegate

```
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        thisModel = RunthisModel()
        cameraCapture.delegate = self
    }

}

extension ViewController: CameraFeedManagerDelegate {
    func cameraFeedManagerDidEndSessionInterruption(_ manager: CameraFeedManager) {

    }

    func cameraFeedManagerDidEncounterSessionRunTimeError(_ manager: CameraFeedManager) {

    }

    func presentCameraPermissionsDeniedAlert(_ manager: CameraFeedManager) {

    }

    func presentVideoConfigurationErrorAlert(_ manager: CameraFeedManager) {

    }

    func cameraFeedManager(_ manager: CameraFeedManager, sessionWasInterrupted canResumeManually: Bool) {

    }

    func cameraFeedManager(_ manager: CameraFeedManager, didOutput pixelBuffer: CVPixelBuffer) {

    }
}
```

5. Override other view control functions

```
  override func viewDidLoad() {
      super.viewDidLoad()
      // Do any additional setup after loading the view.
      thisModel = RunthisModel()
      cameraCapture.delegate = self
  }

  override func viewWillAppear(_ animated: Bool) {
      super.viewWillAppear(animated)
      cameraCapture.checkCameraConfigurationAndStartSession()
  }

  override func viewWillDisappear(_ animated: Bool) {
      cameraCapture.stopSession()
  }

  override func viewDidLayoutSubviews() {
      overlayViewFrame = overlayview.frame
      previewViewFrame = previewView.frame

      let orientation: UIDeviceOrientation = UIDevice.current.orientation

      switch orientation {
      case .portrait:
          previewView.previewLayer.connection?.videoOrientation = .portrait
      case .portraitUpsideDown:
          previewView.previewLayer.connection?.videoOrientation = .portraitUpsideDown
      case .landscapeLeft:
          previewView.previewLayer.connection?.videoOrientation = .landscapeRight
      case .landscapeRight:
          previewView.previewLayer.connection?.videoOrientation = .landscapeLeft
      default:
          previewView.previewLayer.connection?.videoOrientation = .portrait
      }
  }
```

6. Modify cameraFeedManager

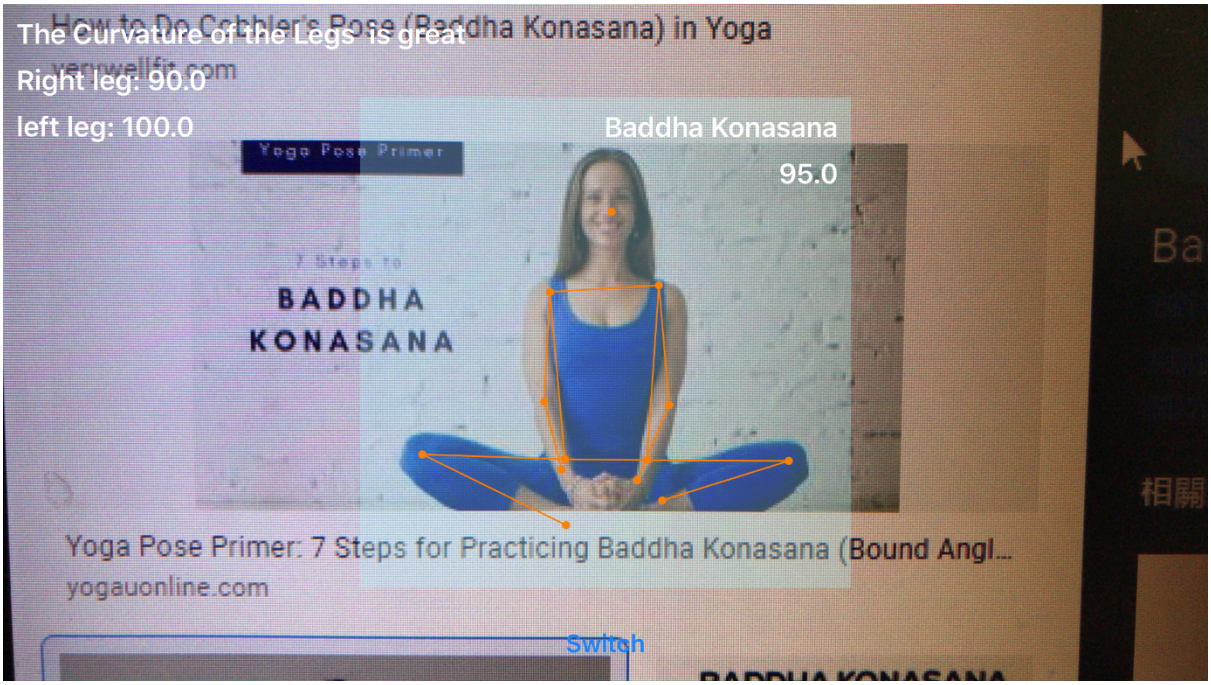
```
func cameraFeedManager(_ manager: CameraFeedManager, didOutput pixelBuffer: CVPixelBuffer) {
    DispatchQueue.main.async {
        self.overlayViewFrame = self.overlayview.frame
        self.previewViewFrame = self.previewView.frame
    }

    let (result, _) = (thisModel?.Run(pb: pixelBuffer, olv: self.overlayViewFrame!, pv: self.previewViewFrame!))!
    let userSelectedPose = Pose.BaddhaKonasana // select any one pose you like
    let feedback = GiveFeedBack(user_input_result: result, user_input_pose: userSelectedPose)
    let score = feedback.getScore()
    let comments = feedback.getComments()

    DispatchQueue.main.async {
        if result.score >= self.minScore {
            self.pose.text = userSelectedPose.rawValue
            self.score.text = String(score)
            self.comment1.text = comments[0]
            self.comment2.text = comments[1]
            self.comment3.text = comments[2]

            let position = self.cameraCapture.showCurrentInput()
            self.overlayview.drawResult(result: result, bounds: self.overlayview.bounds, position: position)
        }
        else {
            self.overlayview.clear()
            return
        }
    }
}
```

