3D Human Activity Recognition by Indexing and Sequencing (RISq)

BY

SADGUN SRINIVAS DEVANAHALLI SHASHIKUMAR

B.E., Visvesvaraya Technological University, India, 2012

THESIS

Submitted as partial fulfillment of the requirements

for the degree of Master of Science in Electrical and Computer Engineering

in the Graduate College of the

University of Illinois at Chicago, 2015

Chicago, Illinois

Defense Committee:

Jezekiel Ben-Arie, Chair and Advisor

Rashid Ansari

Milos Zefran

To my mother, Sharada, and my late father, Shashikumar.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Prof. Jezekiel Ben-Arie for all his invaluable support and guidance. This work would have never been possible without his guidance. I would also like to thank my thesis defense committee members Prof. Milos Zefran and Prof. Rashid Ansari for their support and suggestions.

I would also like to thank the members of the Machine Vision lab namely Kai Ma for all his support and suggestions, Yicheng Wang for helping me out with programming. A special thanks to Uzair Ahmed, Aruna Sundar, and Charenraj Prema Jegadeesan for all their help during the various stages of my thesis.

SS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF ABBREVIATIONS

RISq	Recognition by Indexing and Sequencing
HAR	Human Activity Recognition
LDA	Linear Discriminant Analysis
ANN	Artificial Neural Networks
HMM	Hidden Markov Model
MEMM	Maximum Entropy Markov Model
DTW	Dynamic Time Warping
SVM	Support Vector Machine
ROC	Receiver Operating Characteristic
RGB	Red – Green – Blue
CMOS	Complementary Metal – Oxide – Semiconductor
VGA	Video Graphics Array
IR	Infra – Red
SDK	Software Development Kit
CSV	Comma – Separated Value

SUMMARY

Human activity recognition has been a crucial area of research in computer vision over the past several years. Myriad applications of human activity recognition have been the driving force behind this research. The goal of human activity recognition is to automatically examine and classify activities in a 2D or 3D video sequence. A good deal of state-of-the art recognition algorithms has been researched and developed for this purpose, each of them handling the problem of human activity recognition in a unique way and achieving significant recognition rates. The advent of the kinect camera from Microsoft has been a breakthrough for the research on human activity recognition. The advantages of the 3D kinect camera over a 2D video cameras has enabled researchers to develop much more reliable recognition algorithms which are invariant to human position and orientation. These algorithms are capable of achieving correct recognition rates much higher than what was achieved in the past.

In the thesis presented here, we propose a novel approach for the problem of human activity recognition. We exploit the advantages of the kinect camera and the simplicity of our recognition algorithm: Recognition by Indexing and Sequencing (RISq). We capture the activity sequences using the kinect, and with the help of the software packages associated with the kinect we extract the skeletal information of the person. This information after processing acts as feature inputs for the RISq algorithm. The advantage of the kinect over conventional 2D cameras enables us to acquire this skeletal information with relative ease. With these features we train the RISq algorithm to recognize and classify an unknown activity sequence.

SUMMARY (Continued)

RISq is a non-parametric approach used to classify temporal vector sequences. In the past, RISq has been successfully implemented for human activity recognition, speech recognition, and object detection. The RISq comprises of a two-step classification algorithm: first the vector samples of exemplars which are closest to vectors of the input are identified and weighted with a parallel algorithm (indexing). Next, a maximum weighted bipartite graph matching is found between the input vector sequence and an exemplar vector sequence, respecting a temporal constraint (sequencing).

The basic algorithm of RISq is extended to our problem of human activity recognition with minor changes. Experiments were conducted on two datasets: one acquired at our Machine Vision lab and the other from the Cornell University's Robot Learning lab. The dataset from our lab consists of four activities: Sit, Stand, Jump, and Walk. The dataset of Cornell University's Robot Learning Lab consists of six activities: Chopping, Stirring, Drinking, Talking over the phone, Working on laptop, Writing on whiteboard. Results from experiments on these datasets show that RISq performs better and achieves higher recognition rates. The performance of RISq was compared with state-of-the art HMM based algorithms and results show that RISq outperforms these algorithms with higher recognition scores.

1. INTRODUCTION

Human activity recognition is a significant area of research in today's computer vision. The ultimate aim of human activity recognition is to automatically analyze and categorize activities from a 2D or a 3D video sequence. Such video sequences can describe activities which vary from a simple case of only one activity to multiple activities. Our objective in this work is to classify these activities correctly and to be able to recognize the different activities in a continuous sequence.

Human activity recognition leads to many potential areas of applications. These applications areas can be mainly categorized into four areas namely: control, monitoring, surveillance and analysis.

The application areas which are categorized under 'control' can be related to the applications where in the captured motion and recognized activities are used to provide some sort of controlling functionalities. These functionalities could be those which are used to interface games, virtual environments or control of mobile robots.

Surveillance covers areas where there are one or more subjects (persons) who are being tracked and their activities analyzed. An example for surveillance is analyzing the actions of a person at an airport and detecting any suspicious activity.

Monitoring is the area where a person and his actions are being tracked and monitored. This application ensures the safety of elderly persons or patients who live alone in nursing homes or hospitals.

The applications which require a detailed study of the captured activity fall under the analysis area. This could be in the area of sports where a detailed study of the captured motion is needed. This information is then later used to better understand the athletes and hence improve their performance based on the analysis.

Based on their complexity, human activities can be categorized into different levels. These levels can be roughly categorized as: gestures, actions, activities, interactions and group activities. The first three levels namely gestures, actions, and activities are basically performed by one person. Gestures and actions can be a simple wave of the hand or just stretching of the arm. Activities are comprised of a sequence of multiple actions, for example walking, sitting, jumping are all comprised of multiple actions. The remaining two type's interactions and group activities are based on the actions performed between two or more persons. The activities performed for this thesis belong to the first three levels: gestures, actions, and activities.

In this thesis we tackle the problem of human activity recognition, mainly focusing on videos that describe only one activity. However, the method RISq we employ can recognize also a sequence of activities without needing to segment the input. We present an approach that uses the skeletal information from the kinect camera to recognize activities, in the process we show that this approach is robust in many conditions. The main contributions of this thesis are as follows:

1. Propose a skeletal based recognition framework for recognizing activities that is robust to inter class variations.
2. Exploit the simplicity and advantages of RISq, RISq is a sequence recognition method which will be described in chapter 5, also to equip the framework to be robust to changes in activity rate and to achieve high recognition rates even when there are fewer models for training.
3. Explore the advantages of using a 3D camera, such as the Microsoft kinect, as opposed to that of conventional 2D video camera for activity recognition.

1.1 Outline of the thesis

The goal of chapter 1 is to provide a brief introduction of human activity recognition, some of its applications, and the main contributions of our work. The rest of the chapters in this thesis present the state-of-the art methods for activity recognition, some technical concepts, RISq, and the results obtained from our work.

