Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 1 – No.5, April 2015 – www.caeaccess.org

Distance Physical Rehabilitation System Framework with Multi-Kinect Motion Captured Data

Mohammad Rafiuzzaman
Department of Computer and Information Engineering
Sakarya University
Esentepe Campus, 54040, Adapazari, Turkey

Cemil Öz, Ph.D.
Department of Computer Engineering
Sakarya University
Esentepe Campus, 54040, Adapazari,

ABSTRACT
Visiting physical therapists to the clinics for physical rehabilitation in regular basis is a very long and time-consuming trip where the final result for success is truly hard to see in daily training. That's why technological development in traditional physical rehabilitation system is both important, interesting and its effects on patients' time-management process is huge. In this paper we have proposed a framework for distance physical rehabilitation system using motion captured data from multiple Kinects which can interact directly with the patients, even grasp and track their movements so as to send those data back to the doctors in clinics using Windows Azure. Its goal is to coach patients through their physical therapy exercises and make those exercises a more enjoyable experience and bring physical therapy alive for them at their homes, the same way doctors interact with them in clinics.

General Terms
Motion Captured Data, Movement Recognition.

Keywords
Distance rehabilitation, Multiple Kinects, posture recognition, posture database, PCA, skeleton tracking, Windows Azure.

1. INTRODUCTION
With the advent of range video cameras that capture depth at video frame rates, like ToF 3D range cameras [24] or the Kinect sensor [18], now-a-days it becomes possible to observe deforming human postures over time and to model these deformations accordingly. In the past few years much work investigated traditional dynamic shape techniques such as active motion capture systems using markers [6], or marker-less multi-view shape from silhouette for human motion capture and much has already been achieved regarding posture recognizing and modeling [5]. Improved range and novel imaging technology is now facilitating and significantly influencing these kind of modeling. A recent example is the very successful combination of the Microsoft Kinect sensor with the human motion capture system [19].

According to National Spinal Cord Injury Statistical Center in the United States, approximately 265,000 people have spinal cord injuries which results in partial or full paral-ysis [13]. Some clinical studies have demon-strated that some motor deficiencies after these kinds of spinal cord injuries can be at least partially or fully recovered through physical rehabilitations. But conventional rehabilitation training programs typically require professional therapists to supervise the patients' extensive repetitive range-of-motion and coordination exercises and to assess the progress. Even in the automated motion capture systems, until recently there has been the use of active or passive markers predominantly in the study of human movement where IR cameras and markers are placed on the subject. Though these systems are accurate, but often are very expensive and impractical to move. Before each capturing session, active or passive markers must be correctly placed on the body. Therefore, such systems are only suitable for laboratory settings. Wearable sensors systems are more suitable for ambulatory measurements in home settings as they are small, lightweight, mobile and less expensive [14], [15]. But these sensors must be placed correctly and securely [16], and must account for gravity, noise and signal drift [5]. Moreover, they require changes to the daily routine of the subject in the form of sanitary treatment, charging batteries and uploading data.

For addressing these kinds of problems, new rehabilitation tools based on Virtual Reality (VR) should emerge in the physical therapy arena, which led us to develop a system for virtually monitoring physical rehabilitation programs such as some common physical exercises for patients having SCI or other physical disabilities with the help of Microsoft Kinect. The core idea behind our VR-based rehabilitation technique is to use sensing device like Kinect to capture and quantitatively assess the movements of patients under treatment to track their progress more accurately and easily. We have also tried to make this program as a remote system with the help of Windows Azure [21] so that the doctors can monitor the progress of physical rehabilitation program of a patient from their clinics, while the patients can perform their physical exercises in their homes with an ease.

The rest of the paper is organized as this; we first review some related works in virtual motion sensing technology in Section 2. In Section 3 we explained our proposed approach towards developing our remote physical rehabilitation system. We then explain how we can transform our rehabilitation system to a remote application in Section 4. Finally, in Section 5, we conclude the paper with some discussions, limitations and future research directions of our research work.

2. RELATED WORKS
Creating a virtual rehabilitation system where IT specialists are free to imagine and set their fantasies free to invent new absorbing ideas helpful in this enormous world of problems, has expanded rapidly over the past decade. Several pros and cons of advances in vision-based human motion capture system have been shown in [11]. Virtual environment have been also useful for dismounted soldier training using multiple Kinects, for showing the capability of low cost, non-obtrusive solutions for soldier training [9]. Though a solution for fusing skeletal data from multiple Kinects to provide complete coverage of a user was successfully demonstrated in there; but the condition of having a sign attached with the body to indicate that the person is facing forward the sensor in

their Orientation Based Data Fusion system was definitely a drawback as it limits a person's movement. Kinect has also been used to analyze human gait in [8] but they have not used a complete physical interaction system as they have used in-shoe pressure sensor and a gyroscope which also limits free movements. In [12] the authors also tried to come up with a virtual rehabilitation technique using Kinect, but they have showed a comparison between the OptiTrack optical system and Kinect and in the end they have shown the superiority of the OptiTrack optical system over Kinect which is not true in all the cases as described in our framework.

In order to reduce the probability of missing some degree of freedom (DOF) due to occlusion and noises while capturing normal movements, [3] proposed to use a motion database almost similar to our database system to get the best matched posture from them. But as most of the other approaches they missed to get the whole body rotation of the user because of the use of a single Kinect sensor. As a result, the database posture matching process becomes inaccurate as the number of joints recognized by Kinect is dropped significantly. This is one of the reasons that we have used four different Kinects to capture human posture in our proposed framework.

The accuracy and robustness of Kinect based rehabilitation technique have been tested on [7], where they have presented the problems which occur due to noise and occlusion while capturing data with Kinect. In order to overcome this problem we have proposed to use a custom made posture database (PDB) with which we can replace the missing information from the data that we have captured. Also with the help default skeleton tracking mode as well as seated skeleton tracking mode we hope to recognize the accuracy of the body tracking to great extent as the tracking in seating position is especially relevant in the rehabilitation context as patients may be bounded to a wheelchair or disable to stand-up.

3. EXPERIMENTAL PROCEDURE

For developing our remote VR-based physical rehabilitation system we have come up with the following steps:

3.1 Capturing color images from the Kinect cameras

3.2 Capturing the depth data

3.3 Tracking the human skeleton

3.4 Recognizing different physical exercises

3.5 Developing the system using multiple Kinects

All of these steps are discussed in details in the followings:

3.1. Capturing color image from the Kinect camera

The first step towards our motion capture system is the acquisition of color image stream from the Kinect. An image stream is nothing but a succession of still image frames. Normally Kinect can deliver the still image frame within a range of 12 to 30 frames per second (fps) [4].

