**THE REPUBLIC OF TURKEY**
**BAHCESEHIR UNIVERSITY**

# A FRAMEWORK FOR DEVELOPING ONLINE MULTIPLE KINECT INTERACTIONS

**Master of Science Thesis**

**MUHAMMED EMRE AKKOYUN**

**ISTANBUL, 2014**

**THE REPUBLIC OF TURKEY**
**BAHCESEHIR UNIVERSITY**

**THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**
**COMPUTER ENGINEERING**

# A FRAMEWORK FOR DEVELOPING ONLINE MULTIPLE KINECT INTERACTIONS

**Master of Science Thesis**

**MUHAMMED EMRE AKKOYUN**

**Supervisor: Asst. Prof. Dr. Övgü ÖZTÜRK**

**ISTANBUL, 2014**

**THE REPUBLIC OF TURKEY**
**BAHCESEHIR UNIVERSITY**

**GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**
**COMPUTER ENGINEERING**

Name of the thesis: A Framework for Developing Online Multiple Kinect Interactions
Name/Last Name of the Student: M.Emre Akkoyun
Date of the Defense of Thesis: 16.01.2014

The thesis has been approved by the Graduate School of Natural and Applied Science.

Prof. Dr. Tunç BOZBURA
Graduate School Director
Signature

I certify that this thesis meets all the requirements as a thesis for the degree of Master of Arts.

Assist. Prof. Dr. Tarkan AYDIN
Program Coordinator
Signature

This is to certify that we have read this thesis and we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Arts.

| Examining Comittee Members | Signature |
|---|---|
| Thesis Supervisor<br>Assist. Prof. Dr. Övgü ÖZTÜRK | ---------- |
| Member<br>Assist. Prof. Dr. Tarkan AYDIN | ---------- |
| Member<br>Assoc. Prof. Dr. Alper TUNGA | ---------- |

## ACKNOWLEDGMENTS

I would never have been able to finish my master thesis without the guidance of supervisor Asst. Prof. Dr. Övgü ÖZTÜRK, help from friends, and support from my family also my manager at Turkcell.

I would like to express my deepest gratitude to my advisor, Asst. Prof. Dr. Övgü ÖZTÜRK, for her excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research.

I would like to thank Avedis Boyacı, who as a good manager, was always willing to help and give his best suggestions.

I would also like to thank my parents,my sister Dr.Esma Akkoyun Bilgi and my nephew Bugra. They were always supporting me and encouraging me with their best wishes. I also want to thank my relatives to show patience to me for not visiting them. Finally, I would like to thank my nephew, 1 year old Bugra Bilgi. He was always there to increase my life energy and help me to get ride of from thesis stress.

**Muhammed Emre Akkoyun**

**19 December, 2013**

# ÖZET

A FRAMEWORK FOR DEVELOPING ONLINE MULTIPLE KINECT INTERACTIONS

M.Emre Akkoyun

Bilgisayar Mühendisliği

Tez Danışmanı: Yrd. Doç. Dr. Övgü Öztürk

Aralık 2013, 56 Sayfa

Bu çalışma Microsoft firmasının ürettiği kinect kameralarının ürettiği verilerin internet üzerinden işlenmesiyle ilgilenmektedir. Kinect kameralara internet üzerinden direk olarak erişmek bügünün teknoloji dünyası için popüler konuların başında gelmektedir. Çok sayıda yazılım mühendisi ve araştırma firmaları bu sıcak konu üzerinde çalışmaktadırlar ve kullanıcılara bununla ilgili çözümler sunmaktadırlar. Ancak bu çözümlerin avantajları ve dezavantajları bulunmaktadır. Bu dezavantajların başında kinect verisi internet üzerinden işlenirken kullanılan yoğun ağ bant genişliği problemidir, çünkü kinect kameraların ürettiği veriler direk olarak internet ortamı üzerinde işlenirken ağ üzerinde ciddi miktarda bir yük oluşturmaktadır. Bu nedenle kinect kameralar için yazılım geliştiren mühendislerin bunları dikkate alması gerekmektedir ve üretilen veriler internet ortamında işlenmeden önce gereksiz veriler ayıklanıp temizlenmelidir. Bu uygulamalar ayrıca kinect kamera verilerine istemci tarafında çalışan javascript kodları üzerinden eriştikleri için bu uygulamalar sadece lokal bilgisayarlarda çalışabilecek bir yaklaşım sergilemektedir ve birden çok kinect kamera verisinin aynı internet tarayıcısında görüntülenmesi konusu imkansız hale gelmektedir. Bu çalışmada uygulama geliştirmede karşılaşılan bu problemlere çözüm bulunmuştur. Özetle; ağ sistemlerindeki bant genişliğinin yoğun kullanılmasından kaynaklanan problemleri çözmek için kinect verileri internet ortamı üzerinde işlenmeden önce gereksiz veriler ayıklanmıştır ve uygulamalar için gerekli veriler ayıklandıktan sonra JSON formatına dönüştürülerek ağ bant genişliğinin mümkün olduğunca az miktarda kullanılması sağlanmıştır. Birden çok kinect kameradan alınan verilerin aynı anda senkron bir şekilde gösterebilmek için ise uygulama sunucu tarafı ve istemci tarafı olmak üzere iki alt program olarak yazılmıştır. Bu uygulama geliştirme yaklaşımı sayesinde birden çok kinect kamera verisi aynı internet tarayıcısında datalar ağ üzerinden ayrı ayrı işlenerek görüntülenebilecektir. Ayrıca bu geliştirme yaklaşımı sayesinde farklı firmaların ürettiği iskelet takip eden cihazlar için de sunucu uygulama tarafında küçük değişiklikler yaparak farklı cihazlar için de uygulama geliştirme altyapısı sağlanmış olur.

**Anahtar Kelimeler**: Kinect Kamera, Görüntü Derinliği Takibi, İskelet İzleme, İnternet Veri Aktarımı

## ABSTRACT

A FRAMEWORK FOR DEVELOPING ONLINE MULTIPLE KINECT INTERACTIONS

M.Emre Akkoyun

Computer Engineering

Thesis Supervisor: Assistant Prof. Dr. Övgü Öztürk

December 2013, 56 Pages

This study deals with the Microsoft kinect camera data streaming over internet. Accessing a kinect camera over internet is so popular nowadays. Many developer and research company are working on this hot topic and find several solutions. Most of this applications have some avantages and disadvantages. The main disadvantage of this applications is consuming network bandwidth heavily when transferring kinect data over internet. Because transferring whole of kinect data binds heavily mass of data to the network, so the developers of kinect applications should consider about this topic and should mine and comb out the unnecessary kinect data. This applications also directly access the kinect device from javascript and this approach only works for local applications and not support multiple kinecting display on one browser. There is no best case solutions for this kind of problems. In this study solutions have been found for this problems and disadvantages. Briefly ; for network bandwidth problems kinect data has been mined and unnnecesarry kinect data combed out and the necessary data converted in json format before streaming over network, and for multiple kinect support; application divided into two parts which are server side application and client side application and with the help of this approach some performance problems cleared and network bandwith problem also cleared before network streaming and also with this approach more than one kinect cameras data could streamed over network and displayed on a single web page. On the other hand; with the approach which is used at this study can be implemented for other depth streaming devices with just making modifications on server side application.



**Keywords**: Kinect Camera, Depth Streaming, Skeleton Tracking, Datastreaming Over Internet.

# TABLE OF CONTENTS

## TABLES

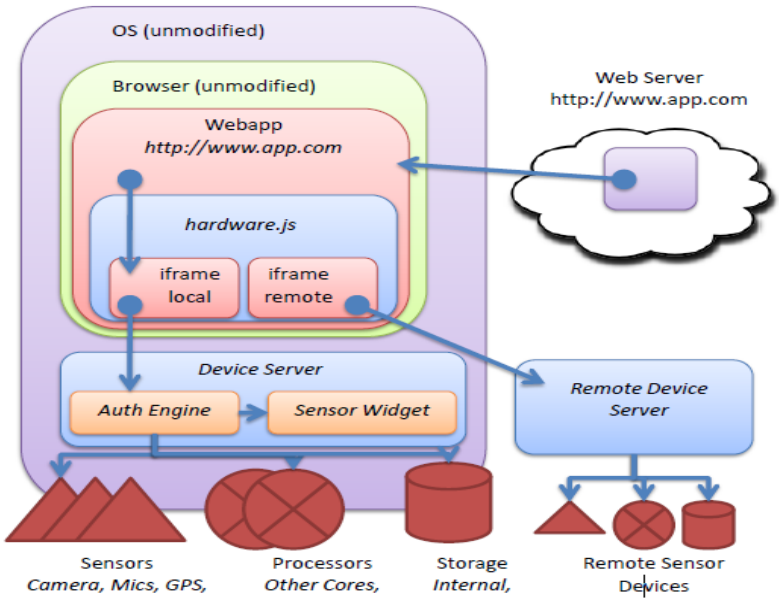
## FIGURES

## CODES

# 1 INTRODUCTION

The rapid evoluation in technology (web technologies and hardware technologies) gives huge opportunities for researchers or companies to create hardware based web applications. So, accesing a hardware device such as, RGB Cameras, depth Cameras from a web site is so popular nowadays for this researchers and companies. The time when embedded devices and factory-floor machines had only minimal interaction with humans is gone. Today, the ability to access a device from anywhere is expected. This is a significant challenge when we are heavily constrained by hardware size and costs. However, with commonly available software components and a little knowledge, we can incorporate a "universal" access interface for data gathering/surveillance that maintains a small footprint and that is secure [1]. With this approach many companies and researchers creates applications and frameworks for accessing hardware devices from web technologies, for example; Microsoft announced a framework which is called Gibraltar for accessing hardware devices from a web application. Gibraltar's architecture is shown in Figure 1.1:

Figure 1.1 Microsoft Gibraltar framework architecture

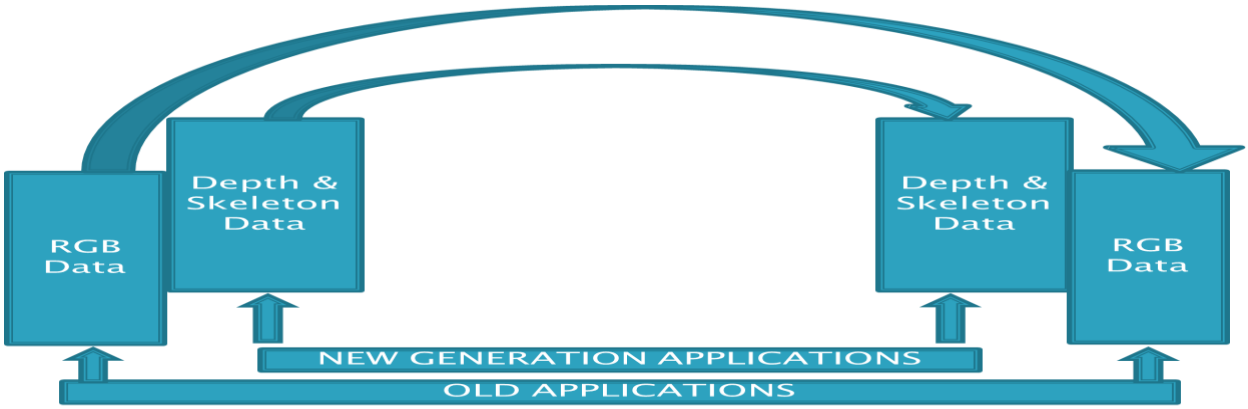



After accesing a hardware device succesfully, another important topic starts which is streaming this devices data over internet in real time. Streaming RGB camera data over internet was popular for last 5 years. There was many applications for streaming RGB cameras data over internet , for example many chat applications are support this feature. After RGB cameras ,

[1] J. Webb, J.Ashley (2009). Beginning Kinect Programming with Microsoft Kinect SDK.

depth cameras has been found. Nowadays, many manufacturer creates their own depth cameras , for example Asus Xtion PRO and Microsoft Kinect.