Chapter 2 discusses the basics of human activity recognition and the problem of activity recognition and also presents a detailed review of some of the other state of the art methods that have been carried out on this topic.

Chapter 3 presents some of the theoretical foundations of our recognition approach. These are some of the concepts, recognition algorithms, hardware, software and tools that are generally used in all human activity recognition frameworks.

Chapter 4 introduces the first step in our framework, i.e. the feature extraction stage. It first explains the method and tools used for extracting the skeletal information. Next it details the techniques used for processing this information. The normalization scheme that is used to convert

our data from the kinect coordinate system to the human centered coordinate system is also presented.

Chapter 5 presents the second stage of our framework, the indexing, sequencing and voting method, which is called RISq (Recognition by indexing and sequencing). The method used for storing the feature vectors is also explained. This is followed by the detailed explanation of the indexing, sequencing and voting stage. This chapter covers the thresholding stage and also presents some of the results obtained from our dataset.

Chapter 6 describes the experimental results of our approach. These results include the results obtained when our method was tested on a dataset acquired at our lab as well a dataset presented by Cornell University's Robot Learning Lab: the CAD-60 dataset. The conclusion and our suggestions for the future work that is to be implemented if one wants to achieve even higher recognition rates is also presented here.

2. BACKGROUND

Human activity recognition has been a key area of research in computer vision due to its myriad applications. The final goal of activity recognition is to successfully identify an activity in a given video, this can be done either in real-time or offline. These activities as mentioned in section 1 can be simple gestures or a more complex set of actions between two or more individuals. Activity recognition has seen many advancements in recent times, with the emergence of depth cameras like kinect, Xtion, etc. methods involving depth and human skeletal information are being researched. However even the most advanced methods are still far behind humans ability to recognize activities. The next section presents some of the challenges faced in activity recognition.

2.1 Challenges faced in activity recognition:

1. Intra-Class Variation:

   Every individual tends to perform an activity in his/her own manner, hence each activity is unique. Due to this, a single activity class can have multiple ways of the same activity being performed.

2. Inter-Class Variation:

   In some situations the difference between two activities can be very minute. An example for this is the ‘walking’ and ‘running’ activity, the difference between these activities is the rate at which these are performed (one being faster than the other) and different hand postures.

3. Activity rate:

    Individuals predominantly perform an activity at his/her pace, due to this fact the time interval required for an individual to perform the same activity is different from one person to other persons.

Other reasons for imperfect activity recognition include occlusions, lack of depth/skeletal information, inadequate information to construct silhouette, etc. Thus the main goal of researchers is to overcome these problems and achieve robust recognition. Having said this however we have to bear one thing in mind, trying to handle only one problem can adversely affect the others. For example, designing a recognition algorithm that can handle large intra-class variations could fail when there are occlusions or the change in activity rate is too high. In the next section we discuss the work of other researchers on this topic, we describe the methods they have used to overcome the above constraints, and also explain the motivation for our work.

2.2 Previous Work:

Many researchers have devised state-of–the art algorithms for activity recognition. Although the final goal of all these algorithms is the same, the ways they approach that goal are quite different. These differences can be in the feature selection stage or in the recognition stage. We review some of these approaches while noting the advantages of one over the other.

2.2.1 Wearable Sensors Approach:

One approach for human activity recognition employs wearable sensors. Sensors are placed on the human body, the data from these sensors is fed as features to classification algorithms like linear-discriminant analysis (LDA), artificial neural networks (ANN), hidden markov models (HMM) etc. These sensors are generally some sort of accelerometers, gyroscope, and magnetometers. The features are generally vectors with a single or combination of acceleration values, average acceleration energy, average velocity, zero crossing rate, average rotation angles, etc. For example the approach by Khan et al. [15] use a single tri-axial accelerometer while others like Zhang et al. [34] use a combination of sensors for acquiring the features. The approach by Khan et al. [15] uses a single tri-axial accelerometer attached to the person's chest, the acceleration signals obtained from the accelerometer are taken as feature vectors after noise reduction. The recognition stage used a combination of LDA and ANN. The technique proposed in Zhang et al. [34] uses a wearable sensing device called MotionNode, which is a combination of a three-axis accelerometer, three-axis gyroscope, and a three-axis magnetometer. According to the technique the device is bundled in a mobile phone pouch and is attached to the person's front hip. The signals from this device are then used to construct an over-complete dictionary. Recognition is achieved using this dictionary and sparse representation theory. The common attribute of the above methods

is that they use some sort of sensor like accelerometer, gyroscope, etc. or a combination of sensors. Using sensors for recognition enables to achieve high recognition rates, but this is well suited for controlled environments like hospitals, fitness centers, laboratories or research centers where attaching the sensor onto a person is feasible. This technique fails when the environment changes, for example an airport or a parking lot, it is impossible to attach a sensor on a person at such a place to monitor the person's activities. In addition, these techniques are unable to recognize many activity classes as we do.

2.2.2 Silhouette-Based Techniques

Silhouette based methods are some of the popular technique for representing human activity. In this approach the silhouette of a human body is extracted from the video sequence, features used in this case could be joint angles, 3D models constructed from the silhouettes, etc. Feng et al. [9] utilized principal component (PC) features from binary silhouettes, optical flow-based models and HMM to recognize different view invariant activities. Uddin et al. [28] approach is similar to that of [9] but instead of using principal component analysis they use independent component analysis and achieve high recognition. Sung et al. [27] proposed to use the MEMM for activity recognition, in this method the person's activity is composed of a set of sub-activities and using dynamic programming a two-layered graph structure is inferred. Uddin et al. [28] proposed to use the 3D joint angles of the human body for activity recognition. The method derives discrete symbols from the joint angle features, obtained from 3D models of a human body, to train the HMM. The database consisted of 6 activities, and 15 images for each activity class was used for training. Using the depth sequences obtained from a depth camera, Li et al. [19] proposed a bag-of-3D-points feature representation for activity recognition. The silhouettes of the depth sequence are sampled to obtain the 3D points, while action graph was used for classification.

The silhouette based methods when compared to the wearable sensors approach has the advantage of eliminating the need of placing a sensor on the person. This approach has many advantages when compared to wearable sensors approach, but when classification algorithms like HMM's are used the training models must be relatively large. The silhouettes approach is susceptible to occlusion problems, and obtaining a good silhouette when the person is not facing the camera or depth sensor is quite a challenge.

2.2.3 Skeletal Based Approach

Skeletal based recognition was introduced by Ben-Arie et all [2][3], but in the work 2D videos were used which limits the recognition to a narrow range of viewing directions. Recent and advanced approaches exploit the skeletal tracking capabilities of the Microsoft kinect sensor. This approach utilizes the software from Microsoft or other ones such as OpenNI and NITE, discussed in chapter 3, to detect a human and map a skeleton of the person from the depth image. Many researchers have followed this approach and achieved high recognition rates, our work is also based on this method.