In order to get any color frame from the sensor we have used the Event Model [22] of Kinect sensor. Using the Event Model, the Kinect sensor sends the frame to the application whenever a new frame is captured by the sensor. For doing that we have subscribed to the specific event handler where the incoming frames are needed to be processed. But before subscribing to the event handler, we have fixed the color type and resolution of image streams that we are looking for. Once it's subscribed to, the sensor will send the data continuously unless the channel is disabled, unsubscribed or the sensor is stopped. In short capturing the color image stream from the sensor and displaying it to the UI can be done by following these steps:

3.1.1 Enabling the color stream channel with an image format

3.1.2 Attaching the event handler with the stream channel

3.1.3 Processing the incoming image frames

3.1.4 Rendering the image frames on the UI

3.1.5 Monitoring the sensor status

3.1.6 Performance enhancement – Memory allocation

The basic operations of these steps in depicted in figure 1 while their discussions are given below in details:

3.1.1 Enabling the color stream channel with an image format

When the sensor is running and the color stream is enabled, it will initialize the Kinect sensor to generate a stream of color images. After that we are going to use an overloaded method that accepts Color Image Formats as an argument.

3.1.2 Attaching the event handler with the stream channel

After enabling the color stream channel, the sensor has to be informed about what to do when it has captured a new image frame. To achieve this, we have fixed an event handler that has to be attached to the sensor stream channel.

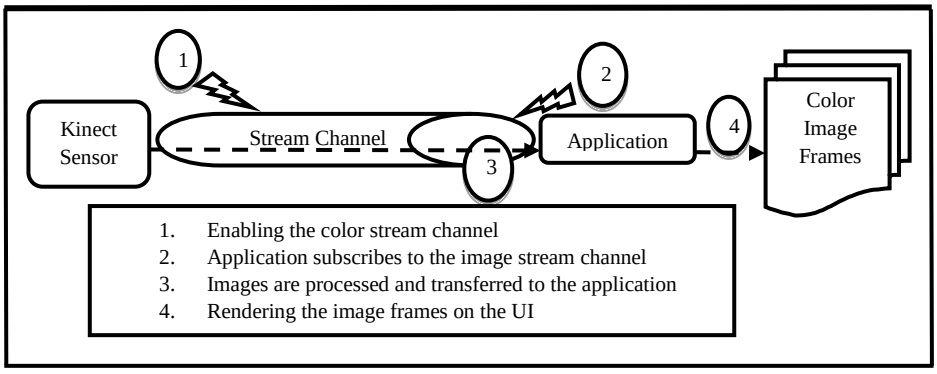

Fig 1: Capturing color image from the Kinect camera

3.1.3 Processing the incoming image frames

Once the event handler is called, it means there is a new frame that has been sent by the sensor and it is time to process it. Whenever there is a new frame sent by the sensor, this event handler in the:

--First step, reads the image frame.

--Second step, we have performed a null detection on the incoming image frame. This is just to make sure that if there is any dropped frame in between frames, so that the application can take care of it.

--Third Step, we have calculated the size of the incoming frame.

--Fourth Step, we have created a bitmap image source with the incoming image frame and assigned the same in the image control.

All of this is for a single image frame; a similar succession of similar image frame operations produces the image stream.

3.1.4 Rendering the image frames on the UI

After this we will display the frames in the UI. While we were creating Bitmap Source from the image frame, most of the rendering part within the event handler were already been done.

3.1.5 Monitoring the sensor status

Of course as a common job for all the applications, starting the sensor before the sensor starts producing image frames and stopping and monitoring the sensor status will also be a common job for us.

3.1.6 Performance enhancement – Memory allocation

The image frames are stored into a buffer before they are used by the application. Normally the image processing in Kinect happens up to 30 frames per second [4]. This means that memory allocation and clean-up is happening around 30 times per second. This makes performance trail, as if there is any delay in reading the buffer data and rendering it as images, the buffer will fill with a new image frame by discarding the old frame data. The unprocessed frames will be dropped, which means the image frames will be lost, and the frame rate will be decreased.

To solve this and make our application perform better, we have used the alternative way where we have allocated the memory for the image frames at once and updated only the pixel data on frame change. In this way we improved the performance by reducing the memory consumption as well as memory allocation and de-allocation.

3.2. Capturing the depth data

After capturing the color frames from Kinect camera we are going to capture the depth data in a similar fashion as capturing the color frames. However, the working principle of the depth sensor and information returned by the depth sensor are totally different than that of the color camera. Each color frame consists of numbers of pixel values, which give the values of red, green, and blue color components; whereas each pixel's information in the depth data represents the distance of an object from the sensor.

Depth information data will be captured using General Stereo Triangulation [10] system where two images from IR depth sensor and IR laser are used to obtain the two different views on a scene and combined to get one view similar as human binocular vision.

After that we are going to calculate the distance from the depth data. The Kinect sensor returns 16-bit raw depth frame data [4]. The first three bits represent the identified person and the remaining 13 bits give the measured distance in millimeters. So from the 16-bit data we will perform a bitwise shift operation(>>) to move the bits to the upper 13 bits to get the distance as shown in figure 2.

3.3. Tracking the human skeleton

When we talk about how to build an application that interacts with human body motion, first of all we need to capture the information about the users standing in front of the Kinect sensor, and from there skeleton tracking comes into picture. Our complete skeleton-tracking feature is built on the depth data processing, internal machine learning and color vision algorithms. Figure 3 shows the overall process flow diagram for capturing skeleton data, while the following steps explain them:

3.3.1 Depth data is processed in the rendering pipeline process and matches with the database labeled data (discussed later) and generates the inferred body segments.

3.3.2 Once all parts are identified based on the labeled data, the sensor identifies the body joints.

3.3.3 The sensor then calculates the 3D view from the top, front, and the left of the found joints.

3.3.4 Then the sensor starts tracking the human skeleton and body movement based on the proposed joint points and the 3D view.

