Tracking the Activities of Daily Lives: An Integrated Approach

Roanna Lun*,† Connor Gordon*, Wenbing Zhao*
*Department of Electrical Engineering and Computer Science
Cleveland State University, Cleveland, Ohio 44115
†Glodon USA, 440 North Wolfe Road, Sunnyvale, CA 94085

***Abstract*—The tracking of the activities of daily living (ADL) may have significant implications in healthcare because it would enable healthcare professionals to receive updates *remotely* regarding the functional status of post-injury and post-surgery patients, and people with disabilities and the elderly. If successful, technologies that enable ADL tracking could dramatically reduce the healthcare cost because people could stay at the comfort of their home instead of at the healthcare facilities. In this paper, we present the design and implementation of a system that integrates two modalities of human motion tracking: computer vision and inertial sensing. For the computer vision modality, we choose to use the Microsoft Kinect sensor due to its low-cost and excellent programming support. For the inertial sensing modality, we choose to use smart watches for the same reason. The Kinect sensor is responsible for indoor ADL tracking, while the smart-watch component is responsible for outdoor ADL tracking. The smart-watch component also facilitates user-identification with the Kinect sensor and user's health related activities information with indoor computing component, as well delivers realtime feedbacks to the users.**

Keywords*—*Activity of Daily Living; Microsoft Kinect; Wearable Sensors; XML; Data Logging and Visualization

I. Introduction

The tracking of the activities of daily living (ADL) may have significant implications in healthcare because it would enable healthcare professionals to receive updates *remotely* regarding the functional status of post-injury and post-surgery patients, and people with disabilities and the elderly [1], [2], [3]. If successful, technologies that enable ADL tracking could dramatically reduce the healthcare cost because people could stay at the comfort of their home instead of at the healthcare facilities. Led by fitness trackers such as Fitbit, Jawbone, and Microsoft Band, inertial-sensor based ADL tracking has been adopted by millions of people [4], [5], [6]. While such trackers work quite well on tracking sleep patterns, steps, and level of activities, they are inadequate to track specific activities that are important for daily living, such as rehabilitation exercises (for post-injury and post-surgery patients) [7], [8], [9], lifting and pulling activities (applicable for everyone, which is important to reduce the risk of back injuries of nurses) [10], [11], [12], and fall detection (highly relevant for elderlies living alone in their homes) [13]. The computer-vision based sensors could complement ADL by tracking these activities far more reliably. Hence, it calls for an integrated approach in ADL tracking including:

- In outdoor environment, fitness trackers or similar inertial-sensor based wearable devices are used to track a user's ADL.
- In the indoor environment, computer-vision based sensors can be installed and programed to track the user ADL with a focus on the specific activities that have to be recorded accurately.

In this paper, we present the design and implementation of a system that integrates motion sensing modalities for ADL tracking. For the computer vision modality, we choose the Microsoft Kinect sensor due to its low-cost and excellent open source programming support. For the inertial sensing modality, we choose the smart watches for the same reason. Smart watches are all equipped with accelerometers, and most are additionally equipped with gyroscopes and magnetometers. The smart-watch component also facilitates user-identification with the Kinect sensor and user's health related activities information with indoor computing component, as well delivers realtime feedbacks to the users.

Driven by the need for ADL tracking, our system is designed to be open so that it can communicate over the network with different systems and mobile/wearable devices other than just receiving sensing data from Kinect sensors. The capability of integrating with other devices and systems makes it possible to continuously track each individual's ADL in the indoor environment.

Kinect has been widely used to track human activities in many different applications [14]. Most of them are standalone applications. Some do have networking capability to stream the Kinect sensing data to a client such as a Web browser, or to log activities at a remote server. However, these applications typically rely on manual input to identify who are tracked. While this limitation might not be a problem for game playing and rehabilitation/fitness exercises, but it adds undue burden to users in the applications that are intended for ADL tracking. The integration of computer-vision and inertial sensing devices solves this problem: we can uniquely identify each individual automatically by the uniqueness of wearable device/app id and a specific gesture defined to allow a user to register with our system, and at meantime be able to track activities and provide realtime feedback. The usability of this system is further improved by using a non-intrusive form of biometrics [15].

We focus on the design and implementing of the Kinect sensor because inertial-sensor based ADL tracking is widely available commercially and the health-related data can be obtained via mobile health Apps such as Google Health and Apple Health Kit. The major contributions of system are:

- Technical details on how to integrate the two dras-

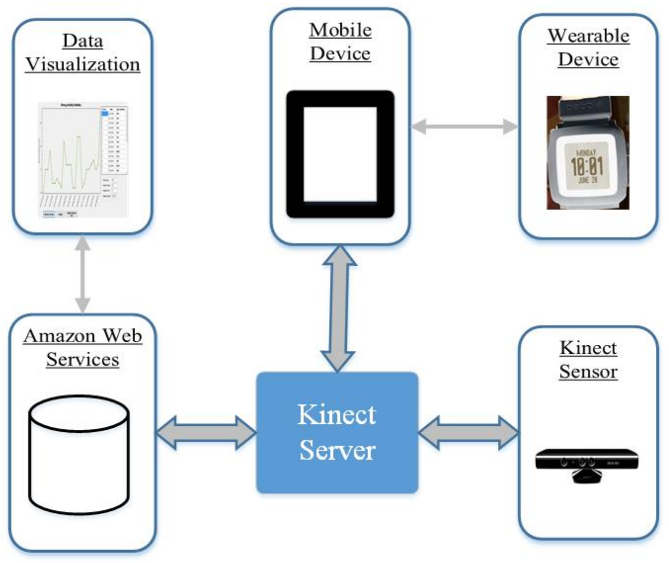


Fig. 1. Main components of our system

tically different motion sensing devices together for ADL tracking.

- An system design for human activity tracking that is not tied to any specific activity or gesture via the use of configuration files encoded using the eXtensible Markup Language (XML). The system can perform the detection of any specific activities provided that the corresponding rules are defined in the configuration following our specification.
- A cloud-based logging mechanism to store user activities in the cloud.
- A visualization interface for reviewing and analyzing user activity data.

II. Overview of the System

In this section, we provide an overview of our system. The main components of our system are shown in Figure 1. We first describe the main hardware components used in our system. We then present the software components of our system. We conclude this section by presenting the demonstration interface.

