JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 26 No. 3, pp. 43-50, March 2021
https://doi.org/10.9708/jksci.2021.26.03.043

A Design and Implementation of Fitness Application Based on Kinect Sensor

Won Joo Lee*

*Professor, Dept. of Computer Science, InHa Technical College, Incheon, Korea

[Abstract]

In this paper, we design and implement KITNESS, a windows application that feeds back the accuracy of fitness motions based on Kinect sensors. The feature of this application is to use Kinect's camera and joint recognition sensor to give feedback to the user to exercise in the correct fitness position. At this time, the distance between the user and the Kinect is measured using Kinect's IR Emitter and IR Depth Sensor, and the joint, which is the user's joint position, and the Skeleton data of each joint are measured. Using this data, a certain distance is calculated for each joint position and posture of the user, and the accuracy of the posture is determined. And it is implemented so that users can check their posture through Kinect's RGB camera. That is, if the user's posture is correct, the skeleton information is displayed as a green line, and if it is not correct, the inaccurate part is displayed as a red line to inform intuitively. Through this application, the user receives feedback on the accuracy of the exercise position, so he can exercise himself in the correct position. This application classifies the exercise area into three areas: neck, waist, and leg, and increases the recognition rate of Kinect by excluding positions that Kinect does not recognize due to overlapping joints in the position of each exercise area. And at the end of the application, the last exercise is shown as an image for 5 seconds to inspire a sense of accomplishment and to continuously exercise.

▸ **Key words**: Kinect Sensor, Color Stream, Depth Stream, Skeleton Stream, Joint, Fitness

[요 약]

본 논문에서는 키넥트 센서를 기반으로 한 휘트니스(Fitness) 동작의 정확성을 피드백 하는 윈도우 애플리케이션 KITNESS를 설계하고 구현한다. 이 애플리케이션의 특징은 키넥트의 카메라와 관절 인식 센서를 활용하여 사용자가 정확한 휘트니스 자세로 운동할 수 있도록 피드백을 주는 것이다. 이때 키넥트의 IR Emitter와 IR Depth Sensor를 이용하여 사용자와 키넥트 간의 거리를 측정하고, 사용자의 관절 위치인 조인트(Joint)와 각 관절의 스켈레톤(Skeleton) 데이터를 측정한다. 이러한 데이터를 이용하여 사용자의 관절 위치와 자세마다 일정 거리를 계산하고 자세의 정확도를 판단한다. 그리고 키넥트의 RGB 카메라를 통해 사용자가 본인의 자세를 확인할 수 있도록 구현한다. 즉, 사용자의 자세가 정확하면 스켈레톤 정보를 초록색 선으로 표시하고, 정확하지 않으면 정확하지 않은 부분을 빨간색 선으로 표시하여 직관적으로 알려준다. 사용자는 이 애플리케이션을 통하여 운동하는 자세의 정확도를 피드백 받기 때문에 혼자서도 정확한 자세로 운동할 수 있다. 이 애플리케이션은 운동 부위를 목, 허리, 다리 세 가지 영역으로 분류하고, 각 운동 부위의 자세에서 관절이 겹쳐서 키넥트가 인식하지 못하는 자세를 제외함으로써 키넥트의 인식률을 높인다. 그리고 애플리케이션 종료 시에는 마지막 운동 모습을 이미지로 5초간 보여줌으로써 성취감을 고취시키고 지속적으로 운동할 수 있도록 구현한다.

▸ **주제어**: Kinect Sensor, Color Stream, Depth Stream, Skeleton Stream, Joint, Fitness

• First Author: Won Joo Lee, Corresponding Author: Won Joo Lee
*Won Joo Lee (wonjoo2@inhatc.ac.kr), Dept. of Computer Science, InHa Technical College
• Received: 2020. 11. 30, Revised: 2020. 12. 28, Accepted: 2020. 12. 28.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

한국건강증진개발원의 조사에 따르면 직장인 61%는 유료 운동 시설 회원권을 등록하고 장기간 이용하지 않는다고 한다. 그 이유는 그림 1과 같이 36%가 “업무 및 일상생활 일정이 불규칙적으로 바뀌어서”라고 응답했고, 30%가 “동기부여 및 의지 상실”이라고 답했다[1].