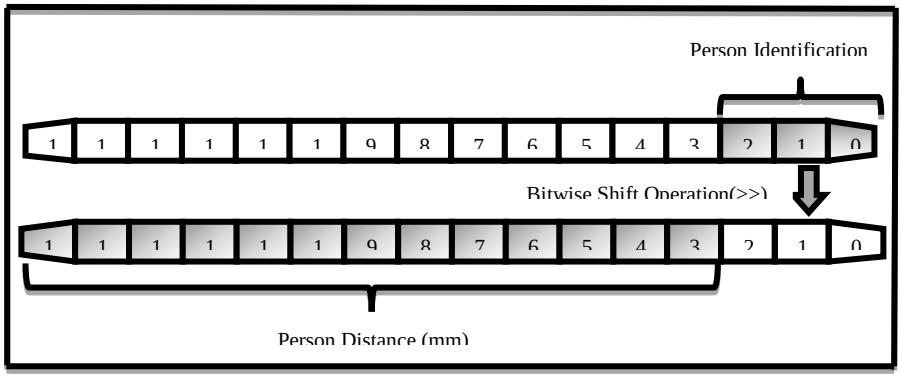

Fig 2: Calculating user identification and distance from the sensor using depth data

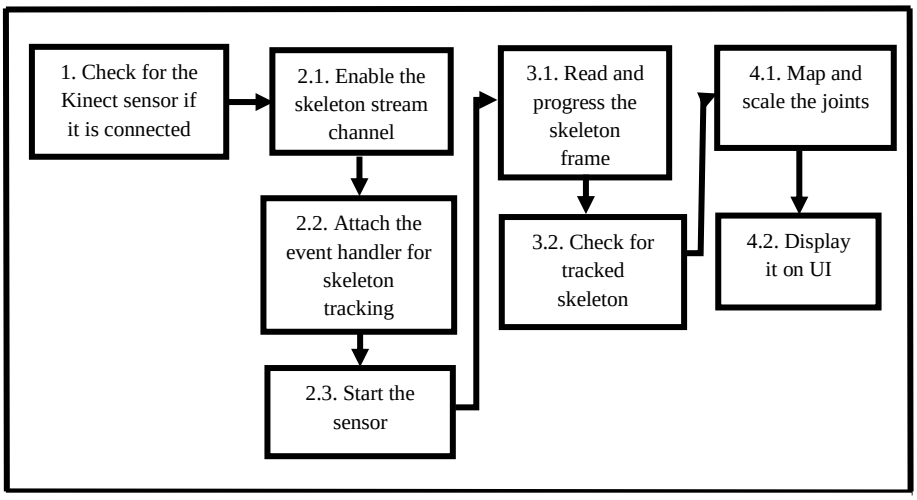

Fig 3: Steps to track the human skeleton

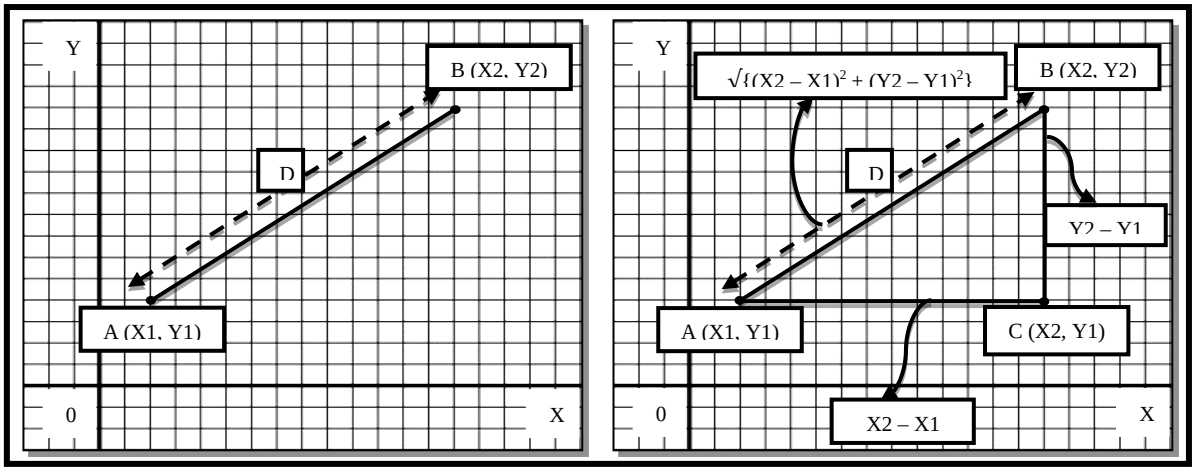

Fig 4: Measuring the distance between two joints

All of these steps are considered for Default-Skeleton tracking mode. We have also considered Seated-Skeleton tracking mode where we have marked all the lower body parts' joints, including Hip Center and Spine as Not Tracked. Seated-Skeleton tracking mode has been used for patients who are bounded to wheelchairs or having other disabilities to stand up.

3.3.5 Performance enhancement - Skeleton smoothing

Skeleton joint movements are not that smooth and because of processing large amounts of data over a period of time, during skeleton tracking there comes some jitters. This shaky movement of skeleton data does not provide a good end user experience and we need to overcome this problem to enable a rich user experience.

In order to do that, we have set the smoothing parameters [27] while setting up the skeleton stream data. The smoothing parameters like, Correction, Smoothing, JitterRadius. MaxDeviation Radius, Prediction [20] solves the jittering problem by filtering the skeleton data and applying a smoothing algorithm to it.

Holt double exponential smoothing [26] procedure has been also used with smoothing parameters to reduce the jitters from skeletal joint data. In order to make a forecast, the exponential smoothing is applied to a series of time-based data. The skeleton engine returns the skeleton frame in a regular time interval. The smoothing algorithm applies to each set of data and calculates a moving average based on the previous set of data. It uses the values passed by the smoothing parameters, during the calculation of moving average.

3.4. Recognizing different physical exercises

After tracking the human skeleton as discussed in the previous section we will start to recognize different human movements or postures so that our system will be able to recognize physical exercises done by the patients. For this purpose we have divided our total system into two different sections, where one section will be used for basic posture recognition system for recognizing basic physical exercises using linear algebraic calculations and the other section is going to be used for advanced posture recognition system to recognize advance physical exercises by constructing posture database and matching them with the postures captured from Kinect.

3.4.1 Basic posture recognition system

The core idea behind our basic posture recognition system is to calculate skeleton's joint points as shown in the previous section and apply basic logic to recognize some basic postures. This kind of recognition depends on some pre-

Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 1 – No.5, April 2015 – www.caeaccess.org

defined set of conditions, known as the result set. If the performed action is matched with the result set, we can say that the user has performed a certain posture, otherwise not.

In the coordinate system of our system each skeleton joint is measured in the three-dimensional (X, Y, Z) plane. X and Y coordinates indicate the joint location in the plane, and Z coordinate indicates how far the joint is from the sensor. The X axis of the joint will change if the joints move from the right hand side to the left hand side or vice versa. Similarly, the value of the Y axis will change for moving joints in the upwards or downwards direction. Forward or backwards moving of the joints from the sensor will be reflected by the Changes in the Z axis.