A. Hardware Components

The system consists of the following hardware components:

- Wearable devices, such as smart watches. Each user is supposed to wear such a device on one of his/her wrists. The wearable device is used to register a user with the Kinect server, and to provide haptic feedback to the user.
- Microsoft Kinect sensors. At least one Kinect sensor is used to track the activity of up to two (if Kinect v1 is used) or four (if Kinect v2 is used) consented workers at a time. More sensors can be used to enable multi-angle tracking of the same worker, and/or to cover a large area that is beyond the view range of a single sensor.
- One or more computers. The computer runs Kinect server. It also connects with Kinect sensors (via USB cables) and wearable devices (wirelessly and bridged by smart phones). If Kinect v1 is used, a single computer could drive up to four Kinect sensors. If Kinect v2 is used, a single computer can drive only a single Kinect sensor. When multiple computers are used, they communicate with each other to share their states.
- One or more smart phones. A smart phone is used to bridge the communication between a corresponding wearable device (and hence a user) and the Kinect server. A wearable device communicates with the smart phone using Bluetooth 4.0 (or referred to as the Bluetooth LE), and the smart phone communicates with the Kinect server using 802.11 wifi. The smart phones are placed close to the Kinect sensor that tracks the corresponding user's activity. It is also essential that the smart phone's wifi connection be kept permanently on to ensure an open communication channel between the wearable device and the Kinect server.

Our system does not dictate the number of Kinect sensors (other than that at least one must be used) to be used and how a Kinect sensor (or sensors) should be placed in a room. Such needs will have to be determined based on the actual deployment requirements. To cover a large area, we recommend to use multiple Kinect sensors that have sufficient overlap views. The view overlap helps ensure a smoother transition across different Kinect sensors using our biometrics-based single sign-on mechanism.

B. Software Components

In our current implementation of the system, we choose to use the Pebble smart watch as the wearable device due to its low-cost and high compatibility with various versions of Android and iOS smart phones. We are in the process of porting the code to Android wear based smart watches. Our system consists of three types of software components:

- A Pebble watch application (WatchApp for short) running on the Pebble smart watch. This application provides an interface for a user to register with the system, to receive alerts (haptic and display based), to check for his/her performance statistics, and to set options (such as how the statistics should be displayed). Figure 2 (bottom leftmost) shows the main interface that is displayed when the WatchApp is launched. The WatchApp is implemented in the C programming language with slightly over 300 lines of code.
- A Kinect server running on the computer. This server subscribes to the skeleton stream for each connected Kinect sensor. This application implements all core logic of our system, including user registration, multi-Kinect support, single sign-on, registered user activity tracking, alert generation, user activity logging. The

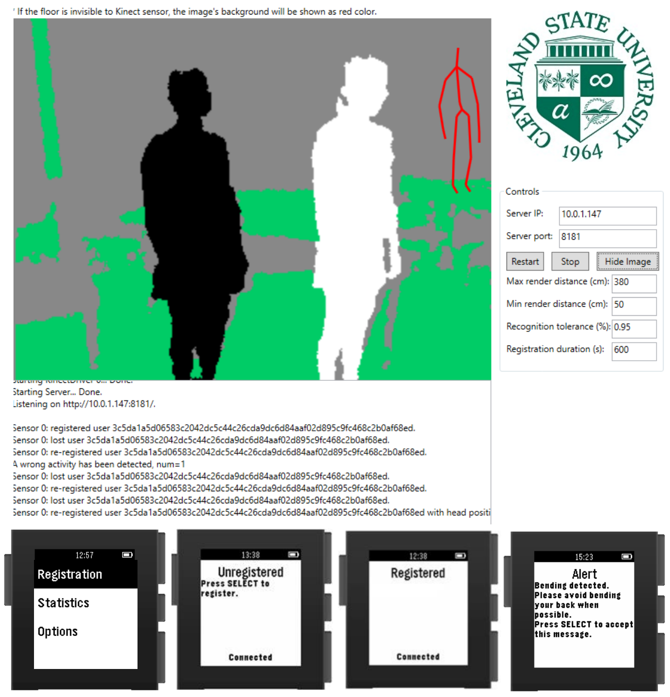
Kinect server is based on Microsoft Kinect Software Development Kit version 1.8 for Kinect v1. Its source code consists of about 20 C# files and an XML configuration file with a total of about 2,000 lines of code. The Kinect server maintains all the system states, including biometric data for the single sign-on mechanism, and user activity data for compliance tracking. All the data are indexed by the unique user id. To facilitate cloud-based logging, a TCP connection is established from the Kinect server to a designed cloud server provided by Amazon Web Services (AWS) [16].

- A mobile application. More accurately, it is the Pebble mobile app running a JavaScript component that we supply. This component contains a single file named pebble-js-app.js. It is used to mediate the communication between the WatchApp and the Kinect application. It is created as part of the WatchApp, and is deployed to the Pebble mobile app running at the smart phone. This JavaScript file (with about 1,000 lines of code) contains code to handle an AppMessage sent by the WatchApp, to invoke on the Web Service application program interface provided by the Kinect application, and send an AppMessage to the WatchApp. The response for the Web Services invocation (XMLHttpRequest) contains a JSON object, *i.e.,* a special type of XML document containing a set of name-value pairs.

The Kinect server has a number of components:

- *Server* and *ServerTask*. They are in charge of handling HTTP communications. The *Server* component encapsulate the HTTP listener for the Kinect server. It waits for incoming HTTP requests. For each HTTP request, a new *ServerTask* object is created to handle the HTTP request. We expect the HTTP request to contain a JSON object. The corresponding response is also encapsulated in a JSON object transported over HTTP.
- *KinectDriver*. This component handles motion sensing data from the Kinect sensor. It handles user registration request with the help of the *Activity Recognizer*. On successful registration of a user, the user is added to the data structure maintained by the *User* object. Furthermore, for each registered user, *KinectDriver* monitors the activities the user engages in, again with the help of the *Activity Recognition*. The specific gesture for registration and activities to be flagged as wrong are defined in the Configuration file.
- *Kinect Coordinator*. This components maintains a list of *User*s and a list of *KinectDriver*. It handles the registration and activity requests by accessing the *User* and *KinectDriver* lists.

To facilitate selective tracking, it is inevitable to define a particular gesture so that the Kinect server could associate the correct skeleton with the user who is registering. However, it is cumbersome to require the user to perform the registration gesture every time the user wants to be tracked. In fact, it is simply impractical to expect a user to use our system with


If the floor is invisible to Kinect sensor, the image's background will be shown as red color.