Figure 1.2 Comparison of old and new generation applications

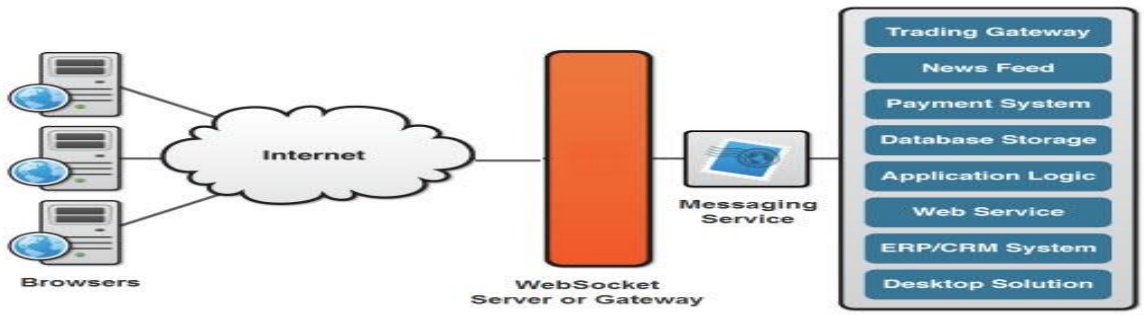



Kinect cameras is a motion sensing input device created by Microsoft for XBOX 360 game consols and Windows PC's. The Kinect sensor is a horizontal bar connected to a small base with a motorized pivot and is designed to be positioned lengthwise above or below the video display. The device features an "RGB camera, depth sensor and multi-array microphone running proprietary software",which provide full-body 3D motion capture, facial recognition and voice recognition capabilities. After Microsoft announced this technology, 24 million units of the Kinect sensor had been shipped and also announced that at new generation personal computers the kinect cameras will be placed integrated. At this point Microsoft created interest on developers and companies for developing applications on this device, but most of the developed applications just working on Windows and Xbox 360 at offline mode. There is limited applications developed for kinect which is working on internet. This work is also related about how to develop kinect applications on internet and stream the kinect data over network.

Also evoluation of web technologies makes easier to stream a kinect camera device data over network. With innovation of HTML5 technology which is standardized at December 2012, gives many significiant opportunities to researchers and companies to stream and process a video input device data. In particular, HTML5 adds many new syntactic features. These include the new <video>, <audio> and <canvas> elements, as well as the integration of scalable vector graphics (SVG) content. These features are designed to make it easy to include and handle multimedia and graphical content on the web without having to resort to proprietary plugins and

APIs [2]. In this study used element for displaying kinect data is the Canvas tag. Canvas tag enables the web browser to natively manipulate, compose, and layer image data [4]. Another advantage of HTML5 technology is the Web-Socket API for transferring the kinect data over network. WebSocket API resolves the issue of sending data directly to the client by allowing the browser to maintain an asynchronous socket connection to a server. By maintaining this connection, the client is able to instantly send data to the server without needing to re-establish a connection. Additionally, the server is able to send data to the client at any time while the connection remains open. The WebSocket API defines a simple protocol to transfer information, and provides a method for creating secure connections which is beneficial for authentication purposes[3].

Figure 1.3 HTML5 Web Socket Protocol




This study is the combination of this technologies to stream kinect data over internet and display this data on web browsers , because there is no best practice of transferring and displaying kinect data over internet , and there is some studies for achieving this problem but the main problem of this applications are using bandwith heavily and connecting to a remote kinect camera is not possible. The difference of this study is to solve this problems, not using bandwith heavily and transferring a remote kinect cameras depth data over network.

[2] J. Webb, J.Ashley (2009). *Beginning Kinect Programming with Microsoft Kinect SDK* (pp. 151-187)
[3] Retrieved December 9, 2013, http://en.wikipedia.org/wiki/Kinect

# 2 RELATED WORKS

## 2.1 KINECT FOR WINDOWS SDK JAVASCRIPT API

The Kinect for Windows SDK JavaScript APIs give HTML5 applications access to Kinect data for interactions and visualization. This allows HTML5 applications running in a browser to connect to the sensor through a server running on the local computer. You can use this to create kiosk applications on dedicated computers. The web server component is a template that can be used as-is or modified as needed.[4] This API enables to connect a kinect server with the help of javascript. When a web page loads , API connects to a locally connected Kinect camera. But this API does not allow to connect a remote Kinect camera , so this is the reason why we did not choose this API for this study.

This API has some advantages and disadvantages. The main advantage of the API is , this API enables to stream all type of Kinect camare data. For using this property the user's need is just a configuration file. The users also can configure the background removal property which is the main observable property of the application. The configuration file is like below : Code 2.1 Configuring a Kinect camera at Kinect for Windows JavaScript API

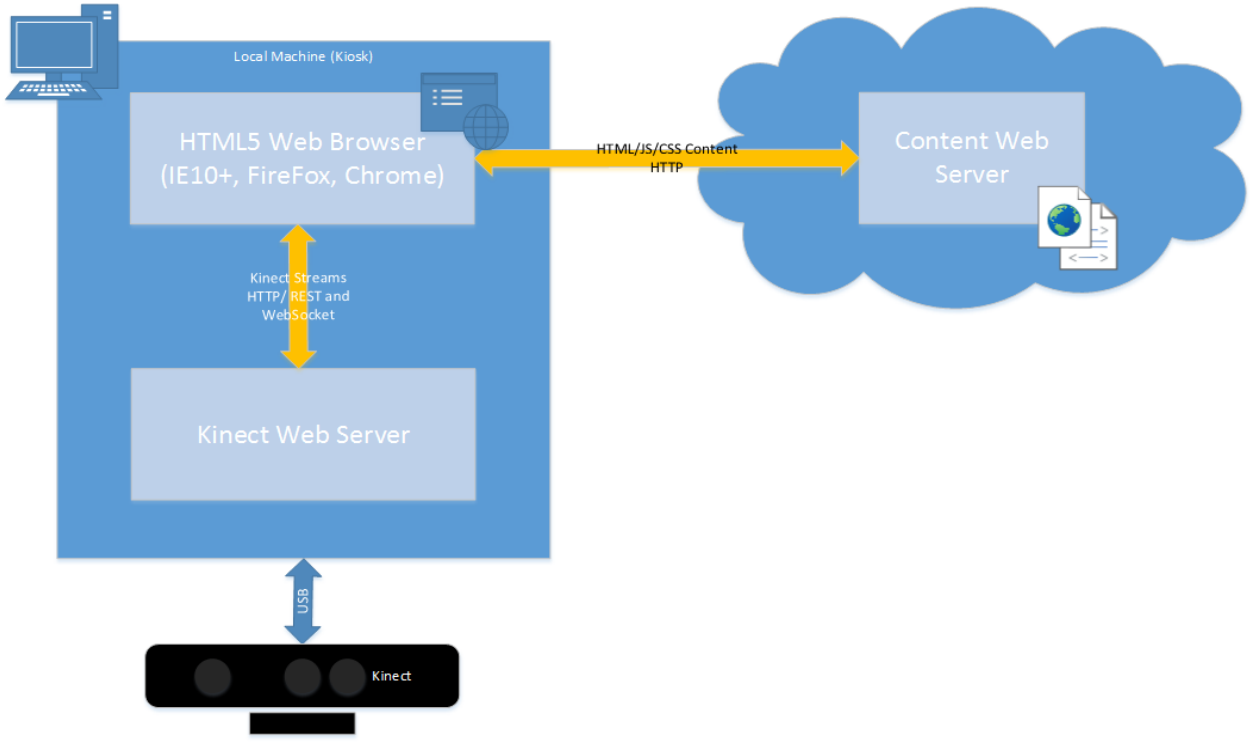
```
var configuration = {
    "interaction" : {
        "enabled": true,
    },
    "userviewer" : {
        "enabled": true,
        "resolution": "640x480", //320x240, 160x120, 128x96, 80x60
        "userColors": { "engaged": 0xffffffff, "tracked": 0xffffffff },
        "defaultUserColor": 0xffffffff, //RGBA },
    "backgroundRemoval" : {
        "enabled": true,
```

[4] Retrieved September 9, 2013, http://msdn.microsoft.com/en-us/library/dn435664.aspx

```
        "resolution": "640x480", //1280x960
    },
    "skeleton" : {
        "enabled": true,
    },
    "sensorStatus" : {
        "enabled": true,
    }
};
sensorToConfigure.postConfig( configuration );
```

The main disadvantage of the application is , the application only takes the locally connected Kinect cameras data. For enabling to connect a remote Kinect camera developers should implement a WPF service to stream the Kinect data. But, this solution also does not allow to stream multiple Kinect data , so this is the another reason for why we did not choose this API. With WPF service the schematic of the application is like below.

Figure 2.1 Microsoft Kinect Javascript API with WPF Service



After accessing a Kinect camera another topic is how to stream and parse this Kinect data. The API gives some functionality about how to stream the Kinect data and how to parse this Kinect data. The developers should add an event listener to stream the Kinect camera's data. And this data is at JSON format so the event listener code will be like below.

Code 2.2 Kinect For Windows Javascipt API event listener

```
sensor.addStreamFrameHandler( function(frame) {
    switch (frame.stream) {
        case Kinect.SKELETON_STREAM_NAME:
            for (var iSkeleton = 0; iSkeleton < frame.skeletons.length; ++iSkeleton) {
                var skeleton = frame.skeletons[iSkeleton];

                skeleton.trackingId;
                skeleton.trackingState;
                skeleton.position;
                for (var iJoint = 0; iJoint < skeleton.joints.length; ++iJoint) {
                    var joint = skeleton.joints[iJoint];
                    joint.jointType;
                    joint.trackingState;
                    joint.position;
                }
            }
        break;
    }
});
```

This API gives some features to stream Kinect data and parse this data ,but does not correspond our needs for streaming the Kinect data over internet so we did not choose this API for this study.

## 2.2 ZIGFU – ZIGJS API

ZigJS allows to have motion-controlled websites using a few lines of code. Written in idiomatic JavaScript, it interacts seamlessly with most popular JavaScript frameworks, including JQuery, prototype.js, and MooTools. The ZDK JavaScript bindings gives access to the depth, image, and skeleton data from the Kinect as well as control over the type of user engagement and some higher-level abstractions for user interface components.[5]

This API has some advantages and disadvantages while streaming depth and skeleton data. The main advantage of the API is , the API allows streaming not only Kinect camera data, but also other types of depth devices such as, Asus X-Motion. But the main disadvantage of the API is it only supports locally connected depth device. So we did not choose this API to implement this study. An example Kinect data streaming code example is like below . This example code gets the Kinect skeleton data and extraxts the position of the head from skeleton data and displays the position of the head on the screen.

Code 2.3 ZigFu Locally Connected Kinect Camera Skeleton Streming Code

```
var engager = zig.EngageUsersWithSkeleton(1);
    engager.addEventListener('userengaged', function(user) {
        console.log('User engaged: ' + user.id);

        user.addEventListener('userupdate', function(user) {
            console.log('Head position: ' + user.skeleton[zig.Joint.Head].position);
        });
    });
    engager.addEventListener('userdisengaged', function(user) {
        console.log('User disengaged: ' + user.id);
    });
zig.addListener(engager);
```

[5] Retrieved Decemver 7, 2013, http://zigfu.com/en/zdk/javascript/

# 3 PROGRAMMING WITH KINECT

Every Kinect application has certain basic elements. The application must detect or discover attached kinect sensors. It must then initialize the sensor. Once initialized, the sensor produces data, which the application then processes. Finally, when the application finishes using the sensor it must properly uninitialized the sensor.

## 3.1 INITIALIZATION PHASE

Application development with kinect starts with KinectSensor object which placed at Microsoft.Kinect namespace. The initialization phase of the Project starts with Kinect initialization. Then KinectSensor status flag which name is KinectStatus ,set to an appropriate status. This status value is listed below.