In our proposed framework calculations for the basic posture recognition is done by:

3.4.1.1 Measuring the distance between different joints.

3.4.1.2 Measuring the joints' positions and the deviation between the joints' positions.

3.4.1.1 Measuring the distance between different joints

As depicted in figure 4, we have used Pythagorean Theorem [15] to calculate the distance between two joints. Here the value of "D", the distance between points A and B, will be the hypotenuse of the right-angled triangle that was formed by the points A, B, and C which can be calculated using following formula:

$$(\text{Distance of } D)^2 = (\text{Distance of } A, C)^2 + (\text{Distance of } B, C)^2 \quad (1)$$

$$\therefore D = \sqrt{(\{(X2 - X1)^2 + (Y2 - Y1)^2\})} \quad (2)$$

This same formula works well for three dimensional planes as the distance between points (X1, Y1, Z1) and (X2, Y2, Z2) can be calculated by the following formula:

$$D = \sqrt{(\{(X2 - X1)^2 + (Y2 - Y1)^2 + (Z2 - Z1)^2\})} \quad (2)$$

Using this simple formula we can recognize some simple physical exercises like Virtual Rope Workout or Hand Swiping as depicted in figure 5 by calculating the distance between two points.

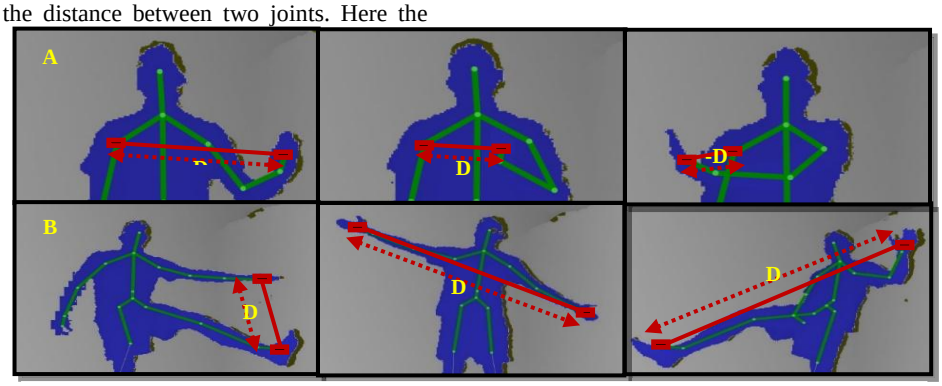

Fig 5: Detecting Virtual Hand Swiping exercise and B. Rope Workout

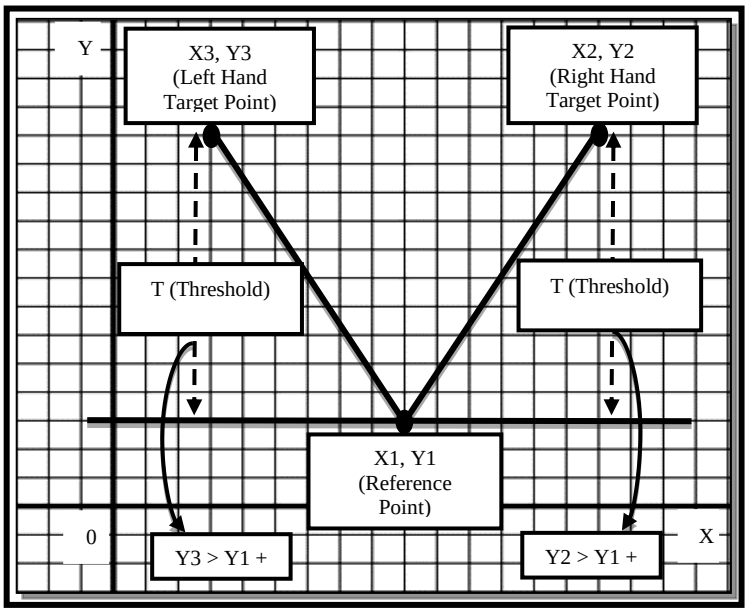

Fig 6: Calculating the joints' positions and the deviation between the joints' positions

3.4.1.2 *Measuring the joints' positions and the deviation between the joints' positions*

Though we can recognize a lot of simple physical exercises in our basic posture recognition system with the preceding procedure, but to recognize some other exercises which need physical movements with respect to some fixed body parts, we need to make some further calculations. For example Hand Raising exercise where either both hands or only left or right hand of a person needs to be raised above his/her head with respect to the head joint.

In this case, the posture recognition of our system involves three different joints; namely, the head, right hand, and left hand. We cannot recognize these gestures by just calculating distances between these three joints; rather, we need to measure the joint positions with respect to the coordinate plane.

Suppose we want to detect if both the hands are raised above the head. On the coordinate plane, movement of the hands in either to upwards or to downwards direction will be based on the Y axis. As a reference point for the other two hand joint positions, we have considered the Y axis of the head joint position as the reference point. The other two hand joint positions have been considered as the target points. Then we have compared the values of both the targeted joints' Y axis movement with respect to the reference points. Once the targeted value crosses the reference point, we say that the posture is identified and the exercise has been done. A threshold value (T) with reference points have been also used to make sure that the targeted joints are crossing as we have expected. The calculation process is depicted in the figure 6 and an example of hand rising exercise is depicted in figure 7.

3.4.2 *Advanced posture recognition system*

Though we can recognize several exercises done by the patients with our basic posture recognition system, we also need advanced posture recognition system to recognize physical exercises involving more complex skeleton joints' movements like Depth Squad Exercise from standing and a Walk and Run Exercise on a gateway treadmill etc. In this recognition system, some predefined postures are first recorded and stored and while matching them, the same set of user actions are taken as input parameters and validated against the stored data. The final result is driven by calculating the sum of squared differences between the existing posture data set and the posture data set that is currently being performed. Overall, the advanced posture recognition system involves the following steps:

3.4.2.1 Constructing the Posture Database (PDB),

3.4.2.2 Comparing Kinect postures with PDB postures and

3.4.2.3 Generating a resultant set of estimated postures.

3.4.2.1 Posture Database construction

Our Posture Database (PDB) has been constructed with postures captured from a commercial optical motion capturing system. But in order to reduce the computational complexity we have removed the global rotation along the vertical axis and global 3D translation of the postures for normalization.

In our proposed PDB we are going to represent each posture by a set of joint positions D(P). For removing the similar postures and thus increasing the efficiency of run-time PDB searches, we have calculated the sum of squared differences of joint positions and if the difference among them is smaller than our predefined threshold values then we are going to remove them.