Controls
Server IP: 10.0.1.147
Server port: 8181
Max render distance (cm): 380
Min render distance (cm): 50
Recognition tolerance (%): 0.95
Registration duration (s): 600

```
Starting Server... Done.
Listening on http://10.0.1.147:8181/.

Sensor 0: registered user 3c5da1a5d06583c2042dc5c44c26cda9dc6d84aaf02d895c9fc468c2b0af68ed.
Sensor 0: lost user 3c5da1a5d06583c2042dc5c44c26cda9dc6d84aaf02d895c9fc468c2b0af68ed.
Sensor 0: re-registered user 3c5da1a5d06583c2042dc5c44c26cda9dc6d84aaf02d895c9fc468c2b0af68ed.
A wrong activity has been detected, num=1
Sensor 0: lost user 3c5da1a5d06583c2042dc5c44c26cda9dc6d84aaf02d895c9fc468c2b0af68ed.
Sensor 0: re-registered user 3c5da1a5d06583c2042dc5c44c26cda9dc6d84aaf02d895c9fc468c2b0af68ed.
Sensor 0: lost user 3c5da1a5d06583c2042dc5c44c26cda9dc6d84aaf02d895c9fc468c2b0af68ed.
Sensor 0: re-registered user 3c5da1a5d06583c2042dc5c44c26cda9dc6d84aaf02d895c9fc468c2b0af68ed with head positi
```

Press SELECT to register.

Bending detected. Please avoid bending your back when possible. Press SELECT to accept this message.

Fig. 2. Demonstration user interfaces on the Kinect server and on the WatchApp

such a requirement because the user-skeleton association can be broken often due to movements (*i.e.,* when the user moves out of the view of the Kinect sensor), or occlusion by other users or obstacles. To improve the usability of the system, it is essential to use additional mechanisms to perform automatic user identification after the user has been registered with our system. We choose to use a non-intrusive form of biometrics using body geometrics [17]. Right after a user has registered, our system records a set of biometrics for the user. When the user is unidentified (*i.e.,* when the skeleton-user association is broken), new incoming skeleton frames are compared with the recorded biometrics. The matching algorithm could be based on simple distance comparison, or based on machine learning models such as support vector machines [18].

We require that the built-in notification be disabled on the smart watch because otherwise the Pebble smart watch would receive all sorts of notifications from the smart phone that are irrelevant to non-compliance tracking, such as notifications generated by email and other applications running on the smart phone. Too many notifications would not be conducive for the user to pay attention to non-compliance alert notifications.

C. Demonstration Interface

The demonstration user interfaces for the system include one for the Kinect server and another for the watch app are shown in Figure 2. A non-consented (and hence, non-tracked) subject is rendered as a white silhouette, and a consented and registered subject is rendered as a black silhouette. The use of silhouettes is to protect the user's privacy. Four snapshots of the watch app are included in Figure 2, from left to right, the initial menu, a window for user registration, a window displayed after the user has been registered, and a window displaying an alert when a non-compliance event (*i.e.,* bending) is detected. Silhouetting of non-consented subjects in our system is achieved by pixel-level manipulation of each color frame based on information contained in the corresponding depth frame. The green background color indicates that the floor can be clearly seen by the Kinect sensor. The visibility of the floor is important because the activity detection is based on the floor clip plane parameters. When the floor is not visible, the background would be colored red to remind the user that the Kinect setup is not right. Although not a primary research objective, this pixel-manipulation technique can be used to facilitate privacy-aware video recording.

III. Kinect Server Operations

The Kinect device provides human full body motion and joint tracking with skeleton image. The communication of HTTP input contains requests for user identification and activities status. A user can send a request to register or query the identification status via WatchApp to Kinect server. The status of user's activities can also be sent to wearable device upon a request. All these requests are initiated either originally by the user via the WatchApp, or on behalf of the user. The WatchApp has a unique identifier that is an id attribute included in every request. The Kinect server uses this attribute to identify the user. A TCP connection is managed by Kinect server to transfer the logging data to a cloud-based database in AWS, where an open sourced mySQL database is utilized in our system. The connection string includes server address, TCP port number, database name, and login credentials.

The registration is a process that employs multi-modality synchronization to identify and track a consented worker. It involves a specific integrated program between wearable device, mobile app and Kinect sensor. First, a consented worker initiates a request by pushing a designated button in the wearable device, which is transmitted to mobile app via blue-tooth, and then to Kinect server application via wireless network. Upon receiving the request, Kinect server implements recognition by a non-natural gesture, the hand is pointing upward with the upper arm and forearm forming a L-shape. Once the registration gesture is derived, the consented worker is registered in the system and a notification is sent back to user via same communication channel.

Non-compliant activities are detected realtime by using invariance conditions for proper movements, such as bending and bending-and-twisting. The Kinect server reports information regarding non-compliant activities the user has engaged. If one is detected, a notification is generated and sent to the WatchApp, which will produce a vibration on the wearable device(e.g. Pebble watch) to alert the user.

The Kinect server is driven by the two types of inputs: (1) skeleton frames from Kinect sensors, and (2) HTTP requests issued by wearable devices (often mediated by smart phones). In this section, we present the detailed design and implementation of the mechanisms that handle and integrate these inputs.

A. Handling of Kinect Skeleton Frames

The Kinect server maintains a communication process with Kinect sensors through (1) initiating an device instance for each connected Kinect sensor; (2) receiving skeleton frames from sensor; (3) recording user's biometrics; and (4) tracking active user's status. The programming flow control for Kinect input handling is shown in Figure 3.

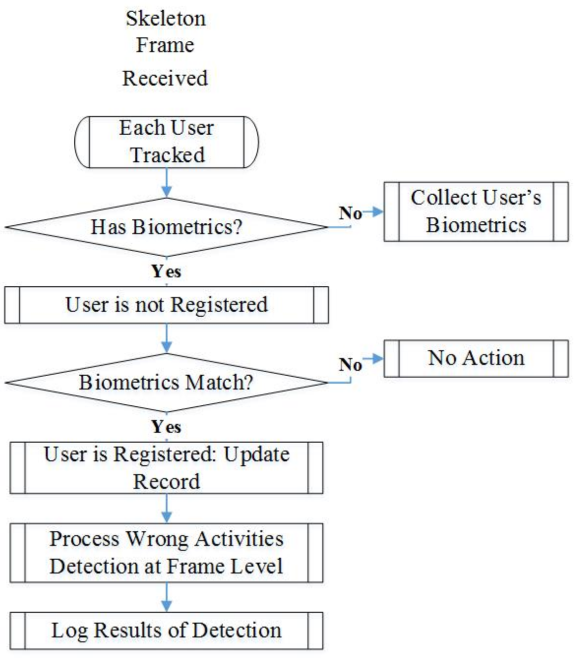


Fig. 3. The flow control for user identification and activity recognition based on Kinect skeleton frames

The *KinectDriver* class that is an object instance for each Kinect sensor connected to server is created by system during initialization process. Hence the object is able to receive image frames from Kinect device using Kinect SDK programming interface. The image frames provided by Kinect SDK program include color frames, depth frames and skeleton frames. In our system, we are particularly interested in skeleton frames. On receiving a new skeleton frame, the Kinect runtime dispatches the frame to a designated thread for processing. The *KinectDriver* first makes a copy of the skeletons received, and then examines each user who is currently tracking. If a user is registered and his/her biometrics recording has not been completed, the *KinectDriver* adds this user to the list. Subsequently for registered user, the average values of the biometrics are stored for further processing, not just a single frame value. Using average values instead of a single frame for computing user's biometric information can increase the accuracy of recognition.