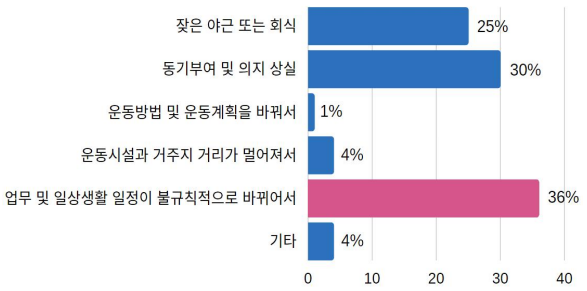


Fig. 1. The reasons why people give up using Paid Fitness Facilities[1]

이러한 환경에서 최근 몇 년 사이에 건강하고 아름다운 신체를 중요시하는 사회적 분위기가 급부상하면서 운동 시설에 가지 않고 집에서 혼자 운동하는 홈 트레이닝(집과 트레이닝의 합성어)이 하나의 트렌드로 자리 잡았다. SNS에서도 홈 트레이닝을 언급하는 게시물의 노출 빈도 수가 크게 증가하고 있다[2].

Fig. 2. The reasons why people do ‘Home Training’[3]

그림 2를 살펴보면 홈 트레이닝을 선호하는 주된 이유는 원하는 시간에 운동할 수 있다는 것이다. 그리고 타인의 시선을 신경 쓰지 않아도 된다는 점과 금전적으로 절약할 수 있다는 점에서도 홈 트레이닝을 선호하고 있다[3]. 홈 트레이닝의 단점은 잘못된 자세로 운동을 지속하게 되면 근육 손상 등의 부상이 발생한다는 것이다. 따라서 최상의 운동 효과를 얻기 위해서는 올바른 자세로 운동하는 것이 중요하다. 이를 위해 보통 사람들은 트레이너를 통해 피드백을 주고 받으며 바른 자세로 운동하지만, 홈 트레이닝에서는 피드백을 받을 수 없다는 단점이 있다. 이러한 홈 트레이닝의 단점을 줄이고, 혼자서도 정확하게 운동하는 것을 도와주는 애플리케이션이 필요하다. 따라서 본 논문에서는 키넥트 센서를 기반으로 한 휘트니스 동작의 정확성을 피드백하는 윈도우 애플리케이션을 설계하고 구현한다.

본 논문은 다음과 같이 구성한다. 2장에서 기존의 유사 휘트니스 애플리케이션을 분석하고, 키넥트의 특징에 대하여 설명한다. 3장과 4장에서는 각각 키넥트 센서를 기반으로 휘트니스 동작의 정확성을 피드백 하는 윈도우 애플리케이션을 설계하고 구현한다. 마지막으로 5장에서 결론을 맺는다.

II. Preliminaries

1. Analysis of Existing Fitness Applications

기존 자세 인식 운동 프로그램들을 분석한 결과는 표 1과 같다.

Table 1. Existing Fitness program

Classification	Just Dance	Fitness Boxing	Nike+ Kinect Training
Type	Motion recognition dance game	Motion recognition fitness game	PT program
Device	Multiple	Will	Kinect
Controller	O	O	X
Exercise posture	A pose that is difficult to follow quickly	Boxing posture	aggressive posture
Real-time posture feedback	Text	Text	Audio
Exercise guide method	Character	Character	Character
How to induce continuous use	showing the game score at the end	Shows the days you exercised on the calendar	sharing exercise achievement rates with friends
How to see the user	X	X	Character

분석 결과 휘트니스 애플리케이션은 3가지 기능이 필요하다는 것을 알 수 있다. 첫째, 캐릭터를 활용해 자세를 실시간으로 보여주어야 한다. 이는 화면에서 캐릭터가 취하는 자세를 보며 정확히 따라 할 수 있도록 도와준다. 둘째, 즉각적인 피드백이다. 실제 PT처럼 느껴지기 위해서는 옆에 트레이너가 있는 것처럼 자세가 정확한지, 잘하고 있는

지를 알려 주어야 한다. 셋째, 사용자의 성취감을 고취시켜야 한다. 유흥성이 없다면 사용자의 꾸준한 사용이 어렵다. 따라서 운동 성취율을 친구들에게 공개하여 경쟁심리 자극, 며칠 동안 운동을 지속했는지를 알려주는 등 여러 방법을 통해 사용자가 꾸준히 운동할 수 있도록 도와준다.

