2014

Badminton SW v1.0

USING VIRTUAL REALITY TECHNOLOGY IN DESIGN PROGRAM TO TEACH SOME OF THE BASIC SKILLS IN BADMINTON

RESEARCHER: MOSTAFA AHMED SHAWKY HASSAN

ABDELRAHMAN HOSNY MOHAMMED | Computer Science Teaching Assistant, Assiut University

Table of Contents

Introduction

Mostafa A. Shawky is a Teaching Assistant at the Faculty of Physical Education, Helwan University. He is pursuing his Master degree in physical and athletic education. The research title is "Using Virtual Reality Technology in Design Program to teach some of the Basic Skills in Badminton", and is conducted under supervision of Prof. Mostafa K. Hamad[1], Associate Prof. Tamer M. Abdelfattah[2] and Associate Prof. Ahmed A. Alsyofy[3].

Referring to the abstract of the research thesis, attached in another document, the purpose of this research is to use virtual reality technology in designing a software program to teach some of the basic skills in badminton. Therefore, this document spots the light on the part of the study that requires a software detailing the software requirements, lifecycle and deliverables.

Terminologies

Terminology	Definition
Badminton	A racquet sport that is played by either two opposing players (singles) or two opposing pairs (doubles). Players score points by striking a shuttlecock with their racquet so that it passes the net and lands on their opponent court.
Virtual Reality	A computer-simulated environment that can simulate physical presence in places in the real world or imagined worlds. It can recreate sensory experiences.
Interactive Video	A software that combines two major factors of education and learning: research and justification. It enables the learner to actively interact and then respond. The learner can pause, repeat and stop, that is controlling his self-actions.
Microsoft Kinect	A hardware device that can be connected to the PC via USB. It captures the motion of a human body without using markers. It will be used in our software as the main sensor.
Badminton SW	Refers to the piece of software that is developed as part of this research thesis to prove the concepts. Badminton SW is the targeted output of this document.

[1] Prof. Mostafa K. Hamad is the head of Athletic Kinetic Science Department, Faculty of Physical Education, Helwan University.
[2] Prof. Tamer M. Abdelfattah is associate professor at the Department of Production Design and Engineering, Faculty of Engineering, Ain Shams University.
[3] Prof. Ahmed A. Alsyofy is associate professor at the Department of Athletic and Kinetic Science, Faculty of Physical Education, Helwan University.

Software Goals (Requirements)

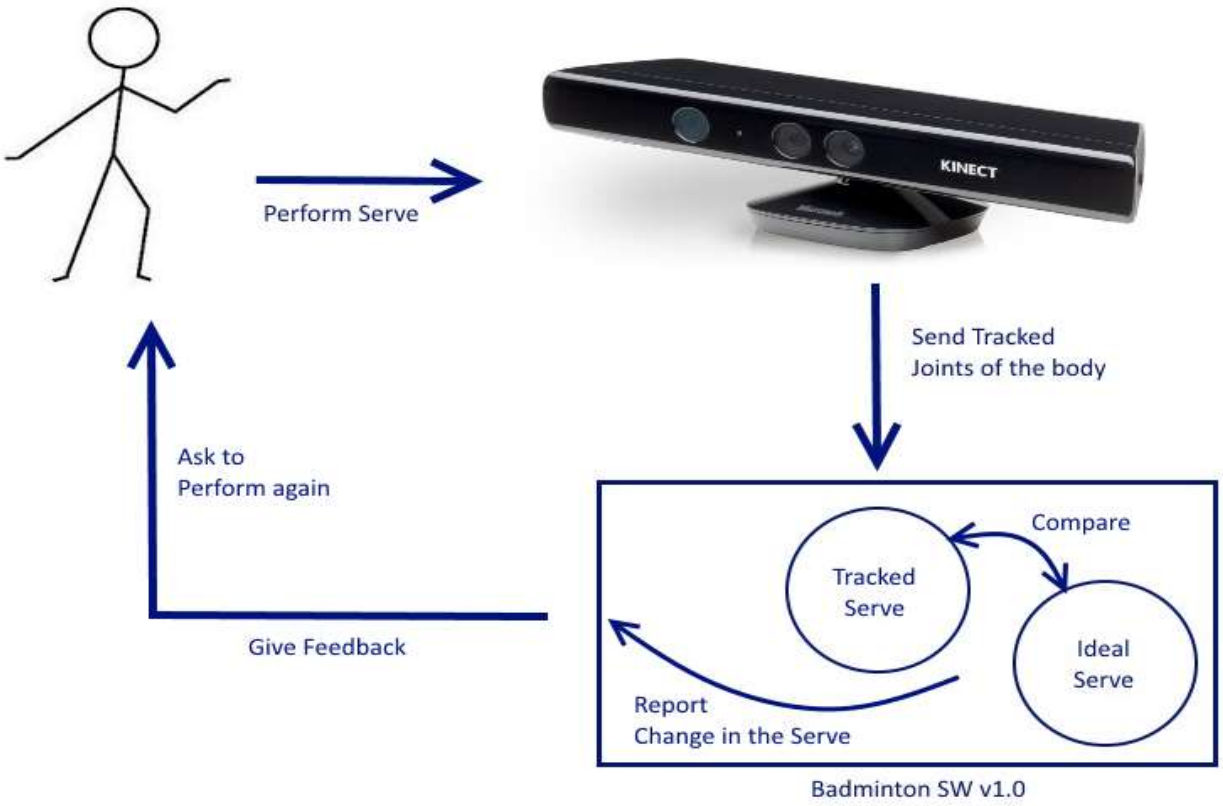
Badminton SW goal is to help in teaching the basic skills of Badminton. It uses Microsoft Kinect sensor to capture trainee's motion and give him feedback about his performance. The requirement of the software are as follows:

ID	**Feature**	**Description**
1	Record a model	A screen to record a 5 seconds motion and save the data to a binary file.
2	Replay a recorded model	A screen to play the recorded model for practicing purposes.
3	Practice screen	A screen to click a button and start recording a practice motion. The screen then gives feedback to the trainee.

By implementing these requirements, we reach the goal of this study which is using virtual reality technology in designing a software program to teach some of the basic skills in badminton. Using Badminton SW v1.0, we can record unlimited number of 5-seconds motions without restriction on the height of the player.

Software Design

Software Architecture



Badminton SW v1.0 is divided into three main modules. The first module is tracking a motion. In this one, we track the model serve and save it to a binary file. We also track trainee's practice motion. The second module is the comparison module; in which we compare trainee's captured motion with the saved model and get differences. The third module is concerned with giving feedback according to the calculated difference and the percentage of error allowed. In the next sections, we are going to investigate each module.

Module 1 - Perform Serve

In this module, we record the human-motion and save it into a file to be processed later. We have tried different approaches to capture the recorded motion and save it. First, we read each skeleton data as joints (coordinates, types, timestamp, position and bone orientation) and save them into a text file. However, this approach was not suitable for later processing. It also takes large processing time and has some accuracy issues (when player's height is different from trainee's height). Then, we tried another approach. We saved the captured motion data as it is into a binary file (not text file) using serialization. There was no need to save each separate skeleton data into a different data structure. It made the accessibility of skeleton data easier for later processing. Here is how we read skeleton data and save it into a list of skeletons:

```
using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
            {
                if (skeletonFrame != null)
                {
                    skeletons = new Skeleton[skeletonFrame.SkeletonArrayLength];
                    skeletonFrame.CopySkeletonDataTo(skeletons);
                }

                Skeleton skeleton = skeletons.Where(s => s.TrackingState ==
SkeletonTrackingState.Tracked).FirstOrDefault();

                if (skeleton != null & recordingStatus)
                {
                    mySkeleton = new Skeleton();
                    timeStamp = skeletonFrame.Timestamp;
                    mySkeleton = skeleton;
                    mySkeletonList.Add(mySkeleton);
                }
```

And here, we serialize the list of skeleton into the binary file:

```
public static void serialize(List<Skeleton> skel, Stream stream)
        {
            try
            {
                BinaryFormatter bFormatter = new BinaryFormatter();
                bFormatter.Serialize(stream, skel);
            }
            catch (Exception ex)
            {
                ex.ToString();
            }
```

```
            finally
            {
                if (stream != null)
                {
                    stream.Close();
                }
            }
        }
```

At this step, we have successfully saved the 5-seconds recoded motion into a binary file which is easier for later processing in the comparison module.