Postures which should be recorder and stored in our PDB depend on the desired application of the system. We are going to construct our PDB with only those postures of exercises that the users i.e. patients are expected to perform. In this way our proposed system is also going to reduce the space complexity.

3.4.2.2 Posture comparison and detection

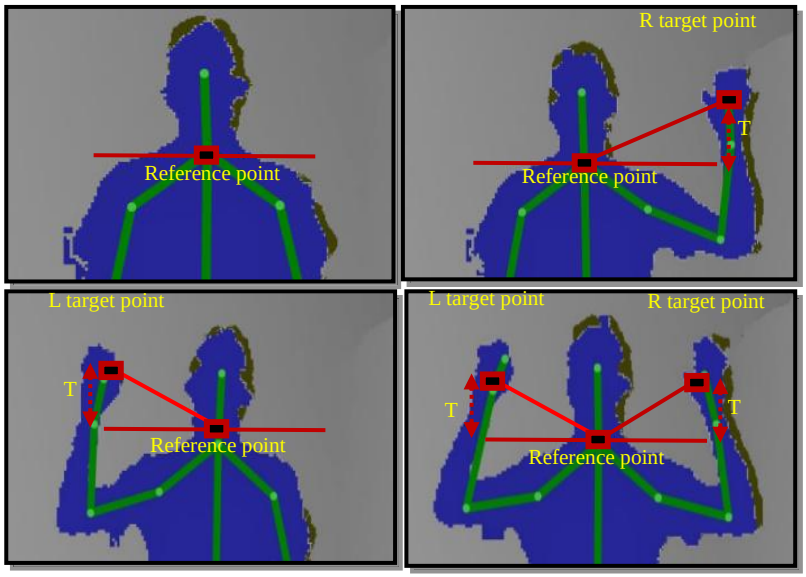

Fig 7: Hand raising exercise.

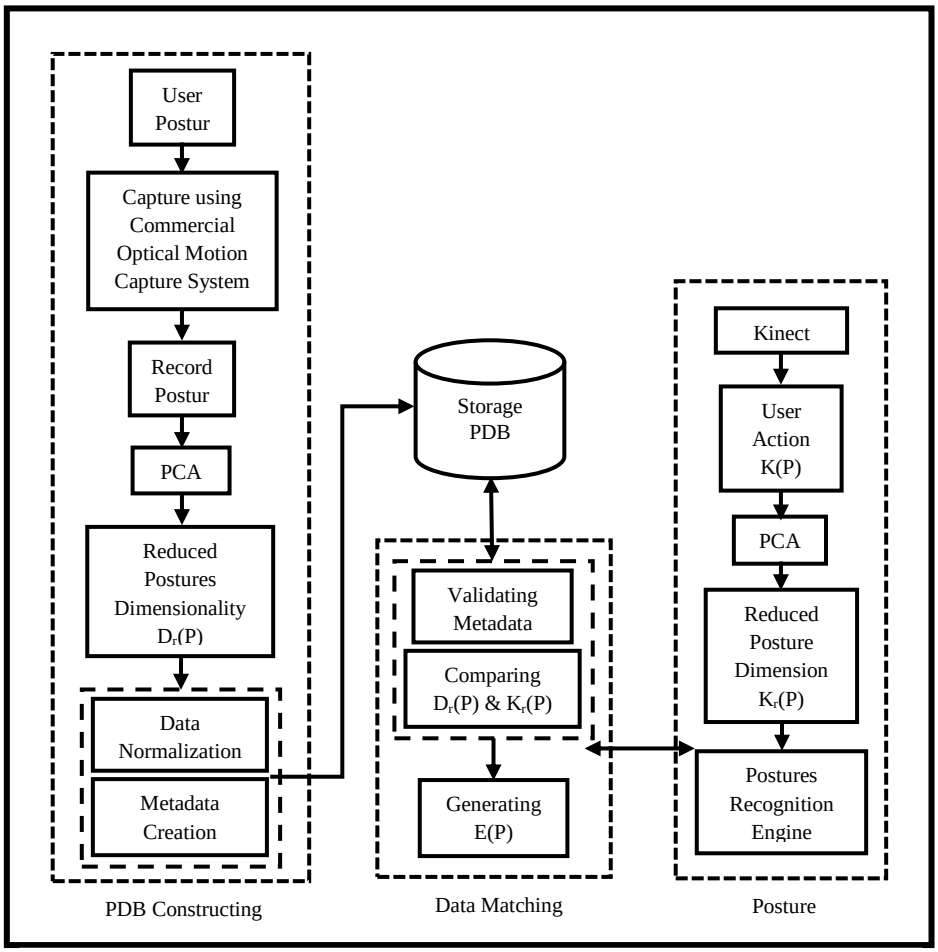

Fig 8: PDB Construction, Posture comparison and detection

We are going to denote the joint positions obtained during run-time from Kinect as K(P). After retrieving it, from the PDB we are going to retrieve the best matched posture D(P) according to K(P). Posture detection and posture comparison works side-by-side. While the user is actually performing the action, data can be matched multiple times to see if it is matching correctly. As data storage, we have proposed to use XML file system.

3.4.2.3 Generating the resultant set of estimated postures

As there exists an intrinsic redundancy among the joints in the postures, if we are going to consider D(P) and K(P) as two point clouds and start the motion comparison [1] among them then it is going to reduce the computational speed. In order to overcome this complexity in our framework we have proposed to reduce the posture dimensionality by using PCA [2] and thus improve the performance of the database queries.

As a result the postures in PDB, D(P) is going to be reduced into Dr(P) and during run-time the postures K(P) captured from Kinect will be reduced into Kr(P) by the projection matrix calculated by PCA. After that we are going to conduct a search among Dr(P) and Kr(P) and find out the most similar postures among them by calculating their sum of squared differences. In this way we are going to find out our resultant set of estimated postures E(P).

During run-time, if a joint is missing in the posture obtained from the Kinect due to occlusion, noise or limited capture volume, we are going to consider that joint position to be the mean value of that same joint from all the same postures in the PDB and compose E(P) accordingly. Figure 8 shows all of these operation needed posture database construction, posture comparison and posture detection.

3.4.3 Constructing the posture recognition engine

After retrieving the postures performed by the user we need to validate if the postures are performed correctly or not. For this purpose we have constructed a posture recognition engine with the following steps:

a) Start

b) Posture validation

c) Predefined rules

d) Finish

As depicted in figure 9, to start with recognizing any posture, there will always be an initial position, we call it the "start" position. This is the entry point for any posture and has to be validated before validating other positions.

Once the start position is validated and the posture for any kind of exercise is being performed by the end user, every single frame has to be validated under the predefined "rules" for the particular posture types by matching them with the PDB. In our system we have proposed to set up these rules by the physical therapists with some predefined exercises for physical therapy.