이러한 분석 결과에 따라 본 논문에서 개발하는 애플리케이션은 사용자와 캐릭터를 동시에 보여주어 자세를 따라 할 수 있는 화면을 구성한다. 그리고 자세 스텝마다 정확도 검사 후 “잘하고 있어” 혹은 “좀만 더 노력하자”라는 피드백과 사용자 자세의 틀린 부위에는 빨간 선을 그어 직관적으로 알림으로써 실시간으로 피드백 받을 수 있게 한다. 또한, 사용자의 성취감을 자극하기 위해 프로그램을 종료할 때마다 사용자가 최근에 정확하게 운동했던 모습을 보여주며 꾸준히 운동할 수 있도록 설계한다.

2. Kinect Sensor

키넥트는 “Kinetic”와 “Connect”의 합성어로 사람의 동작을 인지하여 컴퓨터 시스템에 연결하는 장치를 의미한다. 키넥트에서 RGB 카메라는 렌즈를 통해서 본 이미지를 전송하고, Depth 센서인 IR Emitter와 IR Depth Sensor는 적외선을 이용해 센서와 물체의 거리 데이터를 생성하여 컨트롤러 없이 사용자를 인식한다. 그리고 Tilt Motor는 키넥트 센서를 상하로 움직이는 기능을 제공한다[4].

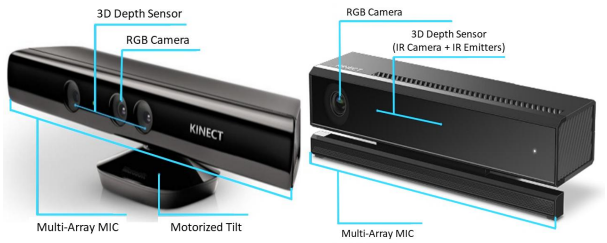


Fig. 3. Kinect Structure

키넥트는 콘솔 게임기로 Xbox 360 및 Xbox One용 주변기기, PC에서 사용하는 개발용으로 나눌 수 있지만, 크게 버전에 따라 그림 3과 같이 v1과 v2로 구분한다. 키넥트 v1과 v2는 같은 센서로 구성되어 있지만, Depth 정보를 얻어오는 방식과 센서 위치에 따른 외관 차이가 있다. 키넥트 v1에서는 Depth 정보를 패턴 레이저를 피사체에 방사하여 나타나는 패턴의 왜곡을 분석, 즉 SL(Strctured Light) 방식을 이용하여 센싱한다. 반면 키넥트 v2는 송신부에서 레이저를 방사하여 피사체에 반사되어 수신부까지 돌아오는 시간을 계산하여 거리를 측정하고, Depth 정보를 얻는 ToF(Time of Flight) 방식을 사용하는 차이점이 있다. 키넥트 v1과 v2의 특징은 표 2와 같다.

Table 2. Specification of Kinect v1, v2 sensor[5]

Classification		Kinect v1	Kinect v2
Color	resolution	640 x 480	1980 x 1080
	fps	30 fps	30 fps * 3
Depth	resolution	320 x 240	512 x 424
	fps	30 fps	30 fps
Player		6	6
Skeleton		2	6
Joint		20	25
Range of Depth		0.8 ~ 4.0m (Near Mode 0.4m ~)	0.5 ~ 0.8m
Range of Detection		0.8 ~ 4.0m (Near Mode 0.4 ~ 3.0m)	0.5 ~ 4.5m
Angle	Horizontal	57 Degree	70 Degree
	Vertical	43 Degree	60 Degree
Tilt Motor		O	X

3. Kinect Sensor Function

키넥트는 총 3가지 센싱 함수를 가진다. Color Stream은 Color Image Stream 클래스의 객체로 웹 카메라처럼 컬러 영상을 출력하는 기능을 제공한다. Depth Stream은 Depth Image Stream 클래스의 객체로 Kinect v1의 경우 Depth Stream을 SL 방식으로, v2의 경우 ToF 방식으로 물체와 센서 간 거리 데이터를 생성한다. 이 데이터는 총 16bit로 상위 13bit는 픽셀의 depth Value, 하위 3bit는 player Index를 가진다. 또한, 대부분의 키넥트는 인식하기 좋은 범위가 0.8 ~ 4m 이지만, Kinect for windows는 근접 모드를 지원하므로 적정 범위는 0.4 ~ 3m이다. Skeleton Stream에서 인체의 관절 위치를 조인트(Joint), 각 관절의 골격 정보를 스켈레톤(Skeleton)이라 한다. 키넥트는 Joint Type를 통해 관절 정보를 알 수 있고, 그림 4와 같이 총 20개 관절을 인식할 수 있다.