If the user is registered and biometrics has already been taken, the *KinectDriver* examines skeleton with all other users who are not currently tracked to see whether their biometrics are matched or not besides current registered and actively tracked user. If a user is identified this way, the status of that user is changed.

After a user is identified as registered, the *KinectDriver* then computes the single-frame level wrong activity detection by invoking the *Activity Recognizer*. What activity is considered to be non-compliant is defined in the Configuration file. The result of detection is reported back to *KinectDriver*. Each user being tracked has a corresponding *User* class, which in turn performs an activity-level wrong activity detection. All detected wrong activities are recorded in cloud-based database and XML format file, which process will be described in Section V.

B. Handling of HTTP Requests

The Kinect server handles two types of HTTP requests. Both types of requests are issued by the smart phone on behalf of the WatchApp. The first type is registration request. In response to this request, the Kinect server attempts to register the user if the user has not been identified. If the user has already been identified and tracked by the system, the user is notified as such. The second type is the status request regarding whether or not the user has engaged in any wrong activity after the user has already been identified. If our current system is extended to incorporate inputs from wearable motion tracking devices, the Kinect server would receive the third type of requests for such reports.

1) Handling of Registration Request: As shown in Figure 4, the communication between server and wearable device is performed via HTTP requests and responses. It includes a mobile app that handles request and response between server and smart phone, and a watch app that handles interaction between smart phone and wearable device. The wearable device (i.e. Pebble Watch) provides an API program allowing bi-directional communication between phone apps and WatchApps. The user initiates registration request by pressing a button on watch. The mobile app receives the request and sends it to server. Once registration request is received by server, a *ServerTask* class object then is created to process the message. The *ServerTask* distributes HTTP request to another class *Kinect Coordinator* object. First *Kinect Coordinator* determines whether the user has been previously registered or not. If so registration request is forwarded to the *User* class to identify. The use's biometrics is used to examine. Upon completion of identification, a success response is sent back to mobile app immediately. Otherwise, a gesture recognition is performed using the most recent copy of the skeleton. The user gets notification of registration status from wearable device, such as displaying the message on watch face.

In the scenario of registration request is for a new user, gesture recognition is performed by *KinectDriver* class. A notification is sent back to requester regardless of the detection result through mobile app and watch app.

Our system is designed in such a way that same registration request can be used for registering a new user and periodically query the status of a previously registered user. The mobile app sends a query to server periodically. The response received from server is parsed. If the message is any of status changing from registered to unregistered, from unregistered to registered, or alert, the mobile app sends the notification to watch app that shows the message on watch face.

2) Handling of Activity Status Request: The activity status request is handled by server via *Kinect Coordinator* class. The process is relatively straightforward, as shown in Figure 5, which is similar to the handling of the registration request. First *Kinect Coordinator* determines whether the user has been previously registered or not. If so, the request is forwarded

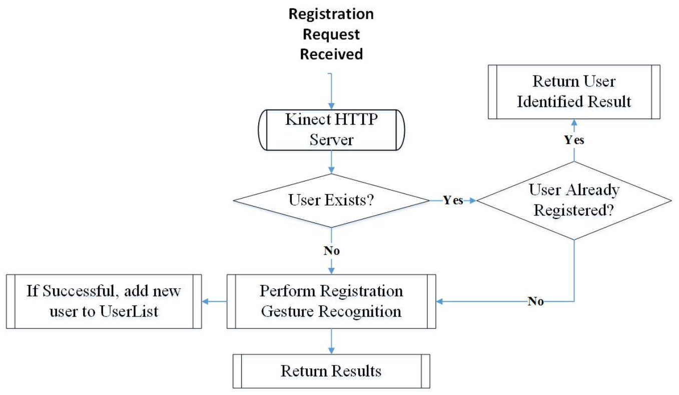

Registration Request Received
Kinect HTTP Server
User Exists?
Yes
User Already Registered?
Yes
Return User Identified Result
No
No
Perform Registration Gesture Recognition
If Successful, add new user to UserList
Return Results

Fig. 4. The flow control for registration request handling

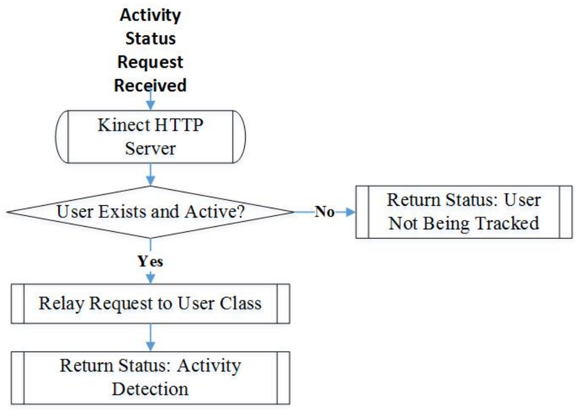

Activity Status Request Received
Kinect HTTP Server
User Exists and Active?
No
Return Status: User Not Being Tracked
Yes
Relay Request to User Class
Return Status: Activity Detection

Fig. 5. The flow control for activity status request handling

to the corresponding *User* class. Otherwise it notifies the requester that user is not being tracked in the system. The *User* class checks to see if any detection of non-compliant activities exists since last query. An alert is returned to user if any non-compliant activity detected. Or a null record is returned for indicating no wrong activity has been detected.

IV. Mobile App Operations

The primary operation of the mobile app is summarized in the finite state machine specification, which is controlled by three types of events, receiving registration request from wearable device, timeout, and receiving response from Kinect server.

- On receiving a registration request from the WatchApp, the requested is repackaged and relayed to the Kinect server for registering the user.
- On a timeout event. The timeout event facilitates a number of periodic tasks, which will be elaborated next. Another type of time out events is the connection error. This happens when the script fails to communicates with the Kinect server, for example, when a user moves outside the Bluetooth range, and when some error conditions occur such as a broken wifi connection between the Kinect server and the smart phone.

- On receiving a response from the Kinect server. This response can be for a registration request, or for a polling request for identification status and activity status.

The mobile app runs the following periodic tasks:

- User identification status check. After a user has registered with our system, the next time the user comes into the view of a Kinect sensor connected to our system, the Kinect server attempts to automatically identify the user based on his/her biometrics profile. To ensure the WatchApp has the same view of the identification status, the script periodically polls the Kinect server regarding the information. A notification is sent to the WatchApp when the status is changed to "user identified" from "user unidentified", and similarly when the status is changed to "user unidentified" from "user identified" (for example, when the user steps out of the view of the Kinect sensor).
- Alert notification on detection of wrong activities. The event loop periodically polls the Kinect server for each identified user regarding their activities. The Kinect server reports information regarding wrong activities the user has engaged in since the last poll. If one is detected, a notification is generated and sent to the WatchApp, which will trigger a vibration on the watch to alert the user.