Module 2 - Inside Badminton SW v1.0 Box

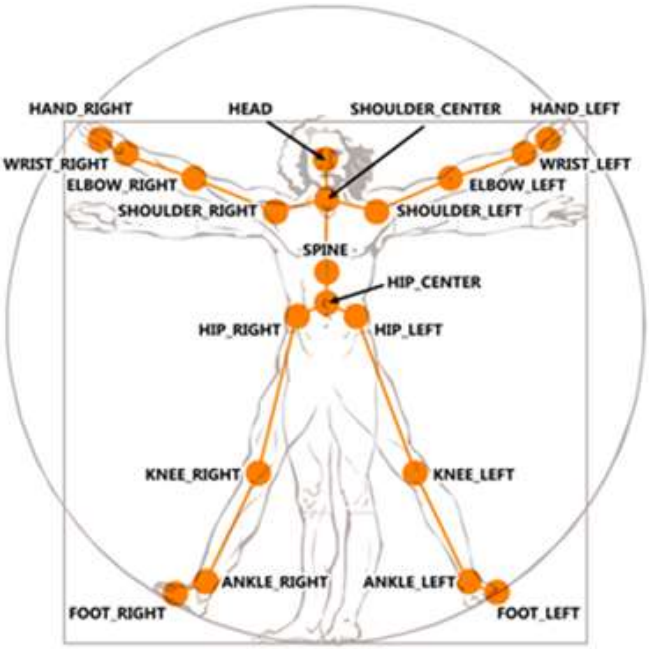
In this module, we should first read the model motion saved in the first module into our skeleton list. Then we pass this list to the comparison algorithm. We read the binary file as follows:

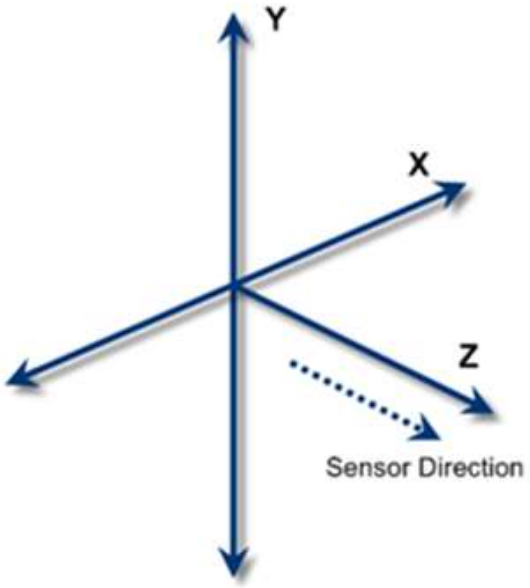
```
public static List<Skeleton> diserialise()
        {
            List<Skeleton> skeletonMotion = null;
            Stream stream = null;
            try
            {
                skeletonMotion = new List<Skeleton>();
                BinaryFormatter bFormatter = new BinaryFormatter();
                stream = File.Open(fileLocation, FileMode.Open);
                skeletonMotion = (List<Skeleton>)bFormatter.Deserialize(stream);
            }

            catch (Exception ex)
            {
                ex.ToString();
            }
            if (stream != null)
            {
                stream.Close();
            }
            return skeletonMotion;
        }
```

After we have successfully read the saved skeleton data, and we get a live motion from the trainee, we need to develop a comparison algorithm to determine how much the tracked motion matches the saved motion. A number of question arises such as what data to compare? How to calculate the difference in the data? How can we achieve this comparison in real time and make it faster?

Before moving on, let's first see what joints the Kinect can understand. Understanding joints data is very crucial in our comparison algorithm. In the following figure, we can see that the Kinect understands 20 joints of the body. For each joint, the Kinect can extract its position, timestamp of the frame and bone orientation.