Using the kinect sensor and Microsoft software for acquiring the skeleton of the person, Bengalur et al. [4] proposed an approach which utilizes SVM for activity recognition. The author has used the 15 joints obtained from the kinect, of which 10 joints have orientation information. These joints are represented as half-space quaternions. The features that are used in this are the joint, velocity, and posture information. The training has a total of 650 videos which covers 13 activities, with each activity having 50 videos. High recognition rates were obtained from this approach. Sung et al. [27] have used the kinect sensor and maximum entropy markov model (MEMM) for their recognition framework. The skeletal information along with the RGB and depth

data are given as input to the learning algorithm. They use a two-layered maximum entropy markov model for training and recognition.

The method of using the person's skeleton is a great technique since it does not depend on the use of any wearable sensor. This method can also be used when the person is not facing the camera and is at a particular angle from the camera. Our work is similar to the approaches followed in this section, but the main difference is the recognition stage. The above methods use SVM and HMM which are great classification algorithms, but both of them require large number of training models. Our method proposed utilizes RISq, a non-parametric technique which is used for recognition of sequences. RISq has been successfully implemented in the past for activity recognition by Ben-Arie et al [2][3]. We extend this method from 2D to 3D skeletal information obtained from the kinect. The next section presents a brief outline of our method and explains the feature selection stage.

3. TECHNICAL FOUNDATIONS

In this chapter we describe the theoretical foundations and some of the tools that we use for capturing and recognizing the human motion.

3.1 3-D Rotations and translations:

In this section, various transformations for 3-D rotations and translations are addressed. 3-D transformations are used throughout our work to transform the locations and orientations of the different body joints obtained in the coordinate system of the kinect to the body centered coordinate system. Computation of the body centered 3-D orientations and locations of the different body joints is necessary in order to achieve view invariant activity recognition system. Computation of the body centered 3-D orientations and locations of the different body joints is done by multiplying the locations of the different body joints in the kinect coordinate system by a rotation matrix (R) and a translation function (t) , mathematically this is written as:

$$\boldsymbol{x}' = R\boldsymbol{x} + \boldsymbol{t} \text{ Or } \boldsymbol{x}' = [R \quad \boldsymbol{t}]\,\overline{\boldsymbol{x}} \tag{1}$$

where,

$\boldsymbol{x}$ and $\overline{\boldsymbol{x}}$ are the point coordinates of the locations of the joints in the kinect coordinate system represented as inhomogeneous coordinates, $\boldsymbol{x} = (x, y, z) \in \boldsymbol{R^3}$, or as an augmented vector $\overline{\boldsymbol{x}} = (x, y, z, 1)$.

$\boldsymbol{x}'$ is the set of the coordinates of the body joints in three dimensions with respect to the body centered coordinate system.

The rotation matrix (R) is a matrix whose multiplication with a vector rotates the vector while preserving its length. The rotation matrix is a 3 x 3 orthonormal matrix with $RR^T = I$ and $|R| = 1$. We generally reference the rotation matrix as follows:

$$R = [r_1 \quad r_2 \quad r_3]$$

$$\Longrightarrow \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2}$$

Translation is a function that moves every point a constant distance in a specified direction. Translation can also be interpreted as the addition of a constant vector to every point, or as shifting the origin of the coordinate system.

Matrix Representation:

To translate a three dimensional object by a vector $\boldsymbol{V}$, each homogeneous vector $\boldsymbol{p}$ of the object can be multiplied by a translation matrix:

$$V = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The multiplication $V * \boldsymbol{p}$, will give us:

$$V * \boldsymbol{p} = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x + v_x \\ p_y + v_y \\ p_z + v_z \\ 1 \end{bmatrix} \tag{3}$$

There are various methods to calculate rotations in 3-D, some of the most common ones are Euler angles, Axis-Angle, and Quaternions. A detailed discussion about how these transformations address the non-euclidean nature of 3-D rotations is given by Grassia et all [11]. Each of these transformations has its own advantaged and disadvantages. Quaternions and axis-angles are popular ways to parameterize 3-D rotations in today's computer vision. The following sections describe the above mentioned transformations in more detail.

3.1.1 Euler Angles

Euler angles are most common and intuitive transformation of 3-D rotation. According to Euler, any rotation in 3-D space can be achieved as a sequence of 3 individual rotations around 3 orthogonal coordinate axes. Since rotation around a single axis can be easily measured by a single angle, we can parameterize rotation in 3-D by three angles. The Euler angle convention is to

1. Rotate about one body coordinate axis (which rotates the other two).
2. Then rotate about a second body coordinate axis (rotated from its original direction) not identical to the first.
3. Lastly, rotate about another body coordinate axis not identical to the second.

Let the Euler angles with respect to *X*-axis, *Y* -axis and *Z*-axis are denoted by α, β and γ respectively, corresponding homogeneous rotation matrices are given as:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Here, the positive angles are measured *counter-clockwise* while looking towards the origin.

Euler angles lead to a very compact representation. They are also very intuitive to user. One important advantage of Euler angles parameterization is that for any specified value of angles, the user is sure to get a valid rotation. In case of other parameterizations such as quaternions or matrix, constraints must be observed for a valid rotation. However *singularities* in the Euler angles parameterization can create problems. Near singularities, one or more of the Euler angles can vary over a wide range without producing any corresponding change in the rotation. A very significant problem with the Euler angles is that they cannot be interpolated in a meaningful way. Halfway between two Euler angle triplets in terms of parameter values is not necessarily halfway in terms of rotation. This is because same rotation can be achieved using different triplets of angles but different triplets of angles would yield different in-between results when interpolated. A key point to remember about Euler angles is that they do not provide three independent controls for controlling a rotation, changing one of the angles alters the meaning of subsequent angles.