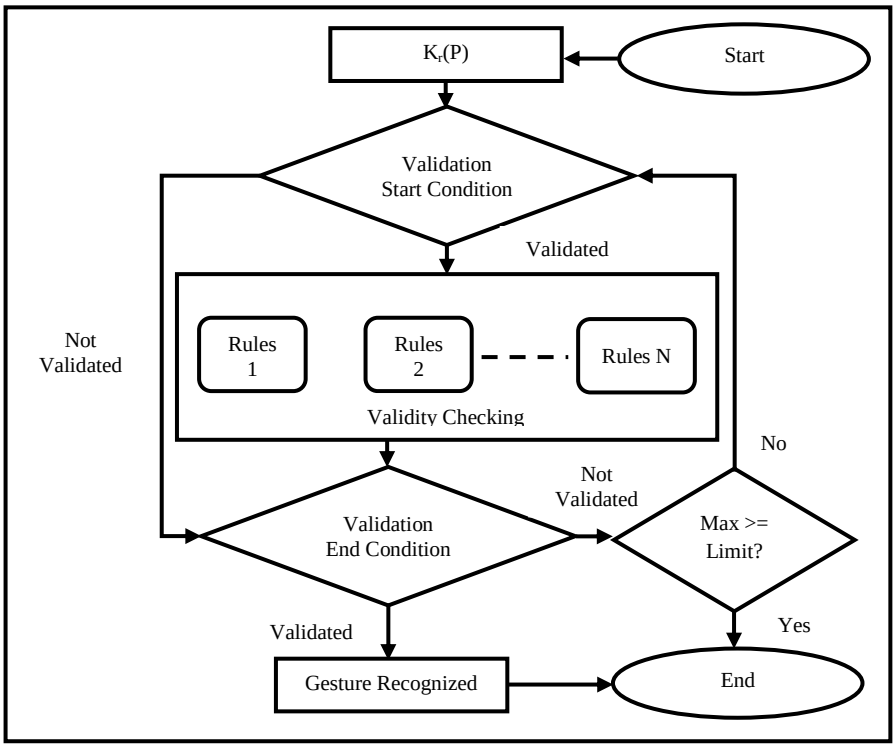

Fig 9: Posture recognition engine

If any of these rules fail to satisfy during the complete execution cycle, we will stop the posture tracking and wait for it to start again.

Finally, there is a condition that triggers the end of the posture and "validates" the final position, which indicates that posture recognition is finished. We have also put a limitation Max for the number of repeat and try of exercises to avoid an infinite loop. This number can be set by the patients in their user interface of the system.

e) Performance enhancement - Making posture recognition engine more flexible

To make the posture recognition engine perform softer we further break down the complete predefined rules block into multiple smaller modules, and call it the "phases". Each phase of a posture will have its own result set that measure the success or failure of that phase. The result of the phases will be dependent on each other, which means recognition will move to the next phase if the previous phase result was passed.

It may occur that all the rules in a phase are not satisfied. But this does not always mean that the phase has failed, instead it could be that the user is "on hold" or "in progress" on that particular position for some time. In these kind of scenarios we mark the state of those phases as pause and wait for the next action for a few frames. These phases can communicate with one another using Inter Phase Communication, to share the information, result, and data with each other.

3.5. Developing the system using multiple Kinects

As depicted in figure 10, in order to remove the restriction for the user to face forward to the sensor like traditional Kinect based motion capturing systems and to see a user from most of the available angels, we have proposed to use four Kinect sensors in our system.

Another reason for using multiple Kinects in our system is failover. Failover is used to make the system more fault-tolerant by providing automatic switching to a redundant or standby system. If we consider Kinect as a system then it could happen, that one system fails (power turns off or gets disconnected) and the application fails to capture data. In such a situation, we can start the other connected sensors automatically to capture data and turn it off once the first device starts again. The other reasons for using multiple Kinects for our system are:

a) Capturing data from a specific sensor.

b) Giving full freedom of orientation of the user within the training area.

c) To reduce the percentage of failed joint estimation as Kinect works best when faced frontally.

Four camera tripods are used with four Kinect sensors to provide stability and adjustability for different person's height and pointing angle for finding out the precise direction of the user. The working procedure for handling multiple Kinects are as follows:

3.5.1 Using multiple USB Controllers:

Kinect for Windows SDK supports at most four Kinect sensors to plug into a single system [4]. But Kinect sensors consume a good amount of bandwidth of the USB port; hence, more than one Kinect can't be operated by a single USB Controller. While we were working with multiple Kinects, we have used external USB Controllers to uniquely handle each individual Kinects to overcome such scenarios.

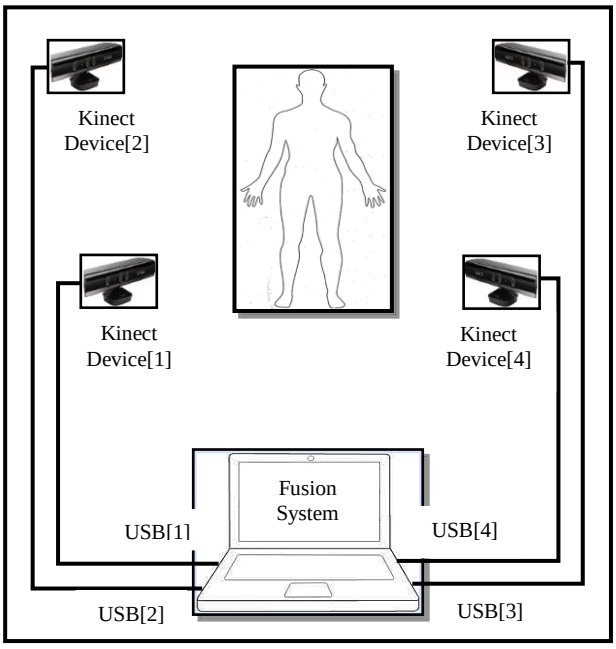

Fig 10: Full system orientation using multiple Kinects

3.5.2 Reducing interference:

In order to handle multiple Kinects we have to handle interference. We know that Kinect measures the depth data by reading the IR patterns projected by an IR emitter [4]. When there are multiple sensors placed in the same area the projected IR from the multiple sensors can interfere with one another where Kinect sensors return incorrect data, as the IR laser is not modulated.

To overcome such scenarios we have used the Shake 'n' Sense technology [17]. Using Shake 'n' Sense, we can just shake the Kinect sensors a little bit so that the position of the IR dots is moved. As a result the IR points are moved a bit and the sensor can read the data on every move. Generally this shake is done by an external motor.