We have tried two approaches. The first one is to compare the positions of joints in each skeleton frame. However, we found difficulties in doing so. For different players' heights, the comparison can never match between the saved motion and the tracked one. For example, a player's height is 1.60m and recorded his serve as a model serve. Then a trainee of height 1.80m is to perform a practice, the position (x,y,z) of the hand joint in space follows a curve that will be similar to the saved curve but with different positions along the frames. Another problem arises if there is a lag between the start frame of the player and the trainee. That is when the trainee starts playing the serve a number of frames later. The comparison then will never match.

A better approach to compare is to use joints' angles. We calculate joint angles of every frame in the recorded motion and save them to a list of joint angles. How do we do so? From the 20 body joints in which the Kinect recognizes, we can calculate vectors joining these points and hence angles between those vectors with respect to their direction.

To calculate angles, we use 3 joint types to calculate vectors joining them. Then, for each skeleton frame, we calculate angle between these vectors with respect to their directions. First, we normalize the vectors, then calculate dot product and cross product of these normalized vectors. From the dot and cross products we obtain two values from which calculation can be made using atan2 function. After calculating these angles, we save them into list data structures to access them later when we compare. Here is the full code of the method:

```
private double getSegmentAngle(Skeleton skeleton, JointType type0, JointType type1,
JointType type2)
        {
            Microsoft.Xna.Framework.Vector3 crossProduct;
            Microsoft.Xna.Framework.Vector3 joint0Tojoint1;
            Microsoft.Xna.Framework.Vector3 joint1Tojoint2;

            Joint joint0 = skeleton.Joints[type0];
            Joint joint1 = skeleton.Joints[type1];
            Joint joint2 = skeleton.Joints[type2];

            joint0Tojoint1 = new Microsoft.Xna.Framework.Vector3(joint0.Position.X -
joint1.Position.X, joint0.Position.Y - joint1.Position.Y,
                joint0.Position.Z - joint1.Position.Z);
            joint1Tojoint2 = new Microsoft.Xna.Framework.Vector3(joint2.Position.X -
joint1.Position.X, joint2.Position.Y - joint1.Position.Y,
                joint2.Position.Z - joint1.Position.Z);

            joint0Tojoint1.Normalize();
            joint1Tojoint2.Normalize();

            double dotProduct = Microsoft.Xna.Framework.Vector3.Dot(joint0Tojoint1,
joint1Tojoint2);
            crossProduct = Microsoft.Xna.Framework.Vector3.Cross(joint0Tojoint1,
joint1Tojoint2);
            double crossProdLength = crossProduct.Length();
            double angleFormed = Math.Atan2(crossProdLength, dotProduct);
            double angleInDegree = angleFormed * (180 / Math.PI);
            roundedAngle = Math.Round(angleInDegree, 2);

            return roundedAngle;
        }
    }
```

So, how do we compare?

Now, we have calculated the angle throughout the body for each frame and saved them to the list data structure. Then, we need to retrieve these lists and compare them with angles from the live skeleton for each frame. We introduced a tolerance of +/- 20 degrees (acceptable range of error in angle comparison). Any tracked angle that lies between these boundaries from the saved angle is considered to be a successful match. We do so for each frame.

The total number of skeletons in a particular motion is countable. So, we have established the mechanism that if live-skeletons match with at least 90% of the total number of saved skeletons, then we result the motion to be re-captured and matched successfully. We can change this percentage as needed.