Table 3.1 : Kinect Device Connection Status

| **KinectStatus** | **What it means** |
|---|---|
| Undefined | The status of the attached device cannot be determined. |
| Connected | The device is attached and is capable of producing data from its streams. |
| DeviceNotGenuine | The attached device is not an authentic Kinect sensor. |
| Disconnected | The USB connection with the device has been broken. |
| Error | Communication with the device produces errors. |
| Initializing | The device is attached to the computer, and is going through the process of connecting. |
| InsufficientBandwidth | Kinect cannot initialize, because the USB connector does not have the necessary bandwidth required to operate the device. |
| NotPowered | Kinect is not fully powered. The power provided |

| | |
|---|---|
| | by a USB connection is<br>not sufficient to power the Kinect hardware. An additional power adapter<br>is required. |
| NotReady | Kinect is attached, but is yet to enter the Connected state. |

## 3.2 KINECT DATA STREAMING

After KinectSensor's KinectStatus flag has been set to the Connected , Kinect device will be ready for data streaming. Kinect cameras provides different types of data for streaming. This data types are ; ColorImageStream , DepthStream , Skeleton Tracking Stream and AudioStream. After the initialization phase of kinect programming, the programmer should enable this streamings and register streaming event handlers to initialized kinect camera. Sample codes for enabling data stream and registering event handlers is like below :

a. Color Image Stream :

```
KinectDevice.ColorStream.Enable();
KinectDevice.ColorFrameReady += Kinect_ColorFrameReady;
```

b. Depth Image Stream:

```
KinectDevice.DepthStream.Enable();
KinectDevice.DepthFrameReady += KinectDevice_DepthFrameReady;
```

c. Skeleton Tracking Stream:

```
this._KinectDevice.SkeletonStream.Enable();
this.KinectDevice.SkeletonFrameReady +=KinectDevice_SkeletonFrameReady;
```

After event handlers has been registered to the kinect camera, when the camera produces new data the event handler will be triggered. A kinect camera can produce 30 frames of data per second for color image streaming , 60 frames of data per second for depth streaming, and 30 frames of data per second for skeleton stremaing. Event handlers sample code is like below :

a. Color Image Streming :

Code 3.1 : Kinect Camera RGB Data Event Handler

```
void Kinect_ColorFrameReady(object sender,ColorImageFrameReadyEventArgs e)
{
    using(ColorImageFrame frame = e.OpenColorImageFrame())
    {
        if(frame != null){
            byte[] pixelData = new byte[frame.PixelDataLength];
            frame.CopyPixelDataTo(pixelData);
        }
}
```

When the Kinect camera produces a RGB data frame , Microsoft Kinect SDK calls the registered event handler method which is shown ar Code 3.1. At this code EventArgs contains the produced data frame. This data frame is a byte array, and the developer can manipulate this data according to application requirements.

b. Depth Streaming :

Code 3.2 : Kinect Camera Depth Data Event Handler

```
void Kinect_DepthFrameReady(object sender,DepthImageFrameReadyEventArgs e)
{
    this._LastDepthFrame = e.OpenDepthImageFrame();
    this._LastDepthFrame.CopyPixelDataTo(this._DepthImagePixelData);
}
```

When the Kinect camera produces a depth data frame , Microsoft Kinect SDK calls the registered event handler method which is shown ar Code 3.2. At this code EventArgs contains the produced data frame. This data frame is a byte array, and the developer can manipulate this data according to application requirements.

c. Skeleton Streaming :

Code 3.3: Kinect Camera Skeleton Data Event Handler

```
private void KinectDevice_SkeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e){
    using(SkeletonFrame frame = e.OpenSkeletonFrame()){
    frame.CopySkeletonDataTo(this._FrameSkeletons);
     // Do Some Data operations
    }
}
```

When the Kinect camera catches a skeleton at captured frame, Microsoft Kinect SDK calls the registered event handler method which is shown at Code 3.3. At this code EventArgs contains the produced skeleton data. This data frame is a class object which is at namespace SkeletonFrame,and this skeleton frame contains maximum 6 persons to track their skeleton data, , and the developer can manipulate this data according to application requirements.

While the data streaming , programmer should consider about the data size and should choose the suitable data streaming format for own program. Data format examples is like below. Color Streming data formats shown at table 1.2 below :

Table 3.2 : Color Streaming Data Formats and Properties

| **ColorImageFormat** | **What it means** |
|---|---|
| RgbResolution640x480Fps30 | The image resolution is 640x480, pixel data is RGB32 format<br>at 30 frames per second. |
| RgbResolution1280x960Fps12 | The image resolution is 1280x960, pixel data |

| | |
|---|---|
| | is RGB32<br>format at 12 frames per second. |
| YuvResolution640x480Fps15 | The image resolution is 640x480, pixel data is YUV format at<br>15 frames per second. |
| RawYuvResolution640x480Fps15 | The image resolution is 640x480, pixel data is raw YUV<br>format at 15 frames per second |

While configuring the Kinect camera at the initialization phase of the application , the developer can choose one of this color image formats, and the Kinect camera produces the RGB data according to this configurative Color Image Format value.

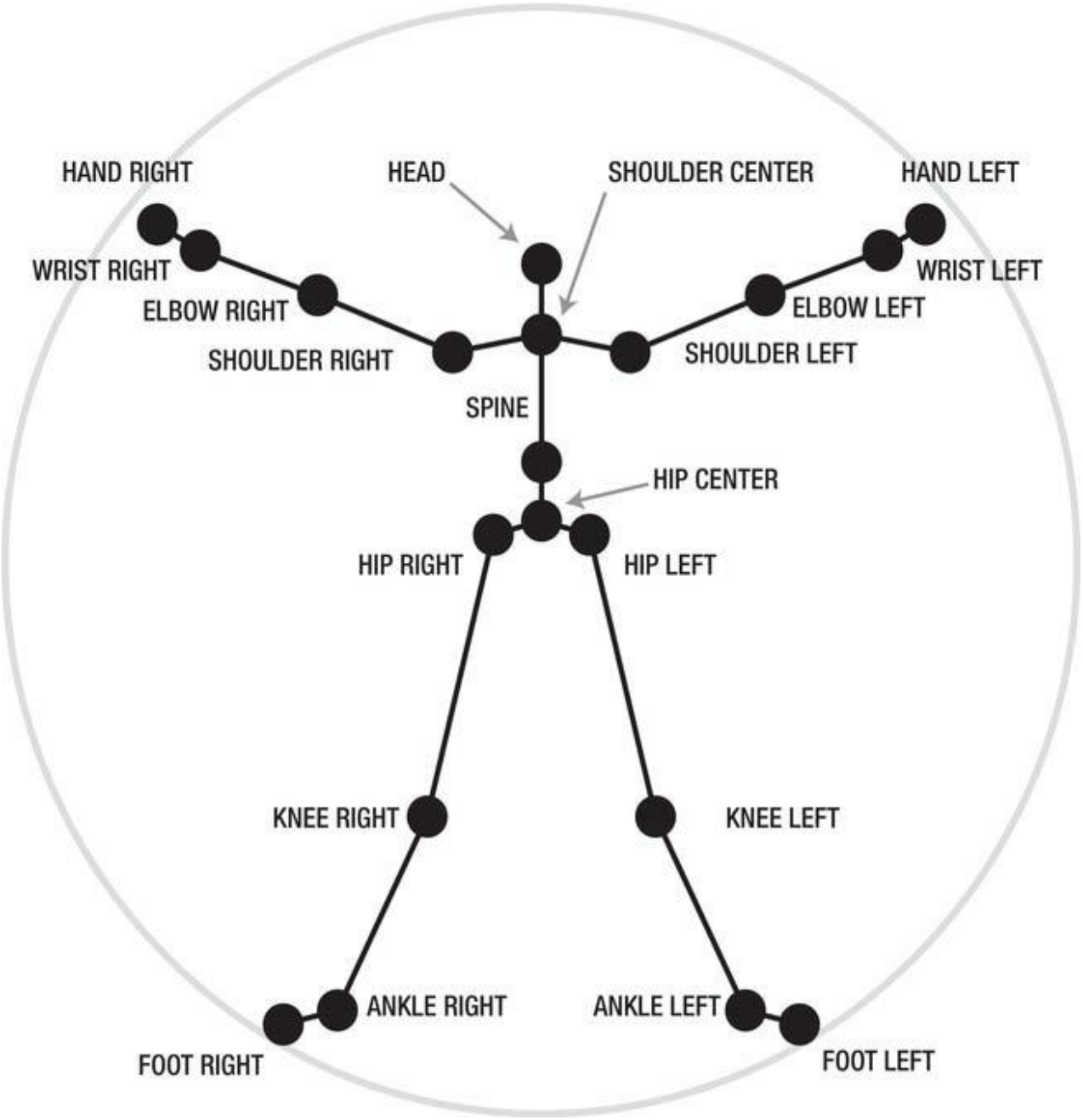
Kinect camera produces 16 bits of data for depth streaming process. This data model is shown at Figure 3.2.

Figure 3.2 : Depth Streming data model


This means for a 640*480 resolution frame kinect produces 16*640*480 bits. Also programmers can determine the player with mask 0000 0111.

At skeleton tracking, when the Kinect camera captures a skeleton , the Microsoft Kinect SDK produces a SkeletonFrame data, and this data contains the skeleton joint's x,y,z, positions to determine the skeletons position. SkeletonFrame data can contain maximum 6 people's skeleton data, because Kinect camera tracks maximum 6 people at real time. This joints is shown at Figure 3.3 which kinect camera produces.

Figure 3.3 : Kinect Skeleton Tracking Joint Positions




Source : J. Webb, J.Ashley (2009). *Beginning Kinect Programming with Microsoft Kinect SDK* (pp. 158-160)

# 4 HTML5 , CANVAS and WEB-SOCKET and JSON

## 4.1 WHAT IS HTML5?

HTML5 is the latest evolution of the standard that defines HTML. The term represents two different concepts. It is a new version of the language HTML, with new elements, attributes, and behaviors, and a larger set of technologies that allows more diverse and powerful Web sites and applications.[6] With HTML5 evolution , for submission to developers some new features and concepts has been come. HTML5 allows developers to communicate with the server in new and innovative ways.

Table 4.1 : New Features of HTML5 Technology

| | |
|---|---|
| Offline & Storage | allowing webpages to store data on the client-side locally and operate offline more efficiently. |
| Multimedia | making video and audio first-class citizens in the Open Web |
| 2D/3D Graphics & Effects | allowing a much more diverse range of presentation options |
| Performance & Integration | providing greater speed optimization and better usage of computer hardware. |
| Device Access | allowing for the usage of various input and output devices |
| Styling | letting authors write more sophisticated themes |

As shown at Table 4.1 HTML5 technology gives some significiant features to developers to develop more interactive and innovative solutions. HTML5 technology gives opportunity to communicate with input devices synchroneous. This is the main reason why we choose this technology at our project.

## 4.2 What is CANVAS?

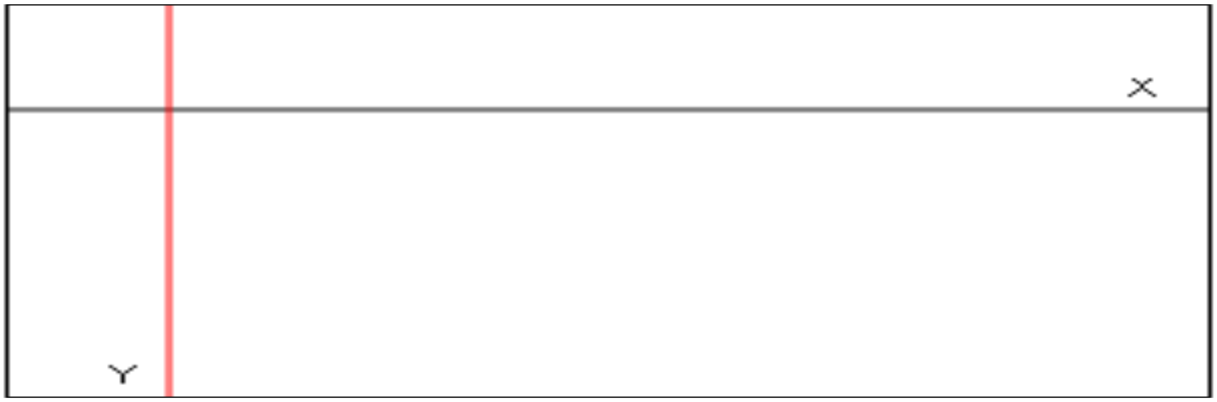
Officially a canvas is "a resolution-dependent bitmap canvas which can be used for rendering graphs, game graphics, or other visual images on the fly". In layman's terms, the canvas is a new element in HTML5, which allows you to draw graphics using JavaScript. It can be used to render text, images, graphs, rectangles, lines gradients and other effects dynamically. Drawing on the canvas is via the canvas 2D API. This API contains a plethora of functions that give you the power

[6] Retrieved December 11, 2013, https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5

to draw pretty much anything you like on the canvas[7] In this study we used canvas element also , because canvas objects gives opportunity to drav the streamed Kinect cameras data. With the reference to the canvas elements context , we could draw the skeleton shapes on 2D surface.