3.5.3 Accessing individual sensors

Each individual sensor has been identified by the position index of the sensors. For example the index starts with 0, which indicates the first device. Some basic information such as device ID, status, and connection ID from the individual Kinect sensors also have been used in order to access them.

3.5.4 Capturing data using multiple Kinects

After connecting the sensors properly and accessing them individually we will start to capture data from them. Almost similar as capturing color and depth data stream from a sensor as discussed in the previous sections, we will follow the following steps for capturing data from multiple Kinects:

a) Identifying the individual Kinect sensors.

b) Attaching the event handler to the individual Kinect sensors.

c) Handling the events for the attached event handler.

d) Checking the sensor status and controlling the start and stop based on the requirements.

We have iterated through the collection of devices and added the required information into our custom collection. The overall implementation has been same as we did for individual Kinect sensors, except identifying and attaching an event handler for each particular sensor. A fusion system is used to combine all the data received from multiple Kinects in order to get a single combined scenario.

4. RESULT PROCESSING: MAKING THE APPLICATION A REMOTE SYSTEM

The final aim of our proposed approach is to send the results of the exercises done by the patients with our proposed physical rehabilitation system to the cloud so that the doctors from the hospitals can monitor them. In this section we are going to discuss how we have designed our system for this purpose so that it can upload done physical exercises of the patients in a video format to the cloud using Windows Azure Media Services [23].

Windows Azure Media Services is a scalable media platform for distributing content to any screen, on any network. It offers a collection of components and technologies from Microsoft and third parties to enable end-to-end media solutions, leveraging on Windows Azure platform [23]. In order to connect our system to the Windows Azure and transfer the physical exercises of the patients to the hospitals in a video format we have used the following steps:

4.1 Media (video) creation

4.2 Media (video) processing

4.3 Media (video) delivery

4.4 Media (video) consume

These steps are shown in figure 11 where the descriptions are given below:

4.1. Media (video) creation

After getting the exercises done by the patients from our fusion system we are going to convert them into video files in order to send them to the cloud using Windows Azure.

4.2. Media (video) processing

Video makes up a large amount of internet traffic. So uploading the videos and watching them with a good speed is a problem that on-demand video applications often face. With some collection of components [23] in media processing as shown in figure 11 Windows Azure Media Services solves this problem. Those components are given below with their functions in our system:

4.2.1 Media ingest

4.2.2 Encoding

4.2.3 Content protection

4.2.4 Media streaming

4.2.1 Ingestion

This allows our system to upload content into the cloud. Two things have to be concerned in this component:

--Speed: Video content for the recognized physical exercises might be very large. In order to handle this large volume data as well as maintain a good speed for video streaming we have used HTTPS Delivery for ingestion in our system.

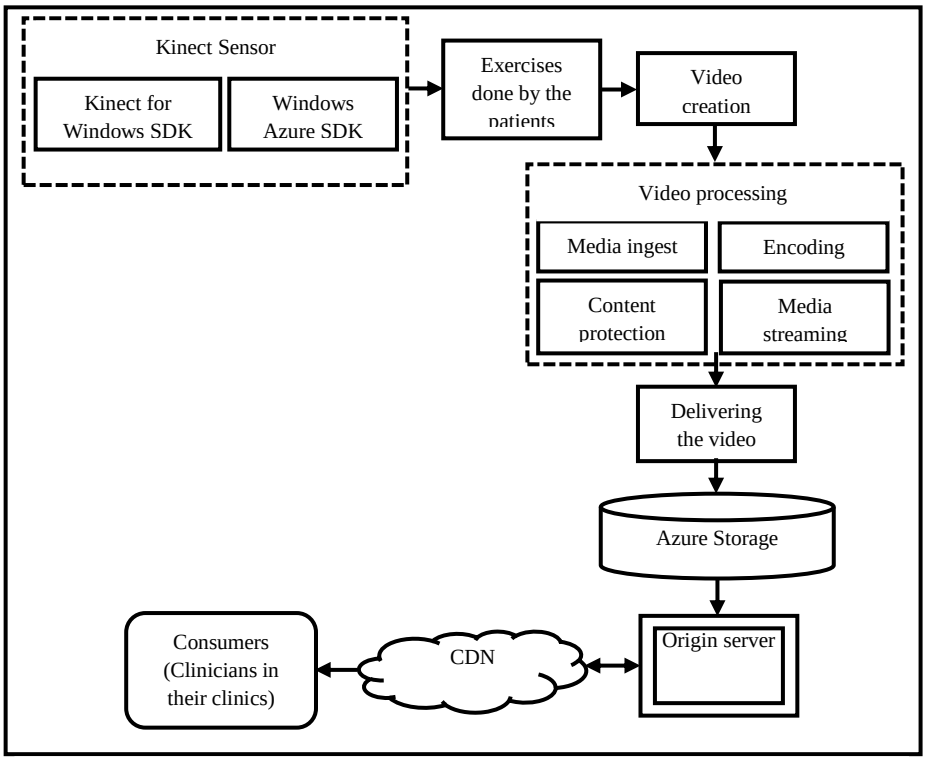

Fig 11: Using Widows Azure Media Service to send our physical rehabilitation system data to a remote location.

--Security: AES 256bit CTR mode has been used for encryption and HTTPS for all the communication. With this we make sure that our video contents are securely delivered to the Blob storage [25].

4.2.2 Encoding

This scheme allows us to compress Video into different formats. In our system we have picked up the brain of Microsoft Expression Encoder. But instead of using this desktop application, we have used Windows Azure Media Encoder in the cloud.

4.2.3 Content protection

It provides DRM (Digital Right Management) solution for content protection. For our remote system we have used MPEG Common Encryption for the content protection.

4.2.4 Media streaming

For on-demand video streaming we have uploaded your encoded content into Origin Server [23] which supports .Mp4, Smooth Streaming and HLS.

4.3. Media (video) delivery

After processing the videos, they are delivered to the Azure cloud storage. After that the Origin Service processes the outbound stream from storage to CDN (Content Delivery Network). This component consists of:

a) *Content* server: Pulls content from storage and delivers it to CDN.

b) *Caching*: Reduces load on Channels and Storage.

c) *Content* encryption: Keeps content protected.

d) *Dynamic packaging*: Stores media in one multi-bitrate format (MP4 for our system) and converts to the format requested by the clients i.e. doctors from the hospitals.

For communicating with Azure Storage we have used the REST API [25] as it allows us to transfer content over a REST protocol without worrying about the platform.