The synchronization of the two modalities is accomplished via the recognition of a predefined gesture. In our current framework, all network inputs come from the dedicated mobile app. The WatchApp enables a user to register with the Kinect server, which is necessary for selective tracking. In essence, the selective tracking is made possible by combining two sensing modalities: (1) computer vision via Kinect, and (2) wearable sensing via the wearable device. The two modalities are combined via gesture recognition. On receiving a registration request, the Kinect server then examines the skeleton data received to identify the particular skeleton that performs the designated gesture. If one is found, that skeleton corresponds to the user who is requesting to be tracked.

V. Cloud-based Data Logging

Our system logs session information and any detected non-compliant activities. The logs are saved both remotely to the a cloud-based database in Amazon Web Services, and the locally in an XML log file. The advantage of logging data to cloud-based database is that all information from clients (i.e. different Kinect servers) can be kept in a centralized place. Utilizing centralized cloud-base database, the administrator of organization can review or analyze collected data from anywhere at anytime. Section VI details the design of our data visualization system.

The session information includes the time at which the user registered/identified, and the time at which the user is no longer identified. The XML file resides locally on the same location with Kinect server. However the cloud-based database resides in AWS, which requires the establishment of a TCP connection.

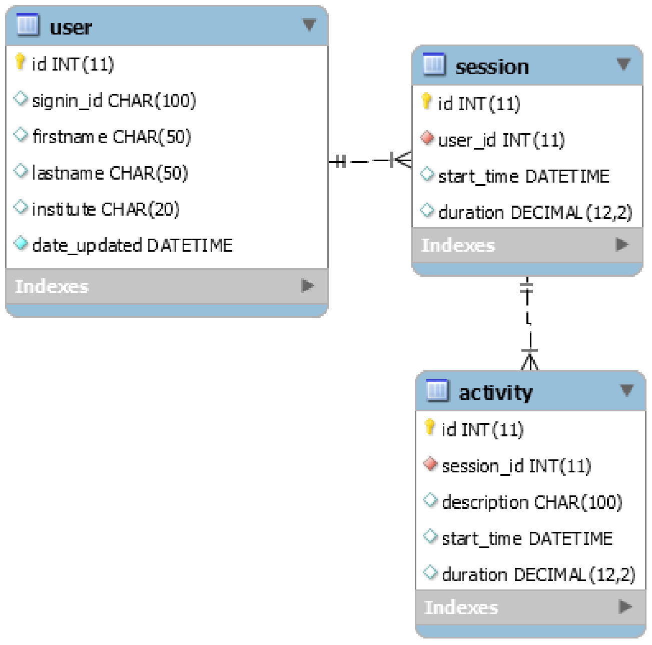


Fig. 6. The schema for logging in a Cloud-based database

The data log is structured in hierarchical relational data model in which information is organized into a tree-like structure. The top of tree is user who has children of sessions, each session has children of activities. Figure 6 shows data model structure. A data log composes two types of information: (1) session that includes the time when the user registered or identified, and time when the user unregistered or unidentified; (2) non-compliant activity that includes the time it is detected, type of activity and activity duration.

We choose mySQL that is an open-source database management system, to host log data in AWS cloud instance. The schema of log data contains entity of user, session and activities. The each data entity(i.e. table) includes a foreign key field that links to ancestor, such as session table has identity field for primary key field in user table (one user may have many sessions), user table has identity field for primary key field in session table (one session may have many activities). A database programming interface is created to handle all data retrieval and update between Kinect server and cloud database. It is designed to be an inheritance hierarchy in object model so each database table has a corresponding programming object that matches each data attribute with programming variable. The data access including insert, update, and retrieval. These tasks are performed by stored procedure with proper input and output parameters. The benefits of using stored procedure are: (1) reduce the network traffic between client and server; (2) stronger security that multiple users and client programs can operate on underlying database objects without direct permissions to those underlying objects; (3) easier maintenance and reuse of code.

In addition to cloud-based database, the logging information is saved to a locally resided XML file. The attributes of XML file are similar to data entities that are in hierarchy data model. The dual-modality in logging ensures high reliability of our framework. Even if the Amazon Web Services is temporarily unavailable, no data are lost because a local file can be a backup. We develop a function that can transfer logging information from XML file to cloud-based database.

VI. Data Visualization

We also implemented a companion application to display the logged records in a visual context for the users. The application can retrieve logged information from cloud-based the Amazon Web Services and also take local log files as input. The cloud-based data service is a centralized resource, which integrates logging data from different systems, such as Kinect server placed in different room. With visualized data graphics, the administrator can easily monitor the trend of activities, analyzing patterns, and correlations that might not be detected in a text-based data.

Our data process is constructed in three models: data acquisition, data aggregation, and data representation. The acquisition process is to collect data from different resource. This process can be performed when a new session starts, or a wrong activity is detected. The information is directly transmitted to cloud database in real time environment. A local XML logging file is also created to be served as a backup to cloud database. The local copy of logging can be load to cloud database at a later time. The data aggregation process is to gather and summarize the logging data, such as creating numeric activity counts from a specific user and a selected date time range. The aggregation group can be by date, or by hour for each user or for all users. Figures 7, 8 and 9 show an example log viewed by date and by hour, or show a detailed activities for a specific day. The summarized data is represented in a visual context that can help administrator or supervisor of health care organization to review, monitor and analyze the workers' activities. The visual graphics can easily demonstrate the trends or a critical time slot in which large number of wrong acuities found.

VII. Configurations

Users of our system can perform ADL tracking by defining rules for custom gestures for registration, and rules for their specific set of activities to detect. The Listing 1 shows an example of a rule definition for registration gesture, and a rule for a wrong activity (bending) to be detected in XML configuration format. The format of the rule definition described previously in [9], [8].

Both rules for registration gesture and non-compliant activity are defined in terms of the bone orientation. The corresponding alert level and type are defined as well. The alert notifications are used to instruct the watch app to inform the user when the registration is successful or when a non-compliant activity is detected. The registration gesture is defined by the orientation between the lower arm (i.e. from elbow to wrist) and the transverse plane, which should be 90 degrees with a tolerance of 20 degrees. One of the sample non-compliant activities is when a bending pose occurs. It is detected when the user's trunk deviates from the frontal plane by more than 20 degrees.

Listing 1. Definition for an example registration gesture and an example wrong activity.