The overlayView only displays result when result score is greater than minScore.

7. Modify IBAction function of the button

```
@IBAction func switchCamera(_ sender: UIButton) {
    self.cameraCapture.switchCamera()
}
```

8. Result



Part IV: ViewController.swift

```
//
//  ViewController.swift
//  tutorial
//
//  Created by hkuit155 on 9/10/2020.
//

import UIKit
import AVFoundation
import HKUITLTD_PoseEstim_Framework

class ViewController: UIViewController {

    @IBOutlet weak var previewView: PreviewView!
    @IBOutlet weak var overlayview: OverlayView!
    @IBOutlet weak var comment1: UILabel!
    @IBOutlet weak var comment2: UILabel!
    @IBOutlet weak var comment3: UILabel!
    @IBOutlet weak var pose: UILabel!
    @IBOutlet weak var score: UILabel!
    @IBAction func switchCamera(_ sender: UIButton) {
        self.cameraCapture.switchCamera()
    }

    private var overlayViewFrame:CGRect?
    private var previewViewFrame:CGRect?

    private lazy var cameraCapture = CameraFeedManager(previewView: previewView)
    private var thisModel: RunthisModel?

    var threadCount: Int = Constants.defaultThreadCount
    var delegate: Delegates = Constants.defaultDelegate

    private let minScore: Float = 0.6

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        thisModel = RunthisModel()
        cameraCapture.delegate = self
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        cameraCapture.checkCameraConfigurationAndStartSession()
    }

    override func viewWillDisappear(_ animated: Bool) {
        cameraCapture.stopSession()
    }

    override func viewDidLayoutSubviews() {
        overlayViewFrame = overlayview.frame
        previewViewFrame = previewView.frame

        let orientation: UIDeviceOrientation = UIDevice.current.orientation

        switch orientation {
        case .portrait:
            previewView.previewLayer.connection?.videoOrientation = .portrait
        case .portraitUpsideDown:
            previewView.previewLayer.connection?.videoOrientation
= .portraitUpsideDown
        case .landscapeLeft:
```

```
            previewView.previewLayer.connection?.videoOrientation = .landscapeRight
        case .landscapeRight:
            previewView.previewLayer.connection?.videoOrientation = .landscapeLeft
        default:
            previewView.previewLayer.connection?.videoOrientation = .portrait
        }
    }

}

extension ViewController: CameraFeedManagerDelegate {
    func cameraFeedManagerDidEndSessionInterruption(_ manager: CameraFeedManager) {

    }

    func cameraFeedManagerDidEncounterSessionRunTimeError(_ manager:
CameraFeedManager) {

    }

    func presentCameraPermissionsDeniedAlert(_ manager: CameraFeedManager) {

    }

    func presentVideoConfigurationErrorAlert(_ manager: CameraFeedManager) {

    }

    func cameraFeedManager(_ manager: CameraFeedManager, sessionWasInterrupted
canResumeManually: Bool) {

    }

    func cameraFeedManager(_ manager: CameraFeedManager, didOutput pixelBuffer:
CVPixelBuffer) {
        DispatchQueue.main.async {
            self.overlayViewFrame = self.overlayview.frame
            self.previewViewFrame = self.previewView.frame
        }

        let (result, _) = (thisModel?.Run(pb: pixelBuffer, olv:
self.overlayViewFrame!, pv: self.previewViewFrame!))!
        let userSelectedPose = Pose.BaddhaKonasana // select any one pose you like
        let feedback = GiveFeedBack(user_input_result: result, user_input_pose:
userSelectedPose)
        let score = feedback.getScore()
        let comments = feedback.getComments()

          DispatchQueue.main.async {
           if result.score >= self.minScore {
               self.pose.text = userSelectedPose.rawValue
               self.score.text = String(score)
               self.comment1.text = comments[0]
               self.comment2.text = comments[1]
               self.comment3.text = comments[2]

               let position = self.cameraCapture.showCurrentInput()
               self.overlayview.drawResult(result: result, bounds:
self.overlayview.bounds, position: position)
            }
           else {
               self.overlayview.clear()
               return
            }
        }
    }
}
```