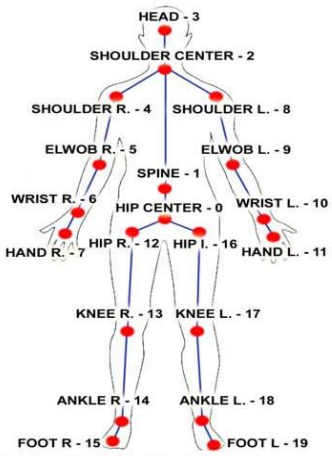


Fig. 4. Joint ID Map recognized by Kinect

III. The Design of a KITNESS

1. 운동 자세 선정

본 논문에서는 키넥트 센서를 기반으로 한 휘트니스 동작의 정확성을 피드백 하는 윈도우 애플리케이션인 KITNESS를 구현한다. KITNESS는 Kinect와 Fitness의 합성어로, 키넥트를 활용하여 정확한 휘트니스 자세로 운동할 수 있도록 피드백 해주는 윈도우 애플리케이션이다. 키넥트의 스켈레톤 인식 기능의 정확도가 어떤 자세에 따라 차이가 있는지 알아보기 위해 실험을 진행한다. 사용자가 다양한 자세를 취할 때 사용자의 스켈레톤을 인식하는 키넥트의 정확도를 알아보기 위해 실험을 진행한다. 그림 5의 A 영역과 같이 사용자 관절들이 겹치면 B 영역과 같이 스켈레톤 위치를 정확하게 인식하지 못하는 단점이 있다. 이러한 스켈레톤의 단점을 줄이기 위해서는 키넥트가 인식하는 사용자 신체가 겹치지 않는 운동 자세를 선택해야 한다.

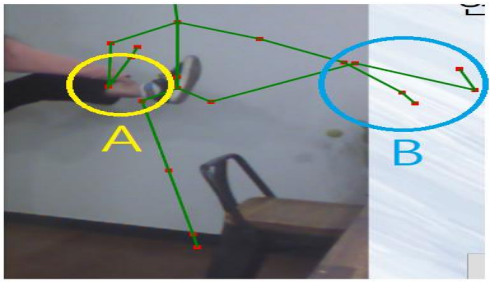


Fig. 5. Incorrect recognition results (B) of overlapped joints (A)

KITNESS의 운동 자세는 크게 목, 허리, 다리로 분류한다. 목은 그림 6의 (a)와 같이 머리 기울여 늘이기, 허리는 그림 6의 (b)와 같이 옆구리 늘리기, '다리는 그림 6의 (c)와 같이 한쪽 무릎 옆으로 구부리기로 분류한다.

(a) Neck (b) Waist (c) Leg

Fig. 6. exercises

2. 애플리케이션 개발 환경

본 논문에서 개발하는 애플리케이션은 확장성이 넓은 윈도우즈 플랫폼 기반의 애플리케이션으로 개발하기 위해 Visual Studio 2017에서 WPF으로 구현한다. 윈도우즈와 키넥트 연동을 위해 Kinect for Windows SDK를 사용한다. 애플리케이션 개발 환경은 표 3과 같다.

Table 3. Development Environment

Classification	Environment
HardWare	• Kinect for windows (Version 1)
SoftWare	• Visual Studio 2017 • Kinect SDK 1.8

IV. The Implementation of a KITNESS

본 논문에서 구현하는 KITNESS 윈도우 애플리케이션은 키넥트의 뎁스(Depth) 센서에서 센싱한 데이터와 사용자의 관절 정보인 조인트(Joint)와 스켈레톤(Skeleton) 정보를 활용하여 휘트니스 자세의 정확도를 판단한다. KITNESS 구현에 필요한 클래스는 표 4와 같다.

그림 7은 운동 자세 선택 화면으로 주요 기능은 Part Selection Class에서 3가지 이벤트로 구성된다.