3.1.2 Axis-Angle

The axis-angle representation evolves from Euler's rotation theorem, which implies that any rotation or sequence of rotations of a rigid body in three-dimensional space is equivalent to a pure rotation about a single fixed axis. Axis-angle representation transforms the rotation in a 3-D Euclidean space by a set of two values. The first is a unit vector ê which indicates the direction of an axis of rotation, the other is angle θ which describes the angle of rotation about that axis. The right –hand rule is followed for the rotation about the axis. The rotation axis is sometimes also referred to as the *Euler axis.*

`

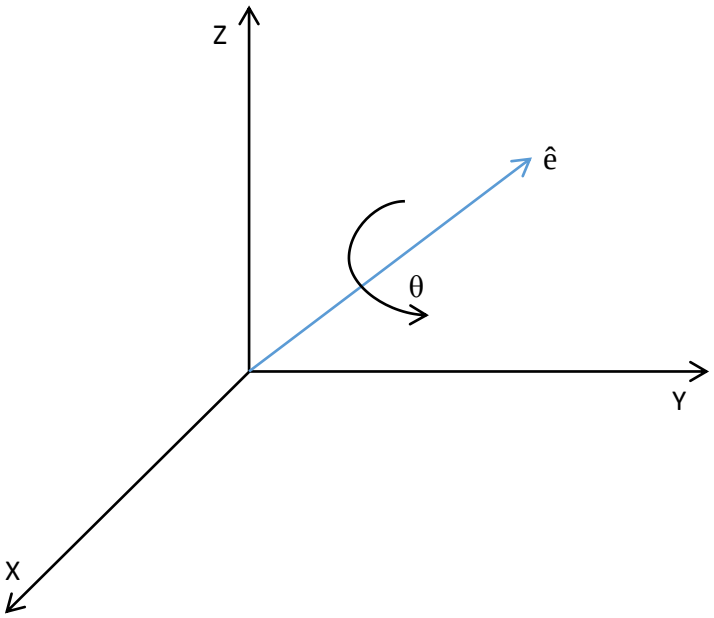

Fig 1. Visualization Of A Rotation Represented By Euler Axis And Angle.

Axis-angle is one of the most easily understood methods for specifying the 3D rotations. However we have to be careful as there is a downside to this method: we cannot directly combine two rotations to give an equivalent third rotation, to do this we need to make use of matrices or quaternions.

For some applications, it is helpful to be able to make a rotation with a given axis. Say we have a unit vector $\boldsymbol{u} = (u_x, u_y, u_z)$, where $u_x^2 + u_y^2 + u_z^2 = 1$, then the rotation matrix by an angle of θ about an axis in the direction of $\overline{u}$ is calculated as:

$$c = cos\,\theta$$

$$s = sin\,\theta$$

$$C = 1 - c$$

rotation matrix as:

$$R = \begin{bmatrix} x \cdot x \cdot C + c & x \cdot y \cdot C - z \cdot s & x \cdot z \cdot C + y \cdot s \\ y \cdot x \cdot C + z \cdot s & y \cdot y \cdot C + c & y \cdot z \cdot C - x \cdot s \\ z \cdot x \cdot C - y \cdot s & z \cdot y \cdot C + x \cdot s & z \cdot z \cdot C + c \end{bmatrix} \quad (4)$$

This can also be written as:

$$R = \cos\theta \cdot I + \sin\theta \cdot [u]_x + (1 - \cos\theta) \cdot u \otimes u$$

where,

$[u]_x$ is the cross product matrix of $\overline{u}$.

I is the identity matrix.

$\otimes$ is the tensor product of $\overline{u}$, given as:

$$u \otimes u = \begin{bmatrix} u_x^2 & u_x u_y & u_x u_z \\ u_x u_y & u_y^2 & u_y u_z \\ u_x u_z & u_y u_z & u_z^2 \end{bmatrix} \qquad [u]_x = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}$$

3.1.3 Quaternions

Quaternions are mathematical notations for representing the orientations and rotations of three dimensional objects. A quaternion is a unit length vector which can be represented as $\overline{\boldsymbol{q}} = (q_x, q_y, q_z, q_w)$ or $\overline{\boldsymbol{q}} = (x, y, z, w)$ in short. Unit quaternions live on a unit sphere $\|\boldsymbol{q}\| = 1$. Antipodal quaternions $\boldsymbol{q}$ and $-\boldsymbol{q}$, as shown in Figure 2, represent the same rotation. Apart from this ambiguity, quaternions offer a unique way to represent rotation. Since rotation in three dimensions can be represented as a combination of a vector $\overline{u}$ and a scalar angle θ (Axis angle representation), quaternions give a simple way to encode this representation in four numbers, and apply the corresponding rotation to a vector. They are simpler and easier to compose than Euler angles. When compared to rotation matrices they are numerically stable and efficient.

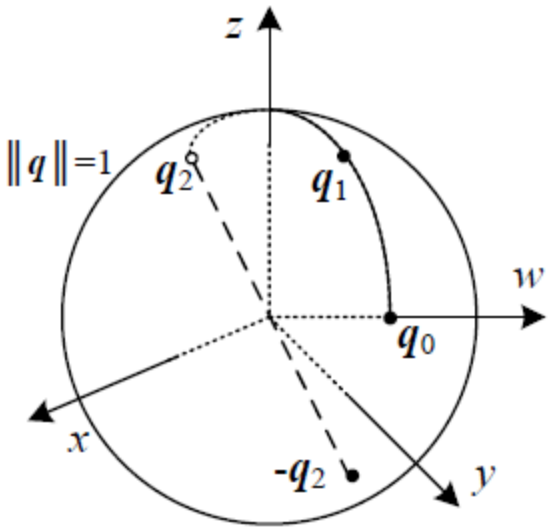

Fig 2. Unit Sphere $\|\boldsymbol{q}\| = 1$. Shows A Smooth Trajectory Through Quaternions $\boldsymbol{q_0}, \boldsymbol{q_1}, and\ \boldsymbol{q_2}. \boldsymbol{q_2}\ and\ -\boldsymbol{q_2}$ Represent The Same Rotation.

Derivation of quaternions from axis angle representation can be easily achieved using the formula:

$$\boldsymbol{q} = \left(\sin\left(\frac{\theta}{2}\right) \cdot \hat{n}, \cos\left(\frac{\theta}{2}\right)\right)$$

where,

$\hat{\boldsymbol{n}} = (n_x, n_y, n_z)$ is the rotation axis and θ is the rotation angle. The different quaternion components (q_x, q_y, q_z, q_w) are calculated as:

$$q_x = n_x * sin\left(\frac{\theta}{2}\right)$$

$$q_y = n_y * sin\left(\frac{\theta}{2}\right)$$

$$q_z = n_z * sin\left(\frac{\theta}{2}\right)$$

$$q_w = \cos\left(\frac{\theta}{2}\right)$$

Similarly we can convert a quaternion $\overline{\boldsymbol{q}} = (q_x, q_y, q_z, q_w)$ to its equivalent axis angle $(\hat{\boldsymbol{n}}, \theta)$ representation using the formula:

$$\theta = 2 * \cos^{-1}(q_w)$$

$$n_x = q_x \,/\, sqrt(1 - q_w * q_w)$$

$$n_y = q_y \,/\, sqrt(1 - q_w * q_w)$$

$$n_z = q_z \,/\, sqrt(1 - q_w * q_w)$$

Key point to remember is that when the above equations are used for calculations they must be checked for singularities as axis-angle has two singularities at $\theta = 0$ degrees and $\theta = 180$ degrees.

Quaternions are popular parameterization due to many reasons, some of them are:

1. This representation is more compact and is less susceptible to round-off errors.
2. The expression for the rotation matrix involves no trigonometric functions.
3. Combination of two individual rotations which is represented as quaternions is simple when quaternion product is used.