Every canvas element has x and y coordinates. X being the horizontal coordinate and y being the vertical coordinate. The following image shows these coordinates on a canvas.

Figure 4.1. : HTML5 Canvas Element coordinate scheme




When we think about our project, while Kinect cameras data has been arrived to the browser , we parse this data and show this data on this canvas object with the help of x and y coordinate with some mathematical equations.

### 4.2.1 Canvas and Hardware Acceleration

Using canvas is the best way to learn about hardware acceleration on the web. In earlier versions of browsers, graphics rendering – like most compute intensive tasks – was handled by the CPU, the central processing unit. Modern browsers have been innovating in this area by shifting graphic-intensive tasks to the GPU, the graphics processing unit, to render the graphics and text on a web page using Direct2D and DirectWrite. The allocation of these tasks to the GPU cores not only accelerates the graphics processing but also eases the load on the CPU while it takes care of the serial tasks more efficiently. [8]

[7] Retrieved December 11, 2013, https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5
[8] Retrieved December 8, 2013,http://www.html5canvastutorials.com/

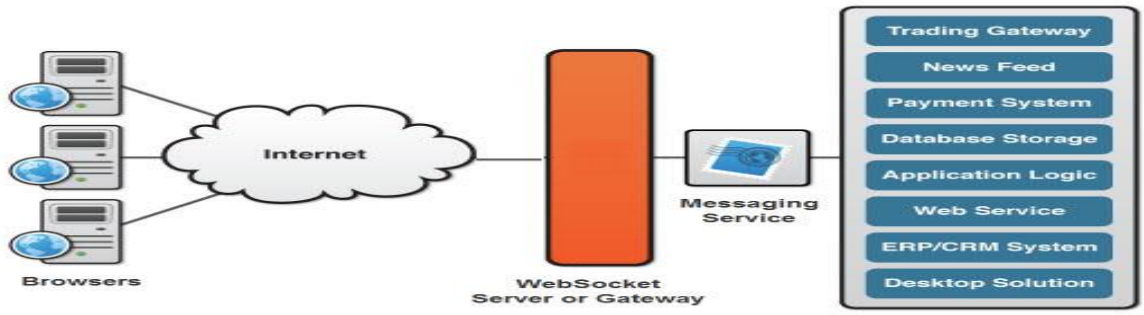
When we send drawing commands to the canvas, the browsers sends them directly to the graphics hardware without further development on your part. The hardware acceleration works incredibly fast to deliver real time animations and interactive graphics, without slowing down the surrounding user experience we're delivering.

## 4.3 WEB-SOCKET

The HTML5 WebSockets specification defines an API that enables web pages to use the WebSockets protocol for two-way communication with a remote host. It introduces the WebSocket interface and defines a full-duplex communication channel that operates through a single socket over the Web. HTML5 WebSockets provide an enormous reduction in unnecessary network traffic and latency compared to the unscalable polling and long-polling solutions that were used to simulate a full-duplex connection by maintaining two connections.[9]

In this study we also used HTML5 Web Sockets to stream remote connected Kinect cameras skeleton and the depth data. Because of the uninterrupted connection support of the Web-Socket technology we choosed this technology. Also web socket technology reduces the network traffic and latency for real time applicatios like our application. And web socket technology also supports the secured connections and proxies . So these benefits of the technology is the reasons why we choose this technology for this study.

Figure 4.2 : HTML5 Web Socket Architecture




[9] Retrieved Decemver 10, 2013,http://www.websocket.org/aboutwebsocket.html

### 4.3.1 The WebSocket Protocol

The WebSocket protocol was designed to work well with the existing Web infrastructure. As part of this design principle, the protocol specification defines that the WebSocket connection starts its life as an HTTP connection, guaranteeing full backwards compatibility with the pre-WebSocket world. The protocol switch from HTTP to WebSocket is referred to as a the WebSocket handshake.

The browser sends a request to the server, indicating that it wants to switch protocols from HTTP to WebSocket. The client expresses its desire through the Upgrade header:

Table 4.1 : Web Socket Protocol Client to Server Handshake Request

```
GET ws://echo.websocket.org/?encoding=text HTTP/1.1
Origin: http://websocket.org
Cookie: __utma=99as
Connection: Upgrade
Host: echo.websocket.org
Sec-WebSocket-Key: uRovscZjNol/umbTt5uKmw==
Upgrade: websocket
Sec-WebSocket-Version: 13
```

If the server understands the WebSocket protocol, it agrees to the protocol switch through the Upgrade header.

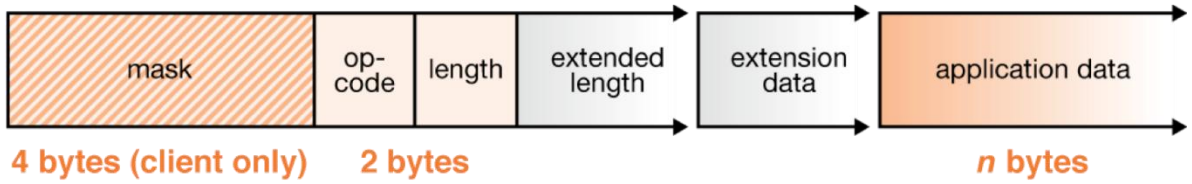
Table 4.2 : Web Socket Protocol Server to Client Handshake Response

```
HTTP/1.1 101 WebSocket Protocol Handshake
Date: Fri, 10 Feb 2012 17:38:18 GMT
Connection: Upgrade
Server: Kaazing Gateway
Upgrade: WebSocket
Access-Control-Allow-Origin: http://websocket.org
Sec-WebSocket-Accept: rLHCkw/SKsO9GAH/ZSFhBATDKrU=
Access-Control-Allow-Headers: content-type
```

At this point the HTTP connection breaks down and is replaced by the WebSocket connection over the same underlying TCP/IP connection. The WebSocket connection uses the same ports as HTTP (80) and HTTPS (443), by default.

Once established, WebSocket data frames can be sent back and forth between the client and the server in full-duplex mode. Both text and binary frames can be sent in either direction at the same time. The data is minimally framed with just two bytes. In the case of text frames, each frame starts with a 0x00 byte, ends with a 0xFF byte, and contains UTF-8 data in between. WebSocket text frames use a terminator, while binary frames use a length prefix.[10]

Figure 4.3 : WebSocket dataframe data scheme




## 4.4 JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.[11] JSON data format uses universal data structures. Virtually all modern programming languages support this data structures. It makes sense that a data format that is interchangeable with programming languages also be based on these structures. JSON data format is built in two data formats. A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence. Because of the JSON format is the closest to the javascript programming language , and also we used javascript at client side to handle the kinect data , we choosed this technology for interchanging the data. If we want to list the advantages of the JSON data format , briefly we can

[10] Retrieved Decemver 10, 2013,http://www.websocket.org/aboutwebsocket.html
[11] Retrieved Decemver 13, 2013,http://www.json.org/

say that interchanging a data makes the data more clear , more efficient and scalable for other applications. Because, using JSON objects means your data has a strict, standardized structure. With this strict structure in place there is less chance for error. The application receiving the JSON data knows what to expected and how to receive the data.[12] Here is an example of how json data looks like.

Code 4.1 : JSON Data Format Example Code

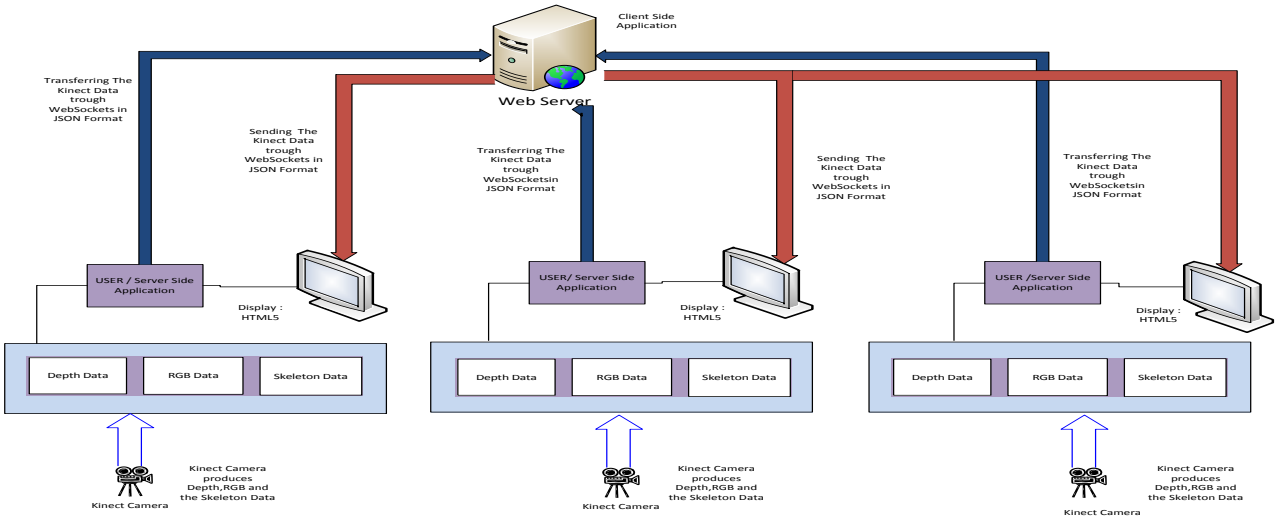
```
{
    "total":3,
    "results":[
        {"id":3459,"fname":"Leon","sname":"Revill"},
        {"id":3460,"fname":"Darth","sname":"Vader"},
        {"id":3461,"fname":"Samwise","sname":"Gamgee"}]
    }
}
```

[12] Retrieved Decemver 13, 2013,http://www.json.org/

# 5 MULTIPLE KINECT STREAMING APPLICATION

Accessing a kinect camera over internet is so popular nowadays. Many developer and research company are working on this hot topic and find several solutions. Most of this applications have some avantages and disadvantages. The main disadvantage of this applications is consuming network bandwidth heavily when transferring kinect data. Because transferring whole of kinect data binds heavily mass of data to the network, so the developers of kinect applications should consider about this topic and should mine and comb out the unnecessary kinect data. This applications also directly access the kinect device from javascript and this approach only works for local applications and not support multiple kinecting display on a single browser. There is no best case solutions for this kind of problems. In this thesis solutions have been found for this problems and disadvantages. Briefly ; for network bandwidth problems kinect data has been mined and unnnecesarry kinect data combed out and the necessary data converted in json format before streaming over network, and for multiple kinect support; application divided into two parts which are server side application and client side application and with the help of this approach some performance problems cleared and network bandwith problem also cleared before network streaming and also with this approach more than one kinect cameras data could streamed over network and displayed on a single web page. Schematic of this approach is like below :

Figure 5.1 : Multi Kinect Application Scheme




Thesis is composed of from two applications. First application is the server side application and second application is the client side application

## 5.1 KINECT STREAMING SERVER SIDE APPLICATION

The server side application is a console application which briefly doing this operations. The user logins to the system and the system assigns a suitable port to this user with the help of a web servis located at client side, after this operation , application takes the frame data from the kinect camera and tracks the skeletons from this data and find the primary skeleton and serialize primary skeletons data to json format and finally write this json data to a assigned port for client applications to read from this port.

Server Side Application composed of from 4 sections.

### 5.1.1 User Module Initialization

When the user runs the server application the console application asks to user a username and password for logged in the application. If the user doesn't provide correct username and password combination the user can not be reach the application. Else; if the username , password combination is correct the application asks the local machine which port is available for transferring the kinect camera data. With this port value and the locale machine's IP adress the users port and ip adress will be updated at database while the user is active and session is not expired.

Code 5.1 : User Module Initialization

```
user=Dao.doLogin(userName, password);
        if (user != null) {
            string port = getAvailablePort();
            user.port = port;
            Dao.updateUserPort(user);
}
```

At Code 5.1 getAvailablePort() function returns the applications runs machine's IP adress and an available port value at local machine to communicate with the remote client application.