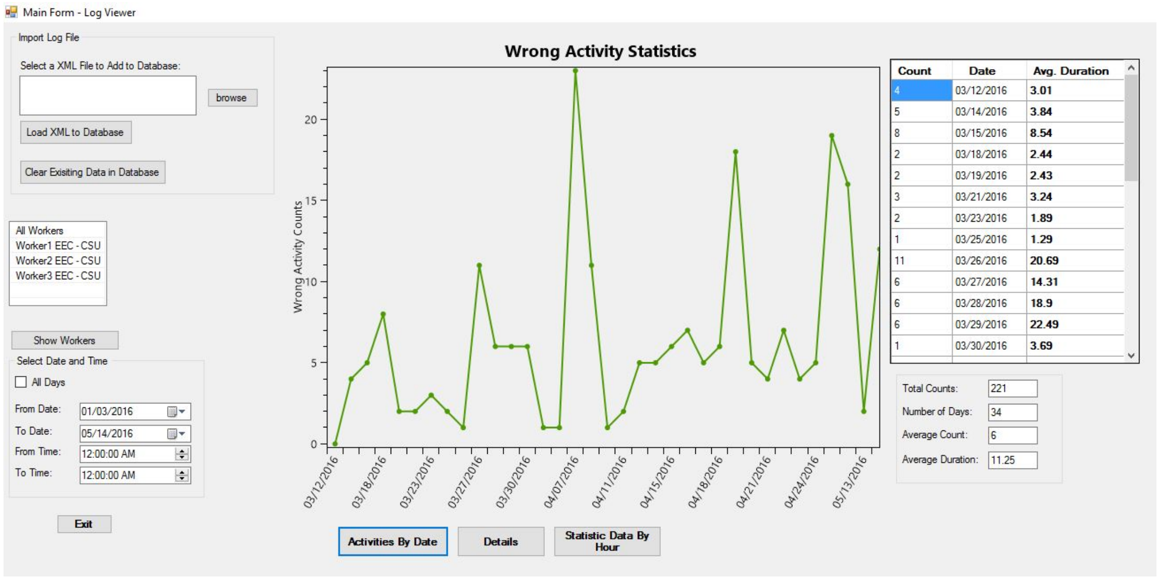
```xml
<RegisterRules>
    <Configuration>
        <Type>BoneOrientation</Type>
        <AlertLevel>"1"</AlertLevel>
        <AlertBody>"Registration"</AlertBody>
        <AlertVibration>"0"</AlertVibration>
        <DownstreamJoint>"WristLeft"</DownstreamJoint>
        <UpstreamJoint>"ElbowLeft"</UpstreamJoint>
        <TransverseAngle>"90"</TransverseAngle>
        <Tolerance>"20"</Tolerance>
        <Condition>"At"</Condition>
    </Configuration>
</RegisterRules>
<ActivityRules>
    <Configuration>
        <Type>BoneOrientation</Type>
        <AlertLevel>"1"</AlertLevel>
        <AlertBody>"Bend"</AlertBody>
        <AlertVibration>"1"</AlertVibration>
        <DownstreamJoint>"HipCenter"</DownstreamJoint>
        <UpstreamJoint>"ShoulderCenter"</UpstreamJoint>
        <FrontalAngle>"20"</FrontalAngle>
        <Condition>"GreaterThan"</Condition>
    </Configuration>
</ActivityRules>
```

The use of XML-based configuration allows users to define any registration gesture rule and wrong activity rule as long as the format that is shown above is followed. In addition, the rules can be easily updated without changing the source code.

VIII. Related Work

The tracking of the activities of daily living may have significant implications in healthcare because it would enable healthcare professionals to receive updates *remotely* regarding the functional status of post-injury and post-surgery patients, and people with disabilities and the elderly. Hence, it has been a hot research area in the last several years [1], [2], [3]. Most ADL tracking systems use wearable devices that are equipped with inertial sensors, such as fitness trackers including fitbit, Jawbone, and Microsoft band, and smart watches including Pebble smart watch, Android Wear smart watch, and Apple IWatch. While these wearable devices are easy to use, they are only geared to track highly rhythmic activities such as walking, running, and sleeping. They are inadequate to track more sophisticated activities, such as correct ways of lifting and pulling, or fitness and rehabilitation exercises. To track such activities, non-intrusive computer-vision based motion tracking sensors, such as Microsoft Kinect, are a better fit [9], [19], [20], [21], [15]. That said, Kinect-based systems do have a limitation, *i.e.,* they can only be used to track human activities in doors because the depth sensing depends on the use of infra-red light, which does not work outdoors. The integration of the two approaches makes it possible to continuously track a person both indoors and outdoors, which is the focus of this paper.

There is a vast body of research on human activity tracking. For a comprehensive review of the research on human activity tracking using Microsoft Kinect, readers are referred to [14]. Approaches to human activity recognition can be roughly divided into two categories: (1) template based and (2) rule based. In the template based approach, the sequence of an activity is first recorded, and the recorded sequence is then used as an exemplar to be compared with the observed activity either directly, or to train a model for the activity, which is



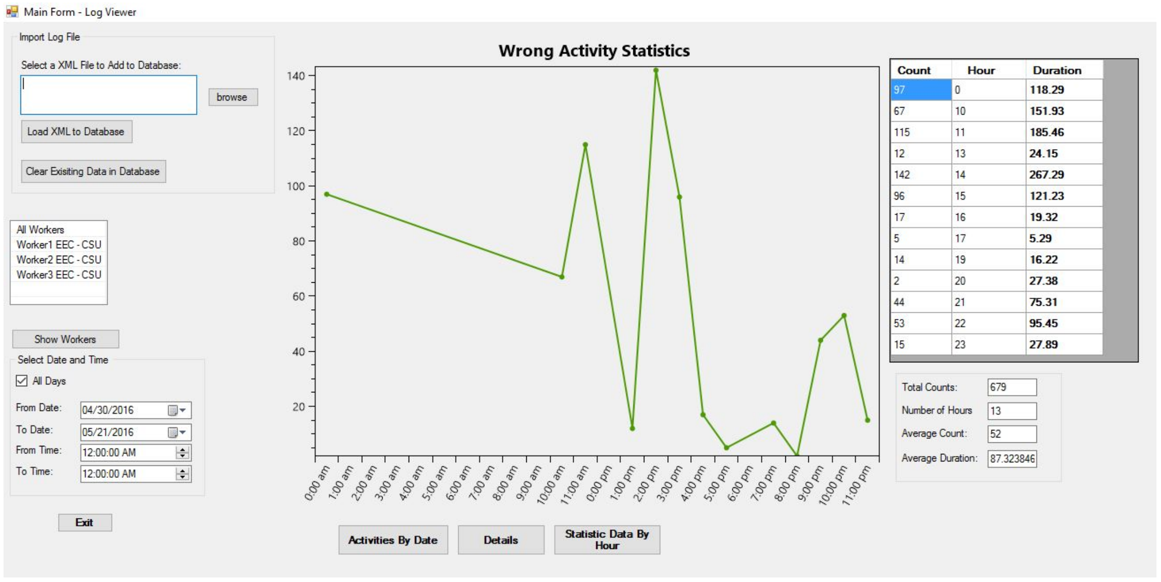
☐ All Days

From Date:	01/03/2016
To Date:	05/14/2016
From Time:	12:00:00 AM
To Time:	12:00:00 AM