4.4. Media (video) consume:

After delivering the media it can be consumed by the receivers i.e. doctors from the hospitals. Since both HLS and Smooth Streaming are transmitted through Http with small fragments, it is easier to utilize CDN (Content Delivery Network) for scaling delivery of video content worldwide which we have used in our system.

5. DISCUSSION AND CONCLUSION

The outcome of our proposed system in this paper is believed to be a methodology for better assessment of the functional outcome of physical rehabilitation techniques for functionally disabled patients. It could also have a direct impact on the way future interventions are planned and performed. Continuation of the study on this project could, over time, improve the way these operations are carried out; the effect on rehabilitation time, the time spent in hospital, the improved range of stable motions and positional accuracy and measurement. Though the implementation with the measurement of accuracy and robustness of our proposed system is yet to be done, we believe the remote physical rehabilitation application like the one we have presented in this paper, in the longer term will provide a large amount of improvement regarding the effect of the relative and absolute function of the SCI problems, especially on the joint stability, relative and absolute displacements. Our proposed new remote physical rehabilitation system will also potentially

offer better functionality and easy accessibility, due to reduced complexity, thus potentially reducing the cost of physical therapy systems to the hospitals. Accurate virtual gait analysis, another possible function of our proposed system, also results in better assessment of the functional outcome of such operations and can be used against national benchmarks in order to inform on ways to reduce costs and rehabilitation times.

6. REFERENCES

[1] L. Kovar, M. Gleicher, and F. H. Pighin. Motion graphs.ACM Trans. Graph., 21(3):473–482, 2002.

[2] Lindsay I Smith. A tutorial on Principal Components Analysis. February 26, 2002.

[3] Hubert P. H. Shum, Edmond S. L. Ho. Real-time Physical Modelling of Character Movements with Microsoft Kinect. VRST'12, December 10–12, 2012 ACM 978-1-4503-1469-5/12/12.

[4] Microsoft Corporation. Kinect for windows SDK programming guide version 1.5. 2012.

[5] H.J. Luinge and P.H. Veltink. Measuring orientation of human body segments using miniature gyroscopes and accelerometers. Medical and Biological Engineering and Computing, 43(2):273–282, 2005.

[6] C.-C. Yang and Y.-L. Hsu. A review of accelerometry-based wearable motion detectors for physical activity monitoring. Sensors, 10(8):7772–7788, 2010.

[7] StepanObdrzalek, GregorijKurillo, FerdaOfli, RuzenaBajcsy, Edmund Seto, Holly Jimison and Michael Pavel. Accuracy and Robustness of Kinect Pose Estimation in the Context of Coaching of Elderly Population. National Science Foundation (NSF) grant 1111965 and by Grant Number HHS 90TR0003/01, 2012.

[8] Moshe Gabel, Ran Gilad-Bachrach, Erin Renshaw and Assaf Schuster. Full Body Gait Analysis with Kinect. The Department of Computer Science, Technion – Israel Institute of Technology.

[9] Brian M. Williamson and Dr. Joseph J. LaViola Jr. Multi-Kinect Tracking for Dismounted Soldier Training. Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2012.

[10] SB Pollard, JP Frisby. Transparency and the uniqueness constraint in human and computer stereo vision. Nature 347, 553 - 556 (11 October 1990); doi:10.1038/347553a0

[11] ChanjiraSinthanayothin, NonlapasWongwaen, WisarutBholsithi, Skeleton Tracking using Kinect Sensor & Displaying in 3D Virtual Scene, International Journal of Advancements in Computing Technology(IJACT) Volume4, Number11, June 2012.

[12] Chien-Yen Chang, Belinda Lange, Mi Zhang, Sebastian Koenig, Phil Requejo, NoomSomboon, Alexander A. Sawchuk, and Albert A. Rizzo. Towards Pervasive Physical Rehabilitation Using Microsoft Kinect. 2012 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops.

[13] National Spinal Cord Injury Statistical Center. Spinalcord injury facts and figures at a glance. Birmingham, Alabama, February 2011.

[14] K.J. O'Donovan, B.R. Greene, D. McGrath, R. O'Neill, A. Burns, and B. Caulfield. SHIMMER: A new tool for temporal gait analysis. In EMBC, pages 3826–3829, 2009.

[15] Eli Maor. The Pythagorean Theorem: a 4,000-year history. 2007 - books.google.com

[16] J. J. Kavanagh and H. B. Menz. Accelerometry: a technique for quantifying movement patterns during walking. Gait & Posture, 28(1):1–15, 2008.

[17] DA Butler, S Izadi, O Hilliges, D Molyneaux, S. Hodges, D. Kim. Shake'n'sense: reducing interference for overlapping structured light depth cameras. CHI '12 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Pages 1933-1936. ACM New York, NY, USA ©2012. ISBN: 978-1-4503-1015-4 doi>10.1145/2207676.2208335

[18] Microsoft, "Kinect," 2010. URL: http://www.xbox.com/en-us/kinect (accessed March 7, 2012).

[19] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," Sensors, vol. 12, no. 2, pp. 1437–1454, 2012.

[20] Denis, L. Tupin, F. ; Darbon, J. ; Sigelle, M. Joint Filtering of SAR Interferometric and Amplitude Data in Urban Areas by TV Minimization. Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International (Volume: 5). 7-11 July 2008

[21] Blogs.technet.com. 2010-02-01. "Windows Azure General Availability - The Official Microsoft Blog - Site Home - TechNet Blogs" Retrieved 28-05-2013.

[22] Carlos F. Crispim-Junior, Baptiste Fosty, Rim Romdhane, Qiao Ma, Francois Bremond, Monique Thonnat. Combining Multiple Sensors for Event Recognition of Older People. MIIRH'13, October 22, 2013, Barcelona, Spain.

[23] Jie Li ; Humphrey, M. ; References. eScience in the cloud: A MODIS satellite data reprojection and reduction pipeline in the Windows Azure platform. Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on. 19-23 April 2010

[24] Reinhard Koch, BogumilBartczak, AnatolFrick, BogumilBartczak, AnatolFrick, FalkoKellner and Ingo Schiller. Time-of-Flight-Range and Color Camera Systems for 3D-TV and Augmented Reality Applications. Institute of Computer Science Christian-Albrechts-University of Kie.

[25] Chappell, David. Introducing Windows Azure. Microsoft. October 2008.

[26] Prajakta S. Kalekar. Time series forecasting using Holt-Winters Exponential Smoothing. www.it.iitb.ac.in/~praj/acads/seminar/04329008_ExponentialSmoothing.pdf December 6, 2004.

[27] Prof. Dr. W.Toporowski, Institutfür Marketing & Handel Abteilung Handel. Smoothing methods.