GetAvailablePort() function code is like below :

Code 5.2 : Getting available port value and the IP adress with using System.Net.Networking namespace

```
public static string getAvailablePort() {
        string returnPort = "";          bool isAvailable = true;
        IPGlobalProperties ipGlobalProperties = IPGlobalProperties.GetIPGlobalProperties();
        TcpConnectionInformation[] tcpConnInfoArray =
        ipGlobalProperties.GetActiveTcpConnections();
        for (int port = 400; port <= 9999; port++)
        {
           foreach (TcpConnectionInformation tcpi in tcpConnInfoArray)
           {
              if (tcpi.LocalEndPoint.Port == port)
              {
                 isAvailable = false;
                 break;
              }
           if (isAvailable)  {
              returnPort = port.ToString();
              break;
            }
        String strHostName = string.Empty;.
        strHostName = Dns.GetHostName();
        IPHostEntry ipEntry = Dns.GetHostEntry(strHostName);
        IPAddress[] addr = ipEntry.AddressList;
        return addr[0].ToString() + ":" + returnPort;
     }
```

At this function the application firstly looks at the active TCP connections at the local machine with the System.Net.Networking namespace. After finding the active TCP connections with the function *"GetActiveTcpConnections"*, applications defines a port with a for loop which is not in use. After port value has been defined , application gets the local machine's IP adress in IPv6 protocol format with using *" IPHostEntry ipEntry = Dns.GetHostEntry(strHostName)"* function which is located at System.Net package. After port and the IP value defined successfully function returns this value in "IP Adress : Port" format.

This database operation codes is like below :

Code 5.3 : User Module Initialization MySql Database Login Operation

```
public static User doLogin(string userName,string password) {
        MySqlConnection connection = new MySqlConnection(MyConString);
        MySqlCommand command = connection.CreateCommand();
        MySqlDataReader Reader;
        command.CommandText = "SELECT * " + "FROM User " +
       "WHERE UserName = '" + userName + "' AND UserPassword = '" + password + "'";
        connection.Open();
        Reader = command.ExecuteReader();
         User user = new User();        Reader.Read();
        if (Reader.HasRows) {
            user.userName= Reader["UserName"].ToString();
           user.password = Reader["UserPassword"].ToString();
           user.userId = Int32.Parse(Reader["UserId"].ToString());
           user.port = Reader["Port"].ToString();
           user.gameRoom = Reader["GameRoom"].ToString();}
        Reader.Close();
        connection.Close();
 return user; }
```

At Code 5.3 application connects to the MySql database. After connection done successfully application runs a select query at database with user's provided username and password value. If the user exists at User table , function returns a user object to the main application.

Update Port :

Code 5.4 : Updating User Port Value Mysql Database Operation

```
public static void updateUserPort(User user)
        {
          MySqlConnection connection = new MySqlConnection(MyConString);
          MySqlCommand command = connection.CreateCommand();
          command.CommandText = "Update User " +
          "SET Port = '" + user.port+ "' WHERE UserId = '" + user.userId+ "'";
          connection.Open();
          command.ExecuteNonQuery();
}
```

At Code 5.4 successfully logined user's port and the Ip value updates with the value of returned value from "*getAvailablePort*" which is described at Code 5.2.

After the initialization of user and the user port , the next action is initialization of Kinect camera.

### 5.1.2 Kinect Camera Initialization

At Kinect camera initialization phase skeleton tracking will be enabled for the connected Kinect camera and if the application can connect to this camera ,a listener function named "*kinect_SkeletonFrameReady*" will be registered to connected camera . When the camera captures a skeleton , the Microsoft Kinect SDK will call this registered function. After all of this operations the camera will be ready for skeleton data streaming.

Kinect initialization code is like below.

Code 5.5 : Initialization of connected Kinect Camera

```
private static void InitilizeKinect(){
                kinect = KinectSensor.KinectSensors.FirstOrDefault(s => s.Status ==
                KinectStatus.Connected);
                if (kinect != null) {
                 kinect.SkeletonStream.Enable();
                  skeletonData = new
                Skeleton[kinect.SkeletonStream.FrameSkeletonArrayLength];
                 kinect.SkeletonFrameReady += new
                EventHandler<SkeletonFrameReadyEventArgs>(kinect_SkeletonFrameReady); //
                kinect.Start();
}
```

### 5.1.3 Web-Socket Initialization

After kinect camera has been initialized and the application defined a suitable port and IP value, application creates a WebSocket for streaming the Kinect camera data.

Initialization of a Web Socket is like below code :

Code 5.6 : Initialization of Web Sockets for transferring the Connected Kinect Camera data

```
 private static void InitializeSockets(string port) {
        _sockets = new List<IWebSocketConnection>();
        var server = new WebSocketServer("ws://localhost:" + port);
        server.Start(socket =>  {
           socket.OnOpen = () => {
              _sockets.Add(socket);
           };
           socket.OnMessage = message => {
              Console.WriteLine(message);
           };
        }
};
```

### 5.1.4 Kinect Data Processing

After all operations completed succesfully , and the kinect camera initialized succesfully a handler function should be registered succesfully to the kinect camera for calling at every data process action. In this case just skeleton tracking function will be enabled and when a skeleton object recognized, the event handler will be called by the Kinect SDK.

Code 5.7 : Registered Kinect Camera Skeleton Frame Ready Handler Code

```
private static void kinect_SkeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e)
{
        using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
        {
           if (skeletonFrame != null && skeletonData != null)
           {
              skeletonFrame.CopySkeletonDataTo(skeletonData);
           }
        }

        if (skeletonData.Length > 0) {
           Skeleton primaryUser = GetPrimarySkeleton(skeletonData);
           string json = primaryUser.Serialize();
           foreach (var socket in _sockets)
           {
              socket.Send(json);
           }
        }
}
```

At this point ; skeleton's data is ready for processing and the handler method reached the data, but the application should choose the primary user for our business model.Kinect may catch more than one skeleton data so; for achieving this operation,application finds the closest skeleton wtih the

skeletons depth data.For achieving to find the closest skeleton form the skeleton list , code compares the skeletons depth value and decides which skeleton is closest to the camera.

Extracting the primary user's skeleton data can be reached from code below :
Code 5.8 : Finding the Primary Skeleton from Skeleton depth value

```
private static Skeleton GetPrimarySkeleton(Skeleton[] skeletons)  {
        Skeleton skeleton = null;
        if (skeletons != null)
        {
           for (int i = 0; i < skeletons.Length; i++)
           {
              if (skeletons[i].TrackingState == SkeletonTrackingState.Tracked)
              {
                 if (skeleton == null)
                 {
                    skeleton = skeletons[i];
                 }
                 else
                 {
                    if (skeleton.Position.Z > skeletons[i].Position.Z) {
                       skeleton = skeletons[i];
                    }
                 }
                 return skeleton;
           }
        }
     }
```

After the primary skeleton data extracted from all skeletons data , application should serialize this data because skeleton object is not serializable and too big for network access. If the application sends the primary skeleton's data directly to the client application, client apllication should be

handle this data at backend and this case is very bad for performance considerations. So the server application converts and serializes the skeleton data.

Skeleton data consists a skeleton ID , and a joint list. Joint list also consists joints and each joint has an x,y,z coordinates and a jointId. For transferring this data over a network the application has a serialize method which converts this skeleton data to JSON format and writes this JSON data to initialized port.

Skeleton data JSON format is like below :

Code 5.9: Skeleton Data JSON Contract Code with C# Programming Language

```
[DataContract]
class JSONSkeletonCollection{
    [DataMember(Name = "skeletons")]
    public List<JSONSkeleton> Skeletons { get; set; }
}
[DataContract]
class JSONSkeleton {
    [DataMember(Name = "id")]
    public string ID { get; set; }
    [DataMember(Name = "joints")]
    public List<JSONJoint> Joints { get; set; }
}
[DataContract]
class JSONJoint{
    [DataMember(Name = "name")]
    public string Name { get; set; }
    [DataMember(Name = "x")]
    public double X { get; set; }
    [DataMember(Name = "y")]
    public double Y { get; set; }
    [DataMember(Name = "z")]
    public double Z { get; set; }
}
```

And a serialize function converts the skeleton's data to this JSON format. This function code is like below :

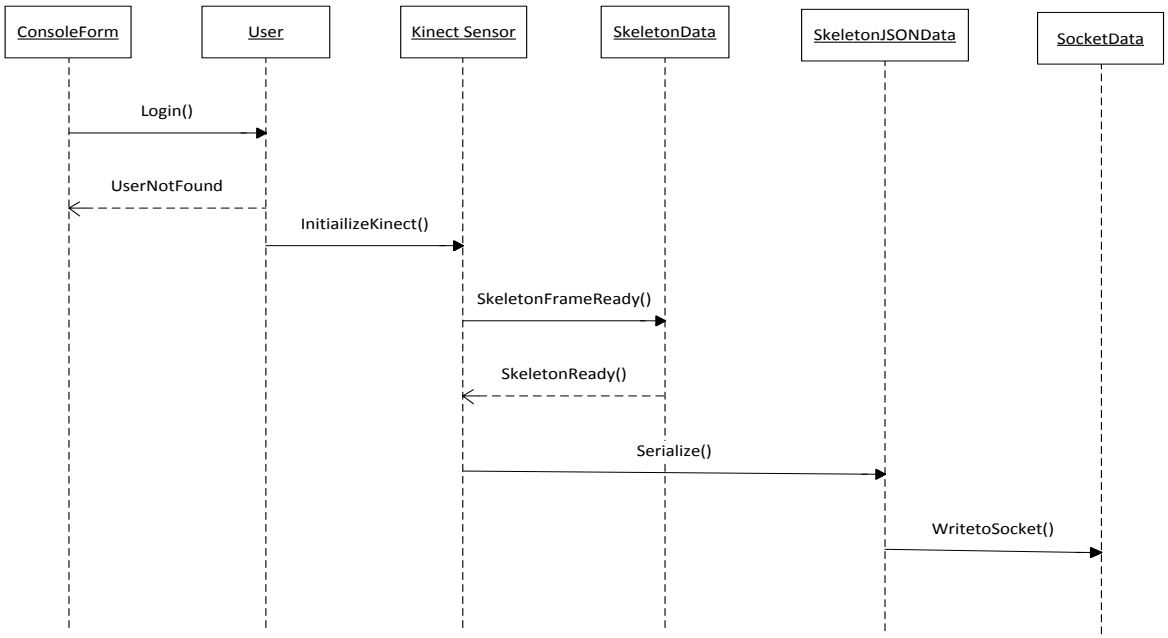
Code 5.10 : Serializing and Converting Kinect Skeleton Data to JSON Format

```
public static string Serialize(this Skeleton skeleton)
{
    JSONSkeleton jsonSkeleton = new JSONSkeleton
    {
        ID = skeleton.TrackingId.ToString(),
        Joints = new List<JSONJoint>()
    };
    foreach (Joint joint in skeleton.Joints)
    {
        Joint scaled = joint.ScaleTo(640, 480);

        jsonSkeleton.Joints.Add(new JSONJoint
        {
            Name = scaled.GetHashCode().ToString(),
            X = scaled.Position.X,
            Y = scaled.Position.Y,
            Z = scaled.Position.Z
        });
    }
  return Serialize(jsonSkeleton);
  }
```

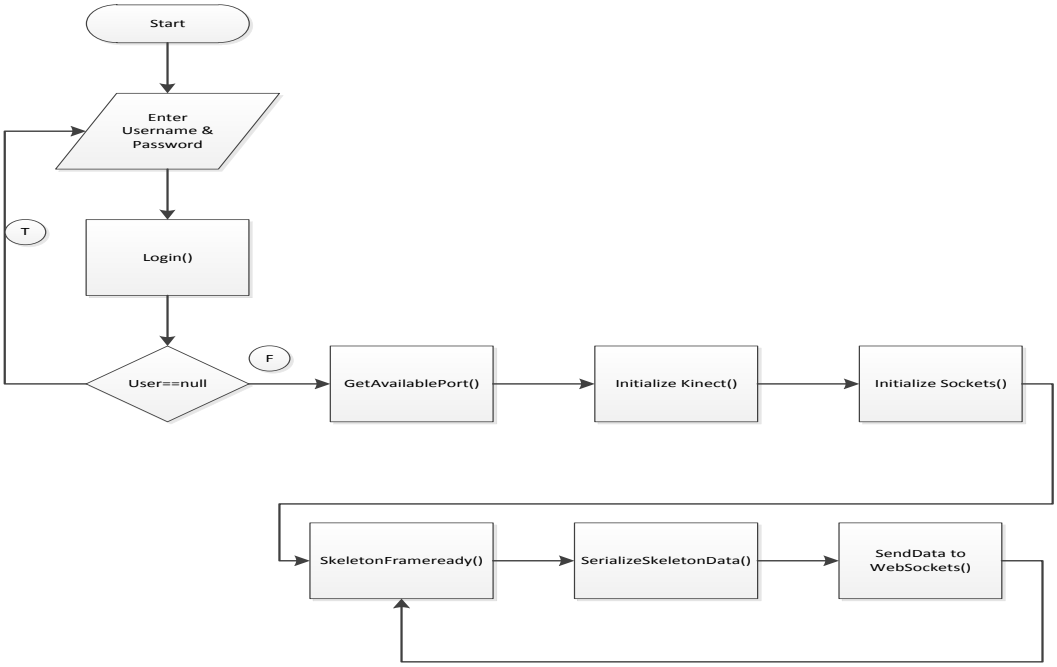
This converted data will be transferred over network from oppened Web Socket connection while kinect camera tracks skeletons.