Count	Date	Avg. Duration
4	03/12/2016	3.01
5	03/14/2016	3.84
8	03/15/2016	8.54
2	03/18/2016	2.44
2	03/19/2016	2.43
3	03/21/2016	3.24
2	03/23/2016	1.89
1	03/25/2016	1.29
11	03/26/2016	20.69
6	03/27/2016	14.31
6	03/28/2016	18.9
6	03/29/2016	22.49
1	03/30/2016	3.69

Total Counts:	221
Number of Days:	34
Average Count:	6
Average Duration:	11.25

Fig. 7. The visualization of an example log by date



☑ All Days

From Date:	04/30/2016
To Date:	05/21/2016
From Time:	12:00:00 AM
To Time:	12:00:00 AM

Count	Hour	Duration
97	0	118.29
67	10	151.93
115	11	185.46
12	13	24.15
142	14	267.29
96	15	121.23
17	16	19.32
5	17	5.29
14	19	16.22
2	20	27.38
44	21	75.31
53	22	95.45
15	23	27.89

Total Counts:	679
Number of Hours	13
Average Count:	52
Average Duration:	87.323846

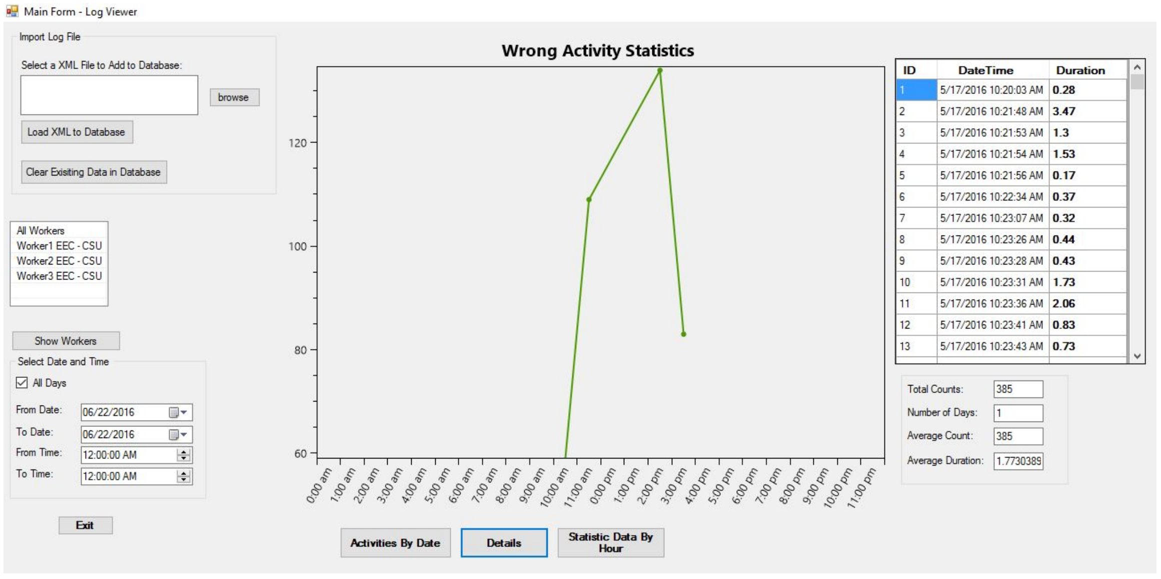
Fig. 8. The visualization of an example log by hour

then used to classify the observed activity. The method used to train the model varies, such as hidden Markov models [22] and neural networks [23]. The main benefit of the template based approach is that either no model is needed, or the model parameters can be fitted automatically using exemplar data if a model is used. As a tradeoff, the feedback provided by these approaches often contains limited information. The rule based approach does not require the recording of exemplars and the training of sophisticated statistical models. Instead, an activity is defined by a set of kinematic rules, created by experts, that capture the key features of the activity. The rule-based methods are less computationally intensive and could provide specific feedbacks regarding exactly which rule is violated. Hence, we choose to follow the rule-based approach in this study.

A well-known rule-based gesture recognition system is reported in [24], where a Gesture Description Language (GDL) is introduced for general-purpose human activity recognition, in which an activity is determined by a set of key frames. All rules are expressed in terms of one or more key frames except the final rule, which defines the gesture in terms of a sequence of basic rules. However, GDL lacks the support for rules that depend on the entire trajectory of a gesture. It also lacks a guideline as to how to identify the key frames for each gesture reliably.

Previously, we introduced three types of rules for each activity for the purpose of assessing the quality of rehabilitation exercises [8], [9], [20], [25]:

- Rules for dynamic movement. Each rule is expressed in terms of the sequence of reference configurations of a particular joint or body segment (such as an arm



From Date: 06/22/2016
To Date: 06/22/2016
From Time: 12:00:00 AM
To Time: 12:00:00 AM

ID	DateTime	Duration
1	5/17/2016 10:20:03 AM	0.28
2	5/17/2016 10:21:48 AM	3.47
3	5/17/2016 10:21:53 AM	1.3
4	5/17/2016 10:21:54 AM	1.53
5	5/17/2016 10:21:56 AM	0.17
6	5/17/2016 10:22:34 AM	0.37
7	5/17/2016 10:23:07 AM	0.32
8	5/17/2016 10:23:26 AM	0.44
9	5/17/2016 10:23:28 AM	0.43
10	5/17/2016 10:23:31 AM	1.73
11	5/17/2016 10:23:36 AM	2.06
12	5/17/2016 10:23:41 AM	0.83
13	5/17/2016 10:23:43 AM	0.73

Total Counts: 385
Number of Days: 1
Average Count: 385
Average Duration: 1.7730389

Fig. 9. The visualization of an example detail log in a specific day

or leg) that delineate monotonic motion segments of each iteration.

- Rules for static poses. Some activities only involve stationary poses. It is also possible for some body parts to remain stationary at their desirable positions while other parts are moving in some other exercises. In these cases, static rules are needed.

- Rules for movement invariance, each of which defines the requirement for a moving body segment that must be satisfied throughout the entire activity.

For extensibility and adaptability, the rules are encoded using XML. In this study, we apply these rules to define specific activities and gestures that we are interested in.