3.2 Human Motion Capture

The capture of human activities or motions is achieved by mainly *active sensing* or *passive sensing*. Active sensing is achieved by placing devices on the person and in well controlled environments where there are devices to transmit or receive signals from the devices placed on the person. Passive sensing is done without the help of these devices. This is achieved with the help of natural signal sources such as visual light or an infrared beam. This type is very well suited for environments where placing devices on the person is not possible. The operational complexity of the two methods is the main trade off.

The thesis presented here employs passive method to capture the motion of the person. This is implemented by using the Microsoft Kinect camera, which projects a grid of IR rays. The pattern of the IR grid is captured by the IR camera, which translates this to a 3D map of the scene. The advantage of the kinect over 2D video camera is in its ability to provide the 3D map of the scene.

3.2.1 Microsoft Kinect

The kinect is a depth sensing input device from Microsoft. The software technology of kinect is developed by Rare, a subsidiary of Microsoft game studios which is owned by Microsoft, while the range camera is developed by PrimeSense, an Israeli developer. The kinect sensor is a horizontal bar which is connected to a small base that is fixed to a motorized pivot. The features of the kinect device include a RGB camera, depth sensor and a multi-array microphone, shown in Figure 3. The device is capable of providing a full-body 3D motion capture, facial recognition, and voice recognition.

The kinect captures video data in 3D under any ambient lighting condition with the help of an infrared laser projector combined with a monochrome CMOS sensor. Depending on the resolution of capture the various sensors output video at a frame rate of ~9 Hz to 30 Hz. The kinect is capable of providing a video output at a resolution of 1280x1024 (at a lower frame rate), while the default is a RGB video stream of 8-bit VGA resolution, i.e. 640 x 480 pixels as shown in Figure 4. The video stream of the monochrome depth sensor is in VGA resolution with 11-bit depth and provides 2048 levels of sensitivity, shown in Figure 5. The kinect device is also capable of streaming the output from the IR camera directly, i.e. before it is converted to a depth map, the resolution of this stream is 640 x 480 or 1280 x 1024 (with a lower frame rate). The effective range for tracking a person is approximately 2.3 – 19.7 ft. The field of the device is 57° horizontally and 43° vertically. The motorized pivot provided in the device can tilt the senor up to 27° either up or down.

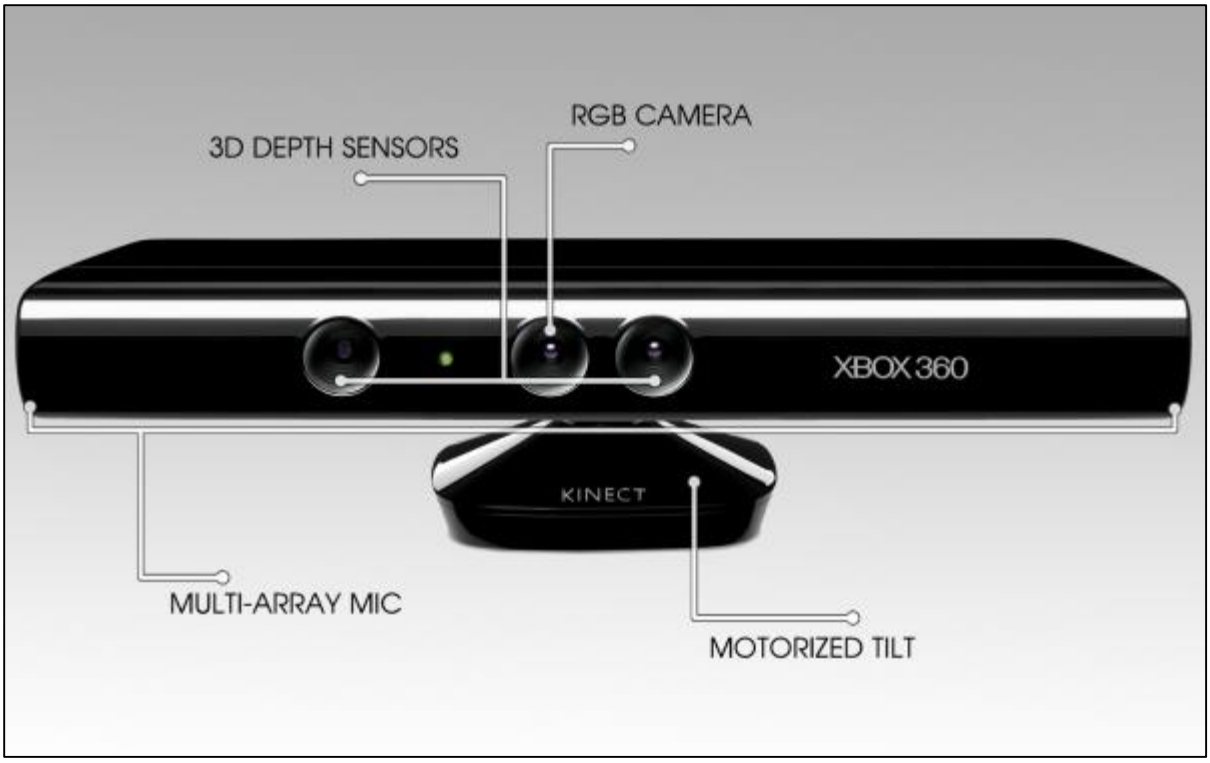

Fig 3. Overview Of A Microsoft Kinect And Its Features.