Server side applications sequence diagram is like below :
Figure 5.2 : Server Side Application Sequence Diagram




Flow chart of the server side application is like below :

Figure 5.3 : Server Side Application Flow Chart Diagram



## 5.2 KINECT STREAMING CLIENT SIDE APPLICATION

Second part of the study is kinect streaming client side application. Client side application is Microsoft .NET Web Application which is deployed on a Microsoft IIS Server. It consists from a Login.aspx page , a Kinect.aspx page and an asmx Webservice for server side applications to ask suitable port. All of the business for displaying kinect data on web page is doing by javascript files because web pages is constructed on web sockets which server side applications write their data to this web sockets. Client Side application composed of 4 parts.

### 5.2.1 Kinect Login Page

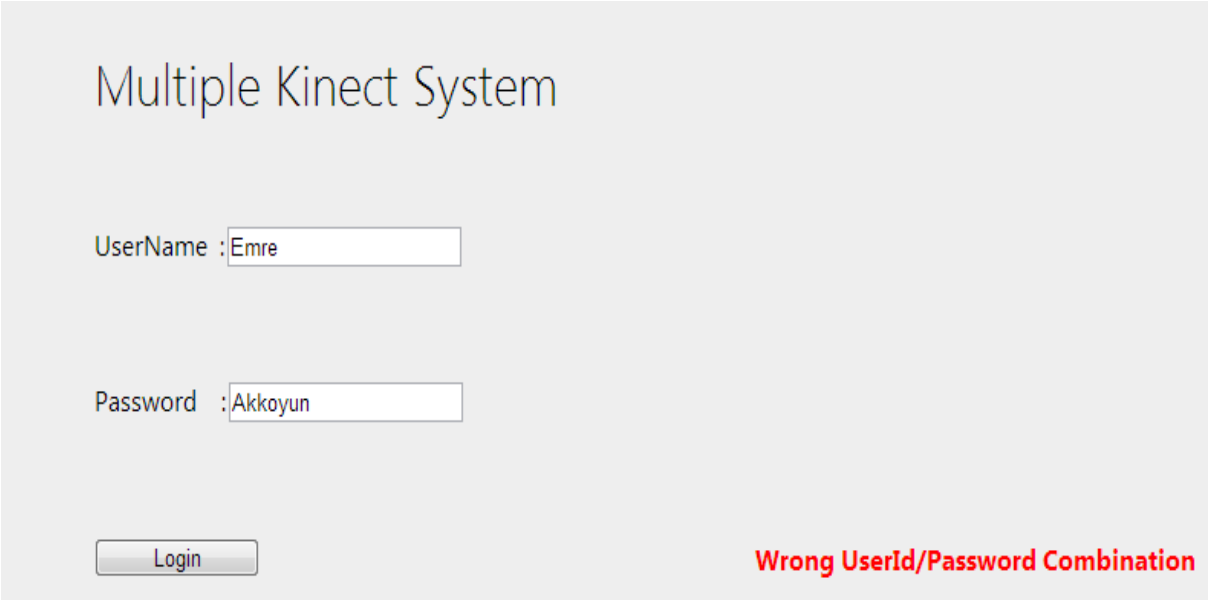
User of the system logins to the system with userName/Password combination. If this combination is true then application takes the user's information and stores this information at HttpSession object. If the user doesn't run the server applcation before the login operation application warns the user to run the server application because before connecting to the client system user have to run server application to take a port for transporting the kinect data to this client application. If the user has runned the server application before login operation ,client application loads the user port value from database and stores it at HttpSession which mentioned before and redirect the user to KinectDataStreaming page. Login operation Code is like below:
Code 5.11 : Client Side Application Login Controller Code

```
User loggedInUser = Dao.doLogin(txtUserName.Text, txtPassword.Text);
if (loggedInUser == null) {
   "Wrong UserId/Password Combination";
}
else{
   if (loggedInUser.port == "")
        "Firstly run the Console Application For Connecting Kinect";
   else{
      Session.Add("CurrentUser", loggedInUser);
      Response.Redirect("Kinect.aspx");
   }
```

And here is some screenshots from the login page of the system :

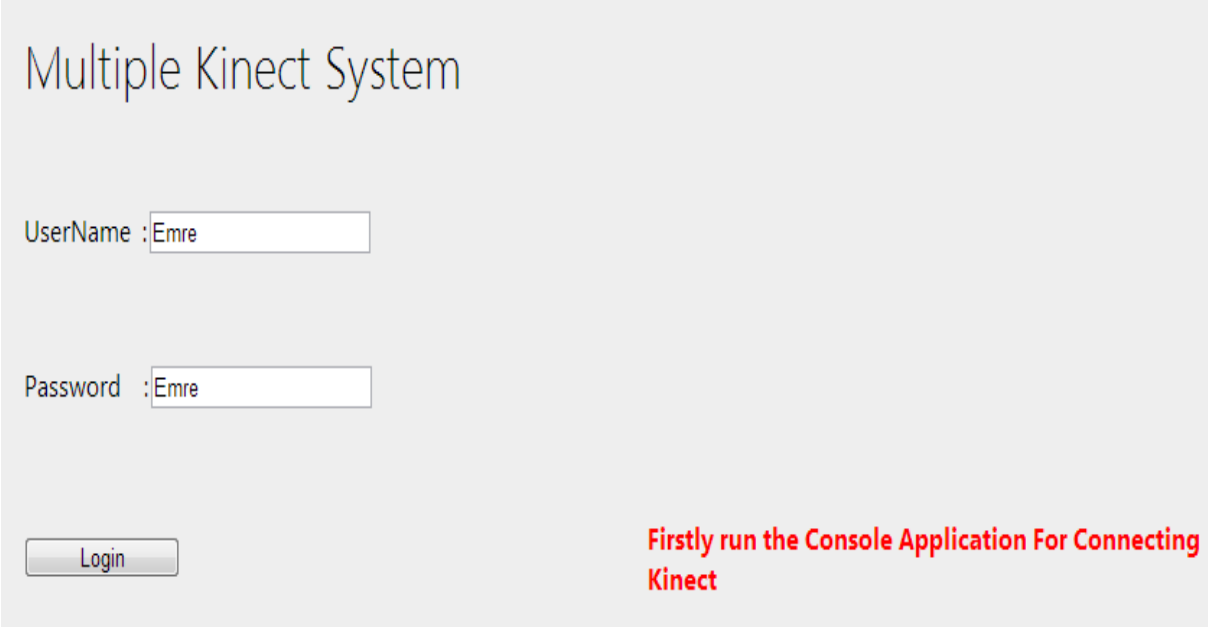
If the username / password combination is incorrect :

Figure 5.4 : Login Page Wrong Login Attempt




If the user gives the correct username/password combination and doesn't run the Kinect Server application before login operation :

Figure 5.5 : Login Page Console Application not Runned



### 5.2.2 Game Room Page

After login operation done succesfully and the server side application has been runned , user will be redirected to game room page. On page load of this page , system looks at the user's gameroom value from database and if the value is null , a list of available users ,who runs the console application and has a valid port value ,will be appear and the current user can choose one of this users for playing the game with. After choosing a user from the list , current user's and the selected user's gameroom value will be the same value and they will can see each others kinect data and their port values will be send to the page as parameter for reading the kinect datas. If the user's gameroom value is not null this means that, another user has been choosed the user and set a gameroom value to this user and at page load the user will be redirect to this gameroom and a list of active users will not be appear to this user. Displaying kinect datas on a web page constructed with the help of HTML5 canvas objects. After all initialization phase of the page finished from backend of the code a javascript function will be triggered and this javascript function creates web socket connection at page and HTML5 canvas object will be filled with data which come from this server ports. Page load code is looks like below :

Code 5.12 : Logged In User's Another User Selection Code

```
if (currentUser.gameRoom == "")
{
    lblStatus.Text = "Please Select a user";
    if (!Page.IsPostBack) {
        ListItem defaultItem = new ListItem("Please Select", "-99");
        drpActiveMembers.Items.Add(defaultItem);
        List<User> userList = Dao.getLoggedInAndNotActiveUsers();
        foreach (User usr in userList)
        {
            ListItem item = new ListItem(usr.userName, usr.port);
            drpActiveMembers.Items.Add(item);
        }
    }
}
```

```
//If  LoggedIn user is registered to a room , Find other user at same room
else {
        List<User> userList =
        Dao.getOtherUserorUserListFromGameRoomInfo(currentUser.gameRoom,
        currentUser.port);
    secondUser = userList[0];
    lblStatus.Text = "Currently Connected to " + secondUser.userName;
    drpActiveMembers.Visible = false;
    //Trigger client side websockets function sequencially,
    //First user will be Session User
    //Second user will be other gameRoom user
        string posttext = "createSocketConnection('" + currentUser.port +  "');";
           ClientScript.RegisterStartupScript(GetType(),  "Create  Socket  Connection",
        posttext, true);
    }
```

When current user chooses an active user from activeUsers list below code block will be run : Code 5.13 : User selection Code triggers a Javascript Function Form BackEnd C# Code

```
//The selected user will be registered with current User at same Room
 if (drpActiveMembers.SelectedItem.Value != "-99"){
secondUser = Dao.getUserFromPortInfo(drpActiveMembers.SelectedItem.Value.ToString());
    Guid g = Guid.NewGuid();
    string GuidString = Convert.ToBase64String(g.ToByteArray());
    GuidString = GuidString.Replace("=", "");
    GuidString = GuidString.Replace("+", "");
    User currentUser = (User)Session["CurrentUser"];
    Dao.updateUserGameRoom(currentUser, secondUser, GuidString);
    lblStatus.Text = "Currently Connected to " + secondUser.userName;
    drpActiveMembers.Visible = false;
```

```
        //Trigger client side websockets function sequencially,
        //First user will be Session User
        //Second user will be other gameRoom user
        string posttext = "createSocketConnection('" + currentUser.port+"');";
              ClientScript.RegisterStartupScript(GetType(), "Create Socket Connection",
            posttext, true);
        }
```