IX. Conclusion and Future Work

In this paper, we presented the detailed design and implementation of a system that integrates computer vision-based server, image sensor, mobile smart phone, wearable device, and cloud web server for motion tracking. By integrating the two forms of motion sensing modalities(i.e. mobile app and watch app), it is possible to use our system to track ADL both in indoor and outdoor environments. Our system can be open to communication with other systems and mobile/wearable devices. The interfaces in our systems are well designed to accommodate different wearable computing platforms, and allow users to incorporate different registration gestures, and to specify different activities to be detected by defining rules in the configuration file. In the future work, we plan to extend our system on two aspects: (1) facilitate the communication between multiple Kinect servers so that we can federate them together to cover a large area and/or multiple rooms; and (2) to integrate the health data collected by various health tracking apps via the Google Health and Apple Health Kit platforms.

Acknowledgments

This study is supported in part by a grant from the Ohio Bureau of Workers' Compensation (SPC1004933944), by an award for undergraduate summer research, and a graduate faculty support award, both from Cleveland State University.

References

[1] D. Foti and L. Kanazawa, "Activities of daily living," *Pedretti's Occupational Therapy: Practice Skills for Physical Dysfunction*, vol. 6, pp. 146–194, 2008.

[2] M. Giné-Garriga, M. Roqué-Fíguls, L. Coll-Planas, M. Sitja-Rabert, and A. Salvà, "Physical exercise interventions for improving performance-based measures of physical function in community-dwelling, frail older adults: a systematic review and meta-analysis," *Archives of physical medicine and rehabilitation*, vol. 95, no. 4, pp. 753–769, 2014.

[3] L. S. Noelker and R. Browdie, "Sidney katz, md: A new paradigm for chronic illness and long-term care," *The Gerontologist*, p. gnt086, 2013.

[4] K. M. Diaz, D. J. Krupka, M. J. Chang, J. Peacock, Y. Ma, J. Goldsmith, J. E. Schwartz, and K. W. Davidson, "Fitbit: An accurate and reliable device for wireless physical activity tracking," *Intl J Cardiol*, vol. 185, pp. 138–40, 2015.

[5] H. E. Montgomery-Downs, S. P. Insana, and J. A. Bond, "Movement toward a novel activity monitoring device," *Sleep and Breathing*, vol. 16, no. 3, pp. 913–917, 2012.

[6] R. Rawassizadeh, B. A. Price, and M. Petre, "Wearables: Has the age of smartwatches finally arrived?" *Communications of the ACM*, vol. 58, no. 1, pp. 45–47, 2015.

[7] C.-J. Su, C.-Y. Chiang, and J.-Y. Huang, "Kinect-enabled home-based rehabilitation system using dynamic time warping and fuzzy logic," *Applied Soft Computing*, vol. 22, pp. 652–666, 2014.

[8] W. Zhao, R. Lun, D. D. Espy, and M. Reinthal, "Rule based realtime motion assessment for rehabilitation exercises," in *Proceedings of the IEEE Symposium on Computational Intelligence in Healthcare and e-health*. IEEE, 2014, pp. 133–140.

[9] W. Zhao, R. Lun, D. D. Espy, and M. Ann Reinthal, "Realtime motion assessment for rehabilitation exercises: Integration of kinematic modeling with fuzzy inference," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 4, no. 4, pp. 267–285, 2014.

[10] J. N. Katz, "Lumbar disc disorders and low-back pain: socioeconomic factors and consequences," *The Journal of Bone & Joint Surgery*, vol. 88, no. suppl 2, pp. 21–24, 2006.

[11] A. K. Burton, "Back injury and work loss: biomechanical and psychosocial influences," *Spine*, vol. 22, no. 21, pp. 2575–2580, 1997.

[12] W. S. Marras, S. A. Lavender, S. E. Leurgans, S. L. Rajulu, W. G. Allread, F. A. Fathallah, and S. A. Ferguson, "The role of dynamic three-dimensional trunk motion in occupationally-related low back disorders: The effects of workplace factors, trunk position, and trunk motion characteristics on risk of injury." *Spine*, vol. 18, no. 5, pp. 617–628, 1993.

[13] G. Mastorakis and D. Makris, "Fall detection system using kinect's infrared sensors," *Journal of Real-Time Image Processing*, pp. 1–12, 2012.

[14] R. Lun and W. Zhao, "A survey of applications and human motion recognition with microsoft kinect," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 29, no. 5, p. 1555008, 2015.

[15] W. Zhao, R. Lun, C. Gordon, A.-B. Fofana, D. Espy, M. A. Reinthal, B. Ekelman, G. Goodman, J. Niederriter, C. Luo, and X. Luo, "A privacy-aware kinect-based system for healthcare professionals," in *Proceedings of the IEEE International Conference on Electro-Information Technology*. Grand Forks, ND, USA: IEEE, May 2016, pp. 205–210.

[16] "Amazon web services." [Online]. Available: https://aws.amazon.com/

[17] R. M. Araujo, G. Graña, and V. Andersson, "Towards skeleton biometric identification using the microsoft kinect sensor," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 2013, pp. 21–26.

[18] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," 2010. [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf

[19] W. Zhao, D. D. Espy, M. Reinthal, B. Ekelman, G. Goodman, and J. Niederriter, "Privacy-aware human motion tracking with realtime haptic feedback," in *Proceedings of the IEEE International Conference on Mobile Services*. IEEE, 2015, pp. 446–453.

[20] W. Zhao, D. D. Espy, M. Reinthal, and H. Feng, "A feasibility study of using a single kinect sensor for rehabilitation exercises monitoring: A rule based approach," in *Proceedings of the IEEE Symposium on Computational Intelligence in Healthcare and e-health*. IEEE, 2014, pp. 1–8.

[21] W. Zhao, H. Feng, R. Lun, D. D. Espy, and M. Reinthal, "A kinect-based rehabilitation exercise monitoring and guidance systems," in *Proceedings of the 5th IEEE International Conference on Software Engineering and Service Science*. IEEE, 2014, pp. 762–765.

[22] J.-S. Lin and D. Kulic, "Online segmentation of human motion for automated rehabilitation exercise analysis," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 22, no. 1, pp. 168–180, 2014.

[23] S. Nomm and K. Buhhalko, "Monitoring of the human motor functions rehabilitation by neural networks based system with kinect sensor," in *Analysis, Design, and Evaluation of Human-Machine Systems*, vol. 12, no. 1, 2013, pp. 249–253.

[24] T. Hachaj and M. R. Ogiela, "Rule-based approach to recognizing human body poses and gestures in real time," *Multimedia Systems*, vol. 20, no. 1, pp. 81–99, 2014.

[25] W. Zhao, "On automatic assessment of rehabilitation exercises with realtime feedback," in *Proceedings of the IEEE International Conference on Electro-Information Technology*. Grand Forks, ND, USA: IEEE, May 2016, pp. 376–381.