Table 4. Function of Class

Class	Function and Method
PartSelection	운동 자세 선택 • PartSelection() • Btpart_Click() • part_mouseover() • part_mouseleave()
Kinect_View	키넥트 인식 이미지를 실시간으로 출력 • Kinect_View() • dispatcherTimer_Tick() • InitializeNui() • nui_ColorFrameReady() • nui_AllFramesReady() • SavePng()
Kinect_Point	키넥트 인식 관절 정보, 자세 정보 관리 • point_set() • GoPosition_Check()
Default_ready	준비 상태, 키넥트 연결 확인 • Default_ready() • dispatcherTimer_Tick() • Btstart_Click() • Start_Check()
Default_start	운동 실행 순서 관리 • Default_start() • dispatcherTimer_Tick() • check_return_val_right() • character_img_change() • restart()
Neck	목 운동 검사 • Check_right() • Check_left()
Spine	허리 운동 검사 • Check_right() • Check_left()
Leg	다리 운동 검사 • Check_right() • Check_left()
Exit	종료 시 가장 최근에 운동한 사진 보이기 • Exit() • set_img_name()

(a) Default

(b) Play the preview Gif

Fig. 7. Exercise posture selection screen

(a) if the Kinect is not connected

(b) if the Kinect is connected

Fig. 8. Preparation Screen

첫째, 마우스 오버 이벤트다. 마우스 오버 이벤트 핸들러는 part_mouse over 메서드로 운동 자세 버튼에 마우스를 올리면 그림 7의 (b)와 같이 움직이는 이미지로 자세를 미리 보여준다. 운동 부위는 크게 목, 허리, 다리로, 각 3개씩 총 9가지 운동을 미리 보고 진행 할 수 있다. 둘째, 마우스 리브 이벤트다. 마우스 리브 이벤트 핸들러는 part_mouse leave라는 메서드로 자세 버튼에서 마우스가 멀어지면 미리 보기 이미지를 자동으로 종료시킨다. 셋째, 마우스 클릭 이벤트다. 운동 자세 버튼을 클릭하면 해당 운동 준비 화면을 출력한다.

운동 자세 미리 보기로 활용한 Gif 파일은 표 5와 같이 'WpfAnimated Gif' 라이브러리를 활용하여 재생한다.

Table 5. WpfAnimated Gif Code

```
1: var uriS = new Uri(@"image path/"+((Button)sender)
                .Name.ToString()+".gif", UriKind.Absolute);
2: var image = new System.Windows.Media.Imaging
                .BitmapImage();
3: image.BeginInit();
4: image.UriSource = uriS;
5: image.EndInit();
6: WpfAnimatedGif.ImageBehavior.SetAnimatedSource
   (img_gif, image);
7: img_gif.Visibility = Visibility.Visible;
```

그림 8은 운동 준비 화면으로 주요 기능은 키넥트 연결 확인 후 테스트하는 기능이다. 그림 8의 (a)와 같이 키넥트가 연결되지 않은 경우, 시작하기 버튼을 비활성화하고 키넥트 연결을 요청한다. 그림 8의 (b)와 같이 키넥트가 연결된 경우, 시작하기 버튼이 활성화되고, 카메라 화면을 미리 확인할 수 있다. 또한, 키넥트가 사용자의 관절을 인식하는지 테스트하거나, 왼쪽에 캐릭터가 운동하는 모습을 보며 미리 연습할 수 있다.

키넥트 연결을 확인하는 기능은 Default ready Class에서 Dispatcher Timer를 활용한 dispatcherTimer_Tick() 메서드로 구현한다. 이 메서드는 kinect_c라는 Boolean 형태인 전역변수 값의 true 여부를 초 단위로 비교한다. 이 전역변수는 표 6과 같이 키넥트를 설정하는 Kinect View Class에서 키넥트가 처음 연결된 경우 초기화와 설정하는 InitializeNui() 메서드를 수행했을 때만 true로 변한다. 따라서 이 전역변수 값이 true라면 키넥트를 사용할 수 있는 경우이므로 그림 8의 (b)와 같이 키넥트의 Color stream을 통해 컬러 영상 미리 보기와 사용자 인식 테스트를 진행할 수 있다.