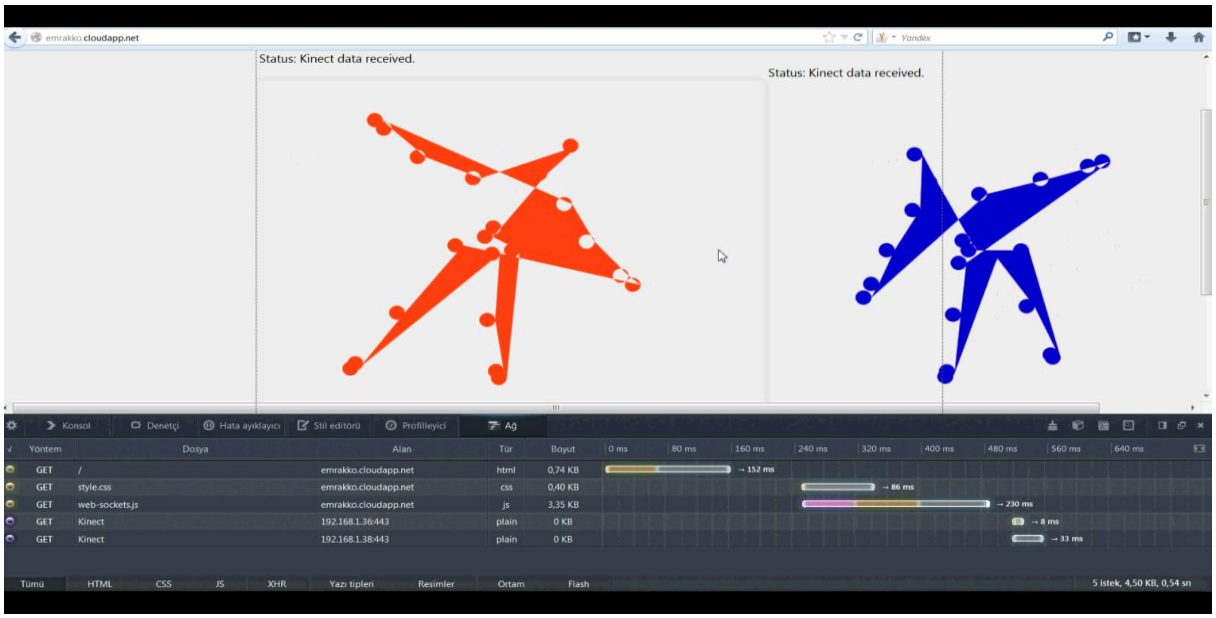
If gameroom has been setted scenario , a javascript will be called form serverside code which sends the port values as parameter to this javascript function. This javascript function opens a web socket connection at client side , and from this port kinect data will be listened and received JSON data will be parsed at this function. And parsed value will be display on HTML5 canvas object. This JSON data has x,y,z coordinate for each joint of the skeleton and according to this coordinates canvas fill method will be called because canvas also use x,y coordinate to fill. This javascript function is like below :

Code 5.14 : Triggered JavaScript Function which creates a Web Socket Client

```
var socket = new WebSocket("ws://" + portValue + "/KinectHtml5");
     socket.onmessage = function (evt) {
  status.innerHTML = "Kinect data received.";
  var jsonObject = eval('(' + evt.data + ')');
  context.clearRect(0, 0, canvasHalfLength, canvas.height);
  context.fillStyle = "#FF0000";
  context.beginPath();
  for (var i = 0; i < jsonObject.skeletons.length; i++) {
    for (var j = 0; j < jsonObject.skeletons[i].joints.length; j++) {
      var joint = jsonObject.skeletons[i].joints[j];
          context.arc(parseFloat(joint.x), parseFloat(joint.y), 10, 0, Math.PI * 2,true);
    }
  context.closePath();
  context.fill();
```

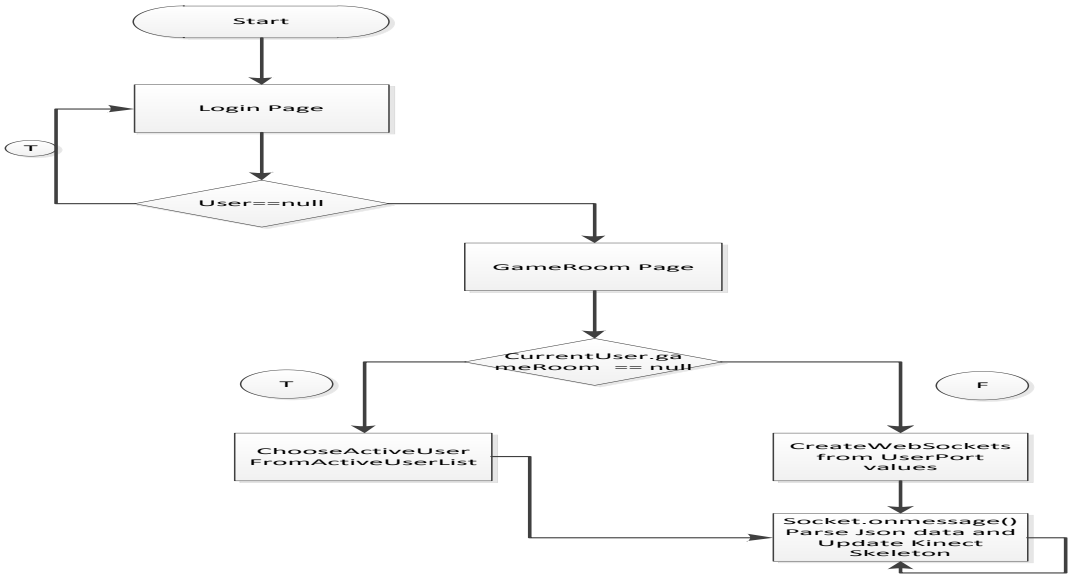
After Web Socket Client has been created data streamng starts for two people. And the screenshoot of the page is like below :

Figure 5.5 : Screen Shot of Client Side application




Kinect Client Side application's flowchart is like below :

Figure 5.6 : Server Side Application Flow Chart Diagram



# 6 EXPERIMENTAL STUDY

In this study lots of techniques has been tried to stream Kinect cameras depth and skeleton data for finding the optimal network solution. Because the main purpose for kind of this applications is, using network bandwith not heavily, transferring the data uninterrupted and transferring only the necessary data not whole of the kinect data.

In the start phase of the study Microsoft Kinect Javascript API has been tried for transferring the Kinect skeleton and depth data. This API has some advantages and disadvantages. This API's advantages are enabling streaming the whole type of the Kinect data such as RGB data , depth data and the skeleton data.,and gives some javascript functions to parse this datas at at client side. The another advantage is it gives an excellent api functions for hand gesture recognition. The Microsoft Kinect Javascript API's disadvantages are, it only supports single Kinect connection and does not transfer the Kinect data over network and also running only on a local Kinect camera. To get better solution this API does not used for our solution.

Another tried framework for transferring Kinect depth and skeleton data over internet was ZigFu Javascript library. This library has some advantages and disadvantages also. The advantages are ZigFu Javascript library supports not only Microsoft Kinect cameras but also supports Asus Depth Camera Xtion and other depth cameras. And this library also has various javasciprt functions to parse and process the depth data. But the main disadvantage was this framework again work only for locally connected depth camera and does not allow remote Kinect camera data processing over network.

In fact, while exploring the Kinect data processing technologies, we find many solutions for processing the Kinect depth and skeleton data at client side , but none of this solutions support remote Kinect camera data processing ,and transferring the depth and skeleton data over internet is not possible, and also all of this technologies only support for a single Kinect camera data processing. So; we created a solutions for processing Kinect skeleton and depth data over internet , and with this solution multiple Kinect cameras data can be shown on a single web page. For bring this solution to maturity we tried many technologies for finding an optimum solution.

## 6.1 Web Service Solution

With this approach for transferring a locally connected kinect cameras data to the internet, the main idea was when the kinect camera produces new data the server application which mentinoed at section 4.1 triggers the web service which is deployed at client side application, and the client side application takes the sended depth and skeleton data and parse this data. After parsing operation the data will be shown on a web page. But, when we implemented this solution we recognized that calling a web service for a real time application like ours application , is mot an optimum solution. Because, web service technology in some cases can treat asynchronous and this disadvantage effected our solution because our solution is a real-time application so the data must be synchronous with the application. Also in some cases we recognized that the data interrupted when we use web service technology to transfer the Kinect camera data. So we did not prefer this technology for transferring a remote Kinect camera data and we preferred the web socket technology for transferring the data over network. Because web socket technology supports synchronized data processing and if any error does not occurred data transferring will be uninterrupted.

## 6.2 Transferring XML Files Solution

After choosing the transferring technology as web-socket technology , another topic was which data type will be streaming. Firstly, we converted the skeleton data and the depth data to the xml files, and this converted data sended to the web page over websocket protocol. But, after transferring the data, we recognized that processing and parsing of an xml file reduces performance of the application too much. And this xml data is 2 times bigger than the data which is converted into the JSON format. When programiing a real time application, developers should think about the performance. In our applcaiton we also should think about the performance, becasuse a kinect camera produces 30 frames per second and while transferring this data over internet we should think about the data sizes. So we did not want to use network bandwidth heavily , we didn’t use the XML files to transfer the Kinect camera data over internet. Another issue for why we did not choose XML files for transferring the data is, the parsing operation of a file at client side with javascipt again reduces the performance of the application.

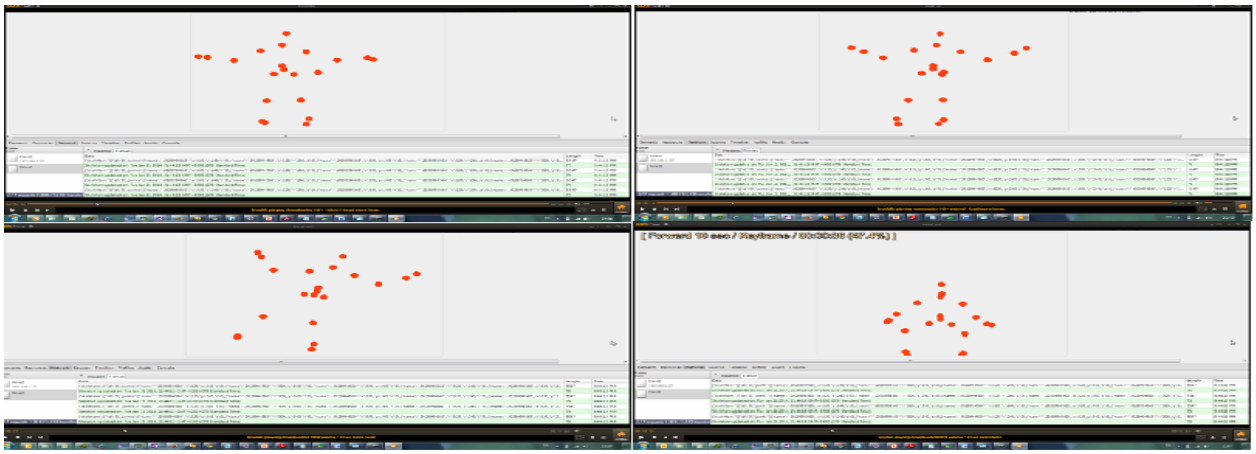
The XML file which is used for transferrind the Kinect skeleton and depth data is shown below. Code 6.1 : Skeleton Data in XML File Format

```
<?xml version="1.0" encoding="utf-8"?>
<joints>
  <joint>
    <id>Head</id>
    <xposition>45.6</xposition>
    <yposition>13.2</yposition>
    <zposition>3.4</zposition>
  </joint>
  <joint>
    <id>Left Arm</id>
    <xposition>43.6</xposition>
    <yposition>14.2</yposition>
    <zposition>3.5</zposition>
  </joint>
  ...
</joints>
```

## 6.3 Test Phase of The Application

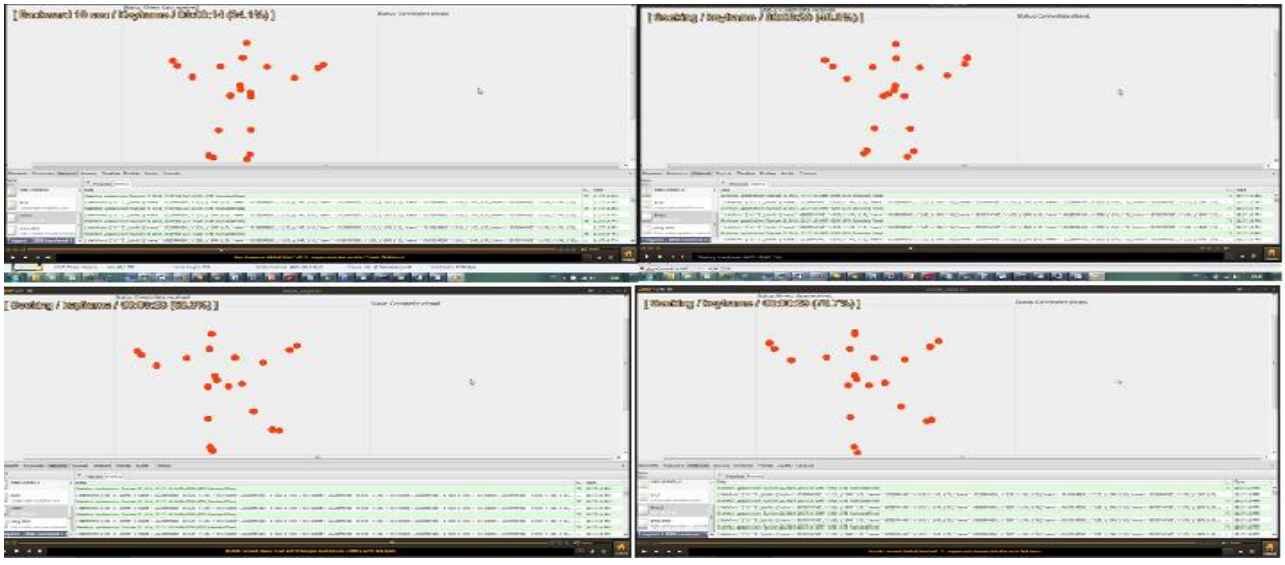
In test phase of our application we firstly tested streaming of locally connected Kinect camera data. In local streaming we observed that there is no network latency occurred on local and the streaming of a locally Kinect camera works stable, and also no data interruption occurred. And also we could track and display the seated skeleton data. Here is some screen shots from locally connected camera tests.