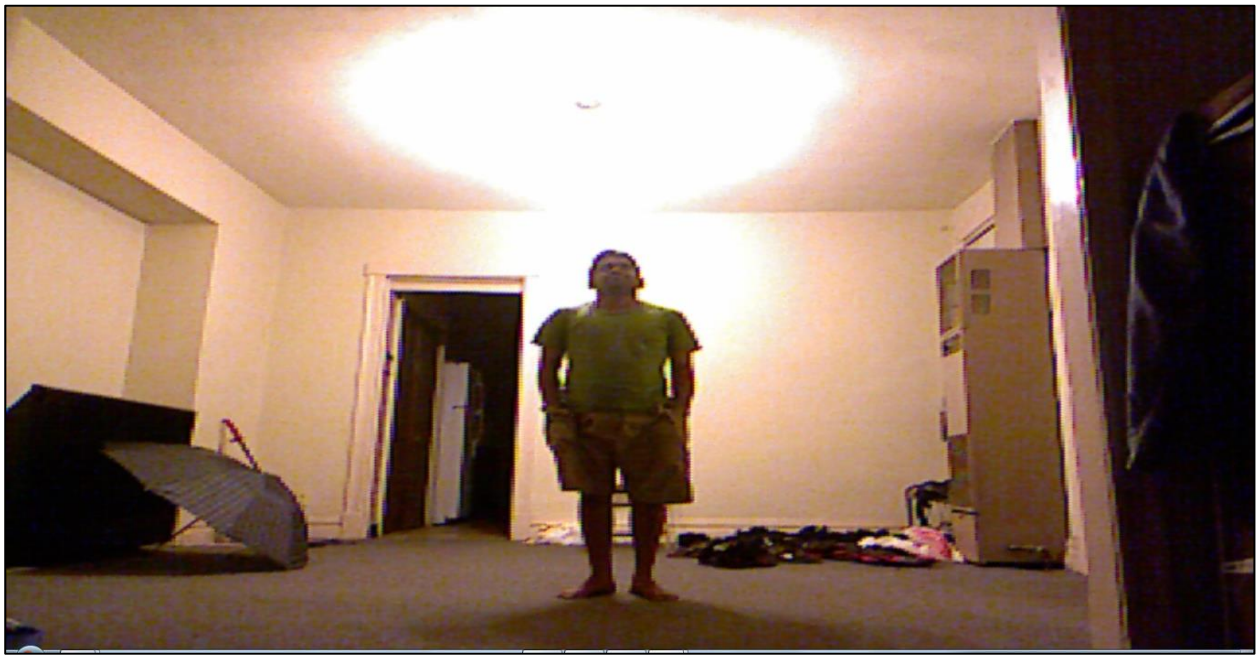

Fig 4. Output From The RGB Camera Of The Kinect.

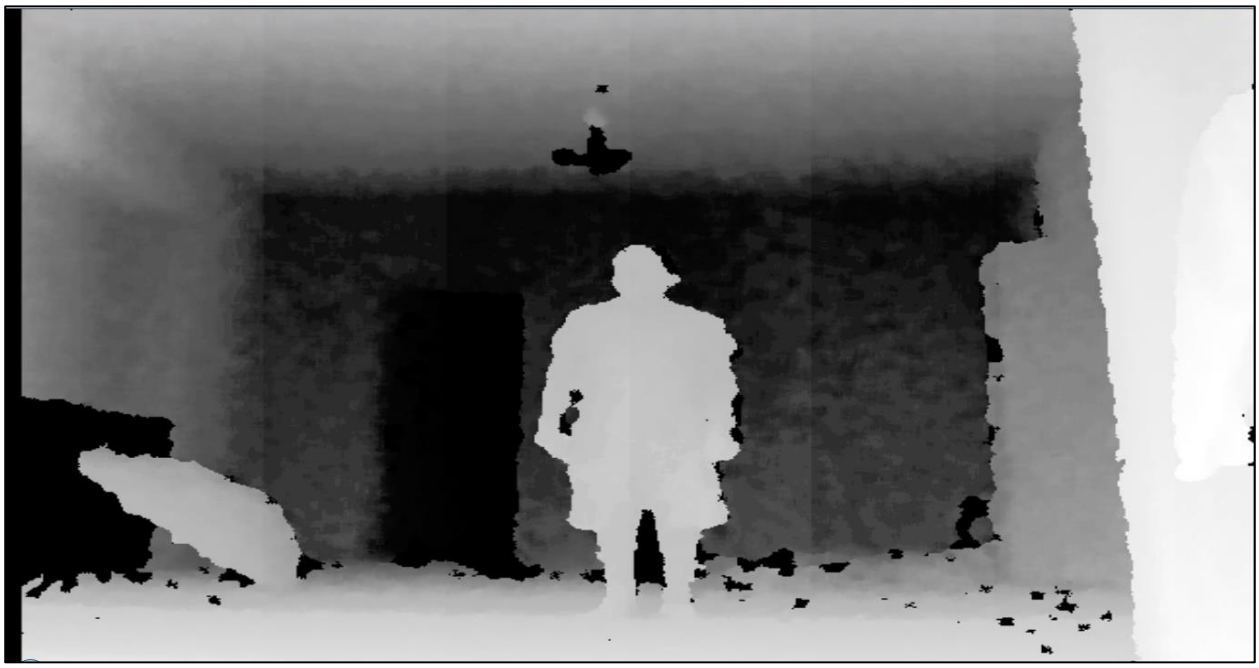

Fig 5. Output From The Depth Camera Of The Kinect.

3.3 Software Packages:

This section presents a brief explanation of the different software packages that is used for the extraction of the human skeleton.

3.3.1 Microsoft Kinect SDK:

The kinect Software Development Kit or in short the kinect SDK is a software package or driver that is used by the kinect camera to integrate the computer to the kinect hardware and use the different features of the camera. This development kit is available from the Microsoft website. The tool kit has many functionalities and one of it is the human skeleton tracking feature. The skeletal information obtained from this tool kit consists of 20 joints per skeleton. These joints correspond to the different joints of the human body, Figure 6.

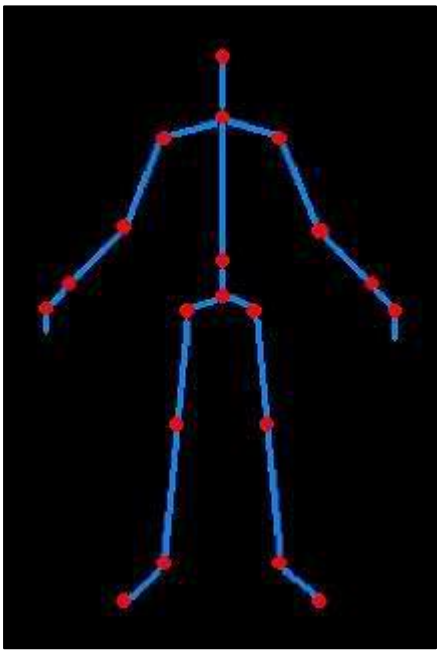

Fig 6. Skeletal Information From The Kinect SDK.

3.3.2 OpenNI and NITE:

The OpenNI and NITE packages are open-source packages that are similar to the SDK from Microsoft. Our work makes use of these packages instead of the SDK for no other reason other than the simplicity in implementation of these packages. The skeleton obtained from these packages contain less number of joints when compared to the SDK. The skeletal information obtained from this package contains 15 body joints per skeleton as in Figure 7. Our work utilizes all the 15 joints for training and testing, however if due to occlusions some of the joints are not captured we can still achieve high recognition if reasonable number of joints were captured. Figure 8, shows the skeleton of the person once it has been extracted from the depth sequence.