Table 6. InitializeNui Method Code

```
1: nui = KinectSensor.KinectSensors[0];
2: nui.ColorStream.Enable();
3: nui.ColorFrameReady += new EventHandler<Color
   ImageFrameReadyEventArgs>(nui_ColorFrameReady);
4: nui.DepthStream.Enable();
5: nui.SkeletonStream.Enable();
6: nui.AllFramesReady += new EventHandler<AllFrames
                  ReadyEventArgs>(nui_AllFramesReady);
7: nui.Start();
8: Kinect_Point.kinect_c = true;
```

휘트니스 운동을 시작하기 위해서는 시작하기 버튼을 클릭하는 방법과 그림 9와 같이 넘기기 자세를 취하면 자세 인식 후 자동으로 넘겨주는 방법이 있다. 이 자세는 페이지를 넘기기도 하지만, 운동 진행이 끝난 후 같은 자세를 취하면 한 세트를 추가할 수 있다. 그림 9의 넘기기 자세는 Kinect Point Class의 GoPosition_Check() 메서드로 검사한다. 표 7과 같이 팔의 중심인 엘보우(elbow)가 상체 중심인 척추(spine)보다 높고 오른쪽 팔 관절들의 y 위치를 비교하여 일자로 폈는지 판단한다.

휘트니스 운동 화면의 오른쪽에서는 현재 몇 세트, 몇 회를 하고 있는지 알 수 있다. 운동은 한 세트당 10회를 진행하며 그림 10과 같이 캐릭터로 운동 자세 순서를 알려주며 진행한다. 운동을 진행하면 Joint Type을 통해 사용자의 관절 위치들을 가져와 자세마다 일정 거리를 계산하고 자세의 정확도를 판단하고 피드백을 준다. 운동 자세와 체형에 따라 같은 자세여도 다른 계산 값이 나올 수 있으므로 모든 부위가 기준 값에서 설정한 여유 범위 내라면 그림 10의 (a)와 같이 모든 부위가 초록색으로 표시하여 올바름을 직관적으로 알 수 있다. 반면 기준 값에서 여유 범위를 벗어났다면 틀린 자세로 판단하고, 틀린 부위를 빨간색으로 틀렸음을 그림 10의 (b)와 같이 명시하고 올바른 자세를 유도한다.

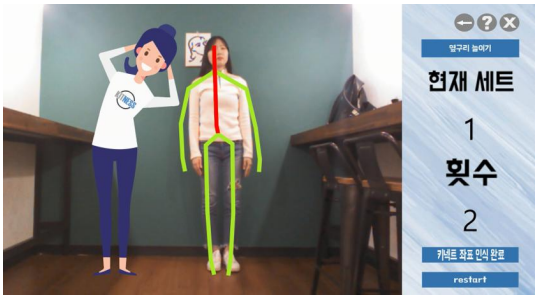
Fig. 9. The flipping position shown in the tutorial

Table 7. GoPosition_Check Method Code

```
1: test = points[(int)JointType.ShoulderRight].X
                  < points[(int)JointType.ElbowRight].X
           && points[(int)JointType.ElbowRight].X
                  < points[(int)JointType.WristRight].X
           ? true : false;
2: if(test) {
3:    if (points[(int)JointType.Spine].Y >
                  points[(int)JointType.ElbowRight].Y){

4:    test = points[(int)JointType.ElbowRight].Y + 20
                  >= points[(int)JointType.ShoulderRight].Y
           && points[(int)JointType.ElbowRight].Y - 20
                  <= points[(int)JointType.ShoulderRight].Y
           ? (points[(int)JointType.ElbowRight].Y + 20
                  >= points[(int)JointType.WristRight].Y
           && points[(int)JointType.ElbowRight].Y - 20
                  <= points[(int)JointType.WristRight].Y
           ? true : false) : false;
5:    if (test) {
6:       if (points[(int)JointType.Spine].Y
                  < points[(int)JointType.ElbowLeft].Y) {
7:          return true;
8:       } return false;
9:    } return false;
10: } return false;
11: return false;
```

(a) posture is correct



(b) Incorrect posture

Fig. 10. Exercise Feedback Screen

자세 검사를 위해 운동 부위에 따라 Neck, Spine, Leg Class가 실행된다. 이 Class들은 Check_left(), Check_right() 메서드를 통해 사용자의 왼쪽과 오른쪽을 나눠 각