Figure 6.1 : Locally Connected Kinect Camera Data Streming and Displaying



As seen at Figure 6.1 data transmission has been done successfully. After local tests , we tested our application at cloud and at a real network topology. The first test at cloud has done with a single user on a cloud application. Data loss has noot been observed .The streamed data for each second was approximately 5K * 30 fps (Frame per Second) . Total data value was approximately ~150 K for each second with a single person test. The screen shot of remotely connected Kinect camera skeleton data display at Figure 6.2 .

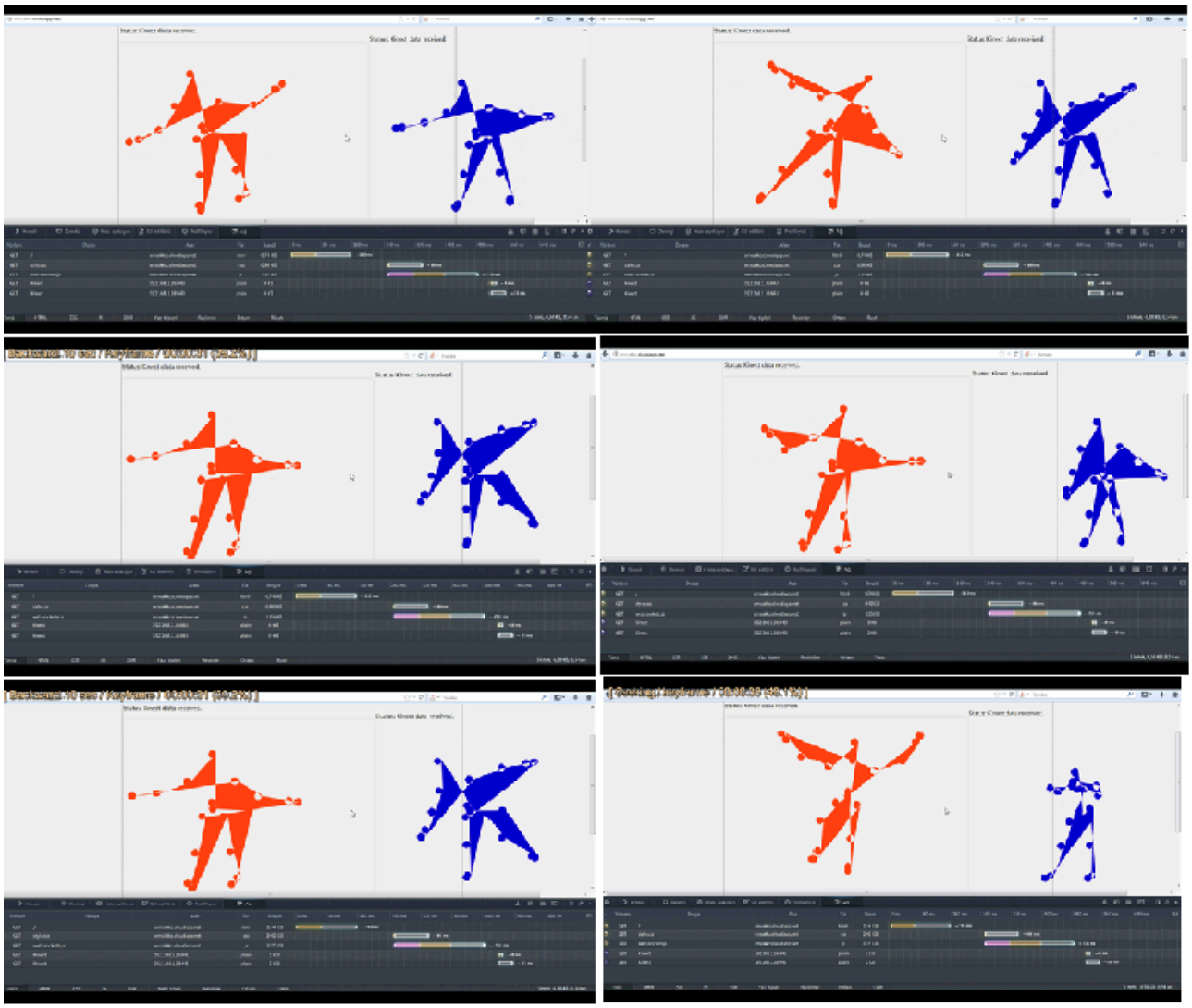
Figure 6.2 : Remotely Connected Kinect Camera with Single User



After remotely connected single user test , we tested the application with two people, each of this user has remotely connected Kinect camera and connects to the same cloud service. And they could

see eah others skeleton data. But; at this test we observed some data losses. For each user the streamed data was approximately 5K * 30 fps (Frame per Second) . Total data value was approximately ~150 K. The people count was 2 so the for each seconf the network traffic was ~300 K. So we observed some data loses. The screen shot of remotely connected Kinect camera skeleton data display for two people at Figure 6.3.

Figure 6.3 : Remotely Connected Kinect Camera with Multiple User



As seen at Figure 6.3 because of the network latency and the data losses the shape of the skeletons deformed.

While we are testing the application , we also observed the WebSocket traffic with a Gooegle Crome browser plugin to check if the sended and received data is correct or not. When we check this data we realized at each phase of the test the data sended successfully (sometimes with small

data losses) to the clients,but sometimes with a network latency. This is the reason of why the shapes displayed deformed on a browser.

When we combine all of this informations and result on a table. The table will be like below : Table 6.1 : Application Test Phases Table

| Kinect Camera Status | People Count | Transferred Data Size per Second | Data Losses Observed ? |
|---|---|---|---|
| Locally Connected | Single | ~150 K Byte | NO |
| Remotely Connected 1 Camera | Single | ~150 K Byte | NO |
| Remotely Connected 2 Camera | Multiple | Between ~290 K and ~300K | YES |

# 7 CONCLUSION & DISCUSSION

With the evolution of web technologies and the depth camera technologies most of the research lab and schools work to combine this kind of technologies to stream and interchange the depth cameras data over internet. Also research companies has found some solutions for this problem. But most of this applications which mentioned before in this study , only stream locally connected depth cameras data. The main idea of this study was to solve this problem and stream not only locally connected depth cameras data but also remote connected depth cameras data over internet synchroneous. To achieve this solution we selected Microsoft Kinect Camera for depth camera and the Web Socket technology for the network data streaming. Because Microsoft Kinect camera works more stable than the other depth cameras, and web socket technology supports the synchroneous data streaming. Also we choosed to transfer the data in JSON data because JSON data is fully supported by the javascript API. After finishing of the study we deployed the application on Microsoft Windows Azure cloud world to test the our thesis is working on real world or not. We also tested our application on mobile devices such as Apple Ipad, and we see that our application works on a cloud server and the remote connected Kinect data can be displayed on web browsers and the mobile browsers which supports the websocket protocol. Today, Most of the modern browsers supports Web Socket protocol.

Finally ;when we think about where we can use this framework, we found many useful usage area. Because of the application is platform independent , the users can use this application at all of browsers which supports the web-socket technology . This property increases the area of usage. For example ;we can use it at remote rehabilitation center project. Most of the people rejects the rehabilitation treatment after injuries, and at some cases the rehabilitation center treatments cost too much for the people. To perevent people from this cases , we can use this application, because we can track all of the skeleton data online. Another future work can be elderly people remote skeleton tracking project. Most of the elderly people is living alone at their home, and in some cases dealing with them couldn't be possible. With this system because of we can stream skeleton data of the people online, at extreme cases we can make an alarm mechanism to notify the elderly people relatives about there is an extreme case at house. Another future project can be ; online dressing room project. Most of the people uses e-markets for buying some clothes without seeing what they buy and how the dress will be seen on her own. With the help of this project people can see the clothes on their own body to have an idea how the dress will look like , and also we have all of the

data to measure the body size , with this data we can make a recommendation about which size of dress to buy. Another one is ; online gaming with skeleton tracking concept. With the help of this project people can play online games with their body movements, for example they can play table tennis oppositely or can play penalty shooting game again oppositely and many other games.

In conclusion , streaming skeleton and the depth data over internet can be used for many industrial project.

## REFERENCES


[1] J. Webb, J.Ashley (2009). Beginning Kinect Programming with Microsoft Kinect SDK. ). SoutheastCon, Proceedings IEEE.

[2] D. J Mickens,L. Zhao,J. Qiu (2010). Gibraltar: Exposing Hardware Devices to Web Pages Using AJAX. San Diego

[3] World Wide Web Consortium (2010). HTML5 Differences from HTML4.

[4] A. Wessels , M.Purvis ,J.Jackson ,S. Rahman (2010). Remote Data Visualization through WebSockets

[5] Retrieved Decemver 9, 2013,http://en.wikipedia.org/wiki/Kinect

[6] Retrieved Decemver 9, 2013,http://msdn.microsoft.com/en-us/library/dn435664.aspx

[7] Retrieved Decemver 9, 2013,http://advertboy.wordpress.com/2011/04/10/silverlight-kinect -apps-of-tomorrow/

[8] Retrieved Decemver 11, 2013, https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5

[9] Retrieved Decemver 10, 2013,http://www.websocket.org/aboutwebsocket.html

[10] Retrieved September 9, 2013,http://msdn.microsoft.com/en-us/library/dn435664.aspx

[11] Retrieved Decemver 7, 2013,http://zigfu.com/en/zdk/javascript/

[12 ] Retrieved Decemver 8, 2013,http://www.html5canvastutorials.com/

[13] Retrieved Decemver 13, 2013,http://www.json.org/

[14] G. Xing, S.Tian1, H.Sun3, W. Liu, H. Liu (2013). People-following System Design for Mobile Robots Using Kinect Sensor , 25th Chinese Control and Decision Conference (CCDC)

[15] A. P. Lanari, M. Hayashibe, P. Poignet (2011) , Joint Angle Estimation in Rehabilitation with Inertial Sensors and its Integration with Kinect

[16] P. H. Shum, S. L. Ho, Y. Jiang,, S. Takagi (2013). Real-Time Posture Reconstruction for Microsoft Kinect

[17] J. Konrad, M. Wang, P.Ishwar (2012). 2D-to-3D Image Conversion by Learning Depth from Examples

[18] G. Tao, P.S. Archambault, M.F. Levin (2013). Evaluation of Kinect Skeletal Tracking in a Virtual Reality Rehabilitation System for Upper Limb Hemiparesis

[19] X. Wang, Q. Ma,W.Wang (2012). Kinect Driven 3D Character Animation Using Semantical Skeleton

[20] J. Fabian, T. Young, P. Jones (2012). Integrating the Microsoft Kinect With Simulink: Real-Time Object Tracking Example

[21] Z. Xiao, F. Mengyin, Y.Yi . L.Ningyi (2012). 3D Human Postures Recognition Using Kinect. 4th International Conference on Intelligent Human-Machine Systems and Cybernetics

[22] H. Mohamedi , T.Val , L. Andrieux, A.Kachouri , Using a Kinect WSN for home monitoring: principle, network and application evaluation

[23] Y. Gu, H. Do, Y.Ou, W. Sheng (2012). Human Gesture Recognition through a Kinect Sensor. International Conference on Robotics and Biomimetics

[24] A. L.Mendez, M.Alcoverro, M. Pardas, J.R. Casas (2011). Real-Time Upper Body Tracking with Online Initialization using a Range Sensor. IEEE International Conference on Computer Vision Workshops

[25] J. Fu, D. Miao, W.Yu, S.Wang, Y.Lu, S.Li (2013). Kinect-Like Depth Data Compression.