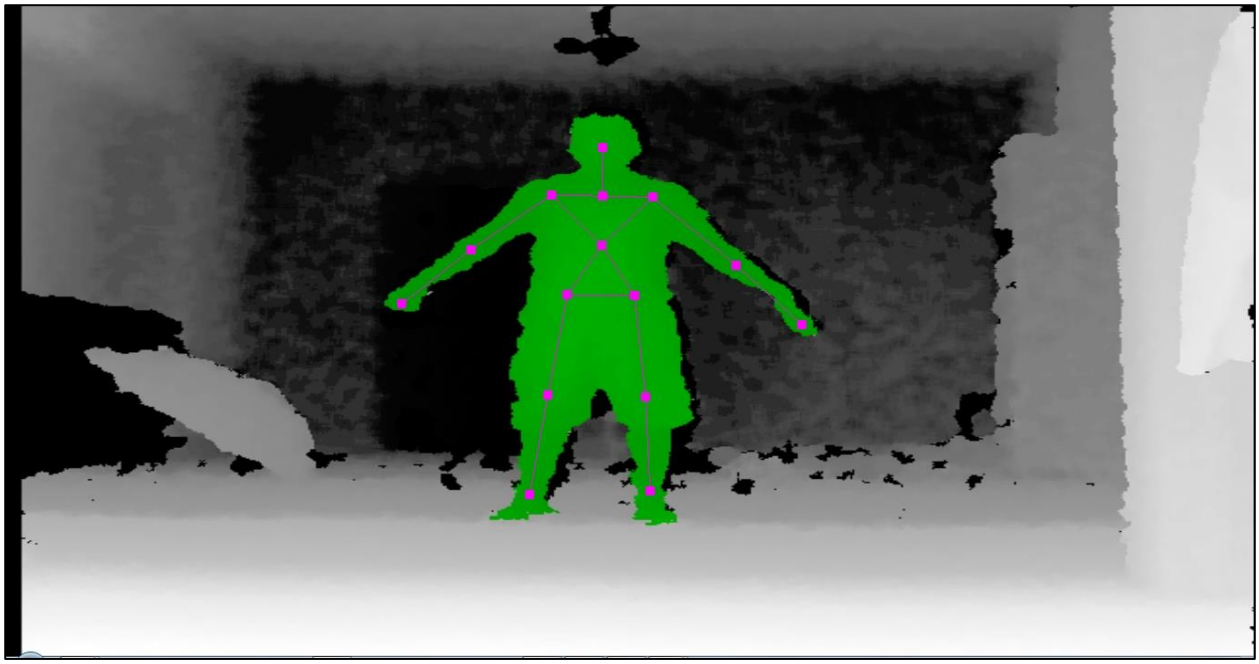

Fig 7. Depth And Skeletal Information Obtained From OpenNI And NITE.

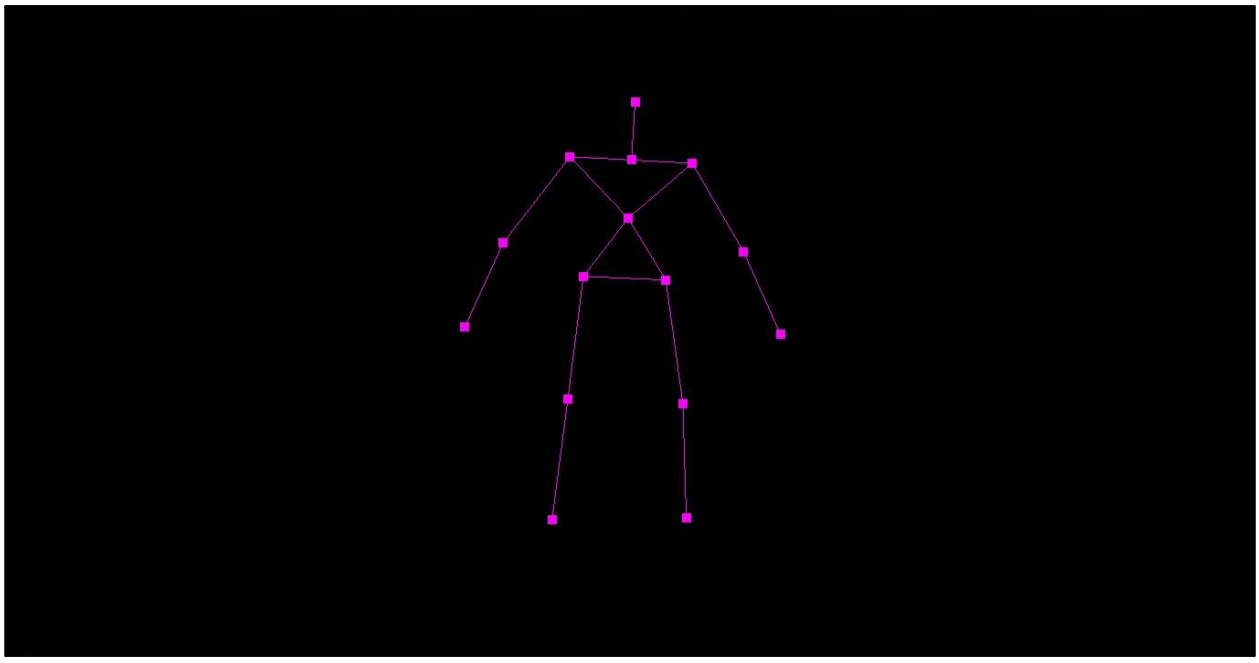

Fig 8. Skeleton Extracted From The Depth Image.

4. OVERVIEW OF THE FRAMEWORK

This section presents an overview of our recognition framework and details our feature selection stage. The flow diagram below presents a general overview as to how our approach works.