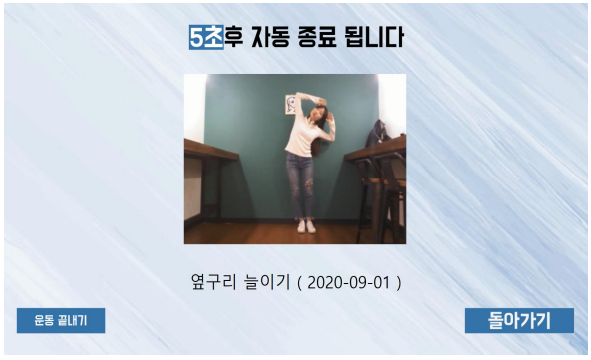
각 검사한다. 예시로 가져온 표 8에 적힌 코드같이 검사 결괏값을 숫자로 구분하여 틀린 부위가 정확히 어디인지 알 수 있게 한다. 결괏값이 10이라면 올바른 자세, 10 이하 숫자라면 각 숫자에 해당하는 부위가 틀렸음을 알 수 있다. 검사 결과를 사용자가 알 수 있도록 그림 10의 (b)처럼 틀린 부위는 빨간 선으로 그린다. 그림 10에 나온 선들은 Skeleton Stream에서 Joint Type을 통해 알아낸 사용자 관절 위치를 전부 이어서 표현한다.

사용자 위에 선으로 해당 부위를 명시하는 것 외에도 그림 10의 (a)와 같이 왼쪽 하단에 강아지 스티커와 문구를 통해 피드백을 줄 수 있다. 해당 자세가 정확하다면 강아지 스티커와 “잘하고 있어”라는 문구를 출력하고, 자세가 틀렸다면 강아지 스티커와 “좀만 더 힘내자”라는 문구를 출력하는 피드백 받을 수 있다.

운동을 진행하기 전, 진행 중, 진행 완료 후 운동 선택 화면으로 돌아가거나 애플리케이션을 종료할 수 있다. 애플리케이션을 종료 시에는 그림 11의 (a)와 같이 Dispatcher Timer를 통해 사용자가 운동하던 모습을 5초간 보여준다. 종료되기 전까지 사용자 자신이 운동하던 모습을 보며 성취감이 고취되어 다시 운동하고 싶다면, 돌아가기 버튼을 통해 운동 선택 화면으로 돌아갈 수 있다. 여기서 나오는 이미지의 1순위는 종료 직전에 실행한 운동 이미지이고 없다면 가장 최근 사진을 가져온다. 단, 운동을 한 번도 하지 않아 이미지가 없다면 그림 11의 (b)와 같이 운동 진행을 권유한다.

Table 8. Position Check Method Example Code

```
1: if (Kinect_Point.points[(int)JointType.ElbowRight].X
       > Kinect_Point.points[(int)JointType.WristRight].X) {
2:   if (Kinect_Point.points[(int)JointType.Head].Y
         >= Kinect_Point.points[(int)JointType.HandRight].Y
         &&Kinect_Point.points[(int)JointType.ElbowRight].Y
         > Kinect_Point.points[(int)JointType.WristRight].Y) {
3:     if (Kinect_Point.points[(int)JointType.ShoulderLeft].Y
           < Kinect_Point.points[(int)JointType.ElbowLeft].Y
           &&Kinect_Point.points[(int)JointType.WristLeft].Y
           < Kinect_Point.points[(int)JointType.ElbowLeft].Y)
4:       { return 10; }
5:   } return 1;
6: } return 2;
7: return 2;
```



(a) Default

(b) If you haven't exercised once

Fig. 11. Program end screen

이미지는 Kinect View Class의 SavePng() 메서드에서 저장된다. 운동 진행 중 정확하게 자세를 취한 경우, 마지막 횟수 직전까지 정확하게 자세를 취한 적이 없으면 마지막으로 운동할 때 메서드 실행을 요청한다. 메서드가 실행되면 키넥트가 출력하는 컬러 영상에서 실행을 요청받은 당시 프레임을 저장한다. 또한 종료 화면에서 이미지를 출력할 때 자세명과 운동했던 날짜를 알 수 있도록 당시 날짜와 운동 자세 이름이 합쳐진 이미지 파일명으로 저장한다.