Module 3 – Feedback

In addition to the live comparison we make, we give an immediate feedback to the trainee for each angle we calculated after the practice is done. The feedback appears as follows:

```
result.Content = "الصحيح بالشكل بالحركة قمت لقد .. رائع";

                        textBox1.AppendText("\n للمقاطع الكلي العدد: " + liveSkeletons +
"\t\t\tالمئوية النسبة");
                        textBox1.AppendText("\n");
                        textBox1.AppendText("\n يمين وكتف يد: " + rightHandCount +
"\t\t\t\t" + Math.Round(((double)rightHandCount / liveSkeletons) * 100, 2) + "%");
                        textBox1.AppendText("\n يسار وكتف يد: " + leftHandCount + "\t\t\t\t"
+ Math.Round(((double)leftHandCount / liveSkeletons) * 100, 2) + "%");
                        textBox1.AppendText("\n يمين وكتف كوع: " + rightShoulderCenterCount
+ "\t\t\t" + Math.Round(((double)rightShoulderCenterCount / liveSkeletons) * 100, 2) +
"%");
                        textBox1.AppendText("\n يسار وكتف كوع: " + leftShoulderCenterCount +
"\t\t\t" + Math.Round(((double)leftShoulderCenterCount / liveSkeletons) * 100, 2) + "%");
                        textBox1.AppendText("\n الفقري العمود مع الأيمن الكتف: " +
rightShoulderSpineCount + "\t\t" + Math.Round(((double)rightShoulderSpineCount /
liveSkeletons) * 100, 2) + "%");
                        textBox1.AppendText("\n الفكري العمود مع الأيسر الكتف: " +
leftShoulderSpineCount + "\t\t" + Math.Round(((double)leftShoulderSpineCount /
liveSkeletons) * 100, 2) + "%");
                        textBox1.AppendText("\n الفقري العمود مع اليمني اليد: " +
rightWristSpineCount + "\t\t" + Math.Round(((double)rightWristSpineCount / liveSkeletons)
* 100, 2) + "%");
                        textBox1.AppendText("\n الفقري العمود مع اليسرى اليد: " +
leftWristSpineCount + "\t\t" + Math.Round(((double)leftWristSpineCount / liveSkeletons) *
100, 2) + "%");
                        textBox1.AppendText("\n الأيمن والكوع الكف: " + rightHandFingerCount +
"\t\t\t" + Math.Round(((double)rightHandFingerCount / liveSkeletons) * 100, 2) + "%");
                        textBox1.AppendText("\n الأيسر والكوع الكف " + leftHandFingerCount +
"\t\t\t" + Math.Round(((double)leftHandFingerCount / liveSkeletons) * 100, 2) + "%");

                        textBox1.AppendText("\n والوسط الأيمن الكاحل: " + rightLegAngleCount +
"\t\t\t" + Math.Round(((double)rightLegAngleCount / liveSkeletons) * 100, 2) + "%");
                        textBox1.AppendText("\n والوسط الأيسر الكاحل: " + leftLegAngleCount +
"\t\t\t" + Math.Round(((double)leftLegAngleCount / liveSkeletons) * 100, 2) + "%");
                        textBox1.AppendText("\n والوسط اليمني الركبة " + rightHipAngleCount +
"\t\t\t" + Math.Round(((double)rightHipAngleCount / liveSkeletons) * 100, 2) + "%");
                        textBox1.AppendText("\n والوسط اليسرى الركبة " + leftHipAngleCount +
"\t\t\t" + Math.Round(((double)leftHipAngleCount / liveSkeletons) * 100, 2) + "%");
                        textBox1.AppendText("\n الفقري والعمود الأيمن الوسط " +
rightSpineThighAngleCount + "\t\t" + Math.Round(((double)rightSpineThighAngleCount /
liveSkeletons) * 100, 2) + "%");
                        textBox1.AppendText("\n الفقري والعمود الأيسر الوسط " +
leftSpineThighAngleCount + "\t\t" + Math.Round(((double)leftSpineThighAngleCount /
liveSkeletons) * 100, 2) + "%");
```

This feedback helps the trainee improve his performance in specific joints.

Testing

Badminton SW v1.0 implies the three pillars of building the software; stability, accuracy and objectivity.

Stability

Badminton SW v1.0 shows how the implemented algorithm is stable. We have tested it by asking a player to perform a motion from different angles with respect to the Kinect. Since skeleton tracking is done in 3D space (x, y, z coordinates), there should be no problem to how a user stands. We have performed a number of test cases and the results were as expected.

Accuracy

Calculations of joint angles has to be precise and accurate to yield accurate comparison. If angle calculations are inaccurate, then our comparison will be in a doubt. To ensure our angle calculations are accurate, we tested joint angles throughout the body and results were in the accepted range of tolerance (-/+ 10 degrees).

Objectivity

By completing Badminton SW v1.0, we have achieved our goal in proving that we have designed a program to teach some of the basic skills of badminton using virtual reality technology.