V. Conclusions

본 논문에서는 키넥트 센서를 기반으로 한 휘트니스 동작의 정확성을 피드백하는 KITNESS 윈도우 애플리케이션을 설계하고 구현하였다. 이 애플리케이션은 키넥트 센서에서 센싱한 조인트와 스켈레톤 정보를 활용하여 정확한 사용자의 휘트니스 자세를 제공한다. 또한, 카메라로 현재 모습을 촬영하여 사용자 관절의 위치를 이어주는 선을 화면에 출력함으로써 사용자의 자세를 정확하게 알 수 있다. 이때 정확하지 않은 자세는 빨간색, 바른 자세는 초록색으로 표현하여 직관적으로 알 수 있도록 하였다. 그리고 정

확한 자세를 하면 사용자의 모습 위에 잘하고 있다고 문구와 스티커로 시각적인 피드백을 준다. 정확하지 않은 자세를 하면 좀 더 분발하자는 문구와 스티커로 시각적인 피드백을 준다. 마지막으로 애플리케이션 종료 시에는 마지막 운동 모습을 이미지로 5초간 보여줌으로써 성취감을 고취시키고, 지속적으로 운동할 수 있도록 구현하였다.

ACKNOWLEDGEMENT

This work was supported by INHA TECHNICAL COLLEGE Research Grant in 2020.

REFERENCES

[1] Korea Health Promotion Institute, Press release of survey results on physical activity of office workers, https://www.khealth.or.kr/board/view?menuId=MENU00907&linkId=501805.

[2] Nielsen, A healthy habit of taking care of my body, home training, http://www.koreanclick.com/insights/newsletter_view.html ?&id=464.

[3] TrendMonitor, 2018 U&A survey on exercise experience and home training in daily life, https://www.trendmonitor.co.kr/tmweb/trend/allTrend/detail.do?bIdx=1677&trendType=CKOREA

[4] Microsoft, Kinect for Windows Sensor Components and Specifications, https://docs.microsoft.com/en-us/previous-versions/windows/kinect-1.8/jj131033(v=ieb.10).

[5] Leepooiye, Differences between Kinect with Kinect 2.0, https://myleepy.wordpress.com/2018/07/16/week-2-kniect/.

[6] Suharjito, Suharjito & Anderson, Ricky & Wiryana, Fanny & Ariesta, Meita & Kusuma Negara, I Gede Putra, “Sign Language Recognition Application Systems for Deaf-Mute People: A Review Based on Input-Process-Output,” Procedia Computer Science. Vol.116. pp. 441-448, Oct. 2017. DOI:10.1016/j.procs.2017.10.028

[7] Microsoft, Microsoft.Kinect Namespace, https://docs.microsoft.com/en-us/previous-versions/windows/kinect-1.8/hh855419(v=ieb.10).

[8] J. K. Ko, "Kinect Programing," Korea Electronics Association, 2012.

[9] Yun, Hye-Jeong, Kim, Kwang-Il, Lee, Jeong-Hun, and Lee, Hae-Yeoun, “Development of Experience Dance Game using Kinect Motion Capture,” KIPS transactions on software and data engineering, Vol. 3, No. 1, pp. 49-56, Jan. 2014. DOI:10.3745/KTSDE.2014.3.1.49

[10] Park, Kyoung Shin, “Development of Kinect-Based Pose Recognition Model for Exercise Game,” KIPS transactions on computer and communication systems, Vol. 5, No. 10, pp. 303-310, Oct. 2016. DOI:10.3745/KTCCS.2016.5.10.303

[11] Kim, ChangGeol and Song, Byung-Seop, “Development of Home Training System with Self-Controlled Feedback for Stroke Patients,” Journal of the Korea Industrial Information Systems Research, Vol. 18, No. 1, pp. 37-45, Feb. 2013. DOI:10.9723/JKSIIS.2013.18.1.037

Authors

Won Joo Lee received the B.S., M.S. and Ph.D. degrees in Computer Science and Engineering from Hanyang University, Korea, in 1989, 1991 and 2004, respectively. Dr. Lee joined the faculty of the Department of Computer Science at Inha Technical College, Incheon, Korea, in 2008, where he has served as the Director of the Department of Computer Science. He is currently a Professor in the Department of Computer Science, Inha Technical College. He has also served as the Vice-president of The Korean Society of Computer Information. He is interested in parallel computing, internet and mobile computing, and cloud computing, data science, artificial intelligence.