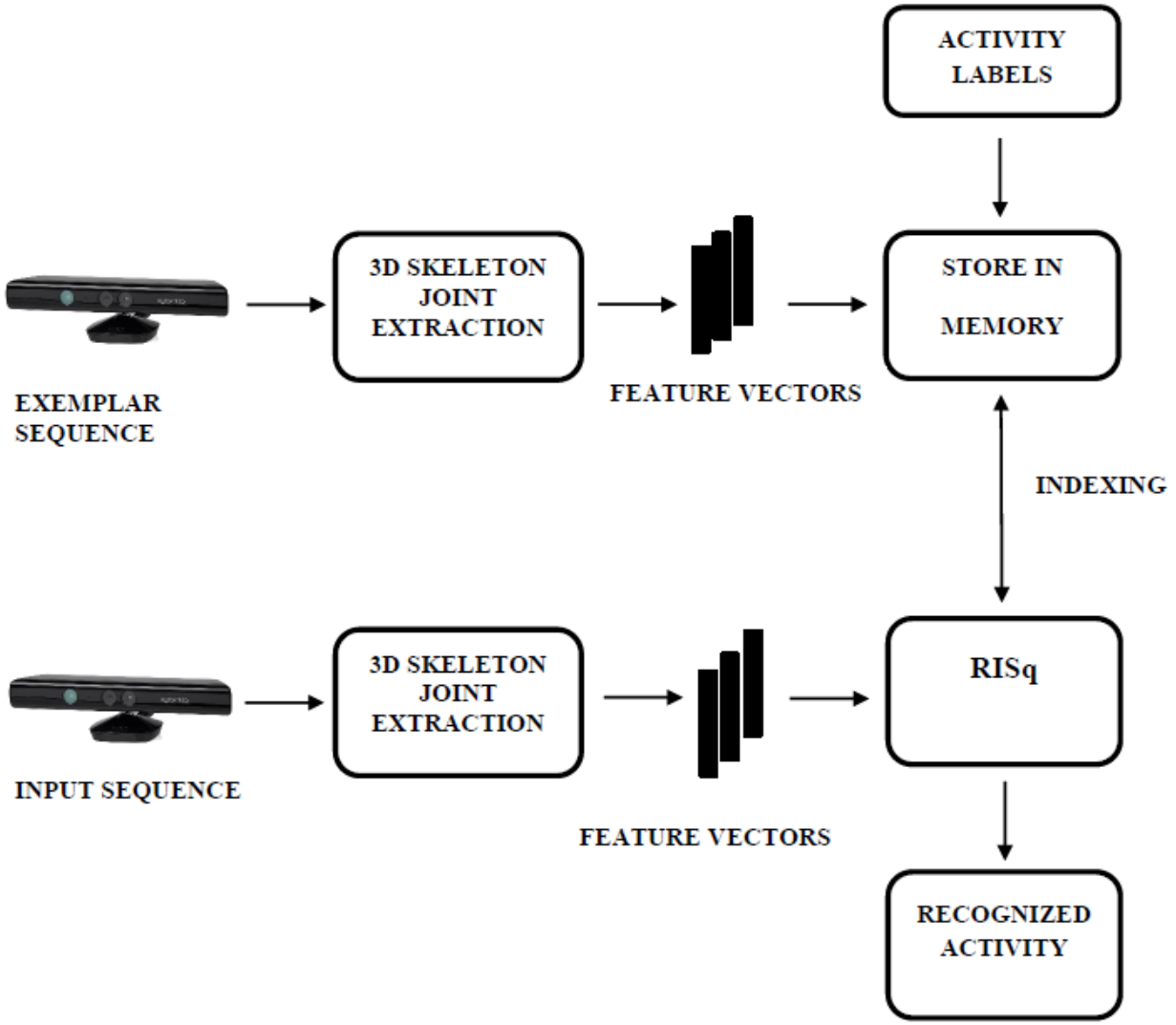

Fig 9. Overall Architecture Of Our Activity Recognition Framework.

4.1 Overview:

The approach shown in Figure 9, can be divided into two stages: *training* and *recognition*. The training phase has the following steps:

1. A video sequence of given activity is captured using the kinect. This video contains the RGB, depth, and skeletal information.
2. Skeletal information obtained from the previous step is used to obtain the joint location and orientation information. Details about the skeletal information is presented in the next section.
3. After processing this information, feature vectors are generated which are stored in the memory along with the activity label.

The recognition stage initially has similar steps as that of testing once feature vectors are obtained RISq is used for recognition.

1. The input sequence (the unknown activity sequence that is to be determined) is captured using the kinect. Captured sequence has RGB, depth and skeletal information.
2. Joint location and orientation information is extracted from the skeletal information obtained from the previous step.
3. Processing of this information leads us to the desired feature vectors (a sequence of vectors which best represent the original sequence).
4. This vector sequence is indexed into the memory bank containing the different activity sequences. A nearest neighbor search is performed and the best 10 nearest neighbors are obtained.

5. Last step is to find an optimal sequence of votes for each class, i.e. we find a maximum weighted bipartite graph matching between the input samples and the retrieved exemplar samples. We make use of dynamic programming to efficiently solve this problem.

The skeletal information obtained from the kinect is obtained as the location of the 15 different body joints (head, neck, torso, two joints for each of the shoulder, elbow, hand, hip, knee, and foot joints) mapped on to the person using the NITE package in the world environment. These locations are the $x, y, and\ z$ coordinates of the body joints in the world with respect to the kinect coordinate system. We represent these coordinates as a 45 dimensional vector for each frame. The kinect output 30 frames per second, hence for each second of an activity we have 30 vectors to represent that activity.

Example let $V_{t_1} = v_1, v_2, v_3, \dots\dots\ v_{15}$ represent the vector for the first frame of the captured activity. where $v_1, v_2, v_3, \dots\dots\ v_{15}$ are numbers which each represent the $x, y, and\ z$ coordinates of the 15 different body joints, example $v_1 = (x, y, z)$ is the coordinate of the head joint, $v_2 = (x, y, z)$ is the coordinate of the neck joint, etc. V_{t_1} is the vector which comprises the set of numbers representing the coordinates of all the joints at time t_1. Hence for a given activity of length n the vector representation will be:

$$\boldsymbol{V(t)} = \begin{Bmatrix} \boldsymbol{V_{t_1}} \\ \boldsymbol{V_{t_2}} \\ \boldsymbol{V_{t_3}} \\ \vdots \\ \boldsymbol{V_{t_n}} \end{Bmatrix} = \begin{Bmatrix} v_{t_1}^1 & v_{t_1}^2 & & v_{t_1}^{15} \\ v_{t_2}^1 & v_{t_2}^1 & \cdots & v_{t_2}^{15} \\ v_{t_3}^1 & v_{t_3}^1 & & v_{t_3}^{15} \\ & \vdots & \ddots & \vdots \\ v_{t_n}^1 & v_{t_n}^2 & \cdots & v_{t_n}^{15} \end{Bmatrix}$$

$\boldsymbol{V(t)}$ collectively represents the activity sequence of length n. $\boldsymbol{V_{t_1}}, \boldsymbol{V_{t_2}}, \ldots \boldsymbol{V_{t_n}}$ each $\boldsymbol{V_{t_i}}$ represents the 45 dimensions containing the joint coordinates at time intervals $t_1, t_2, \ldots t_n$ respectively.

4.2 Video capture, Feature selection and training:

This section details the first stage of our approach, i.e. capturing the video sequence from the kinect, selecting the features and training. We first explain how we capture the video sequence that contains 3D skeletal information.

4.2.1 Video capture:

We use the kinect camera along with the OpenNI and NITE package to capture the video. OpenNI provides the necessary software to integrate the kinect to the computer, NITE provides the skeletal tracking capabilities. The flow diagram shown in Figure 10, provides some light on how we capture the video. Understanding the flow diagram is very easy, the steps followed are:

1. Initialize the kinect and check if initialization was successful.
2. Once initialization is a success we start tracking, we place a check here to ensure that tracking was successful.
3. Green light from *tracking check* means that we start recording the activity sequence until a desired time frame.
4. Extract the skeletal information from this video, this is achieved using the open source NITE package.
5. We store this extracted information as text/CSV file, this file contains the locations of all the skeletal joints with respect to the kinect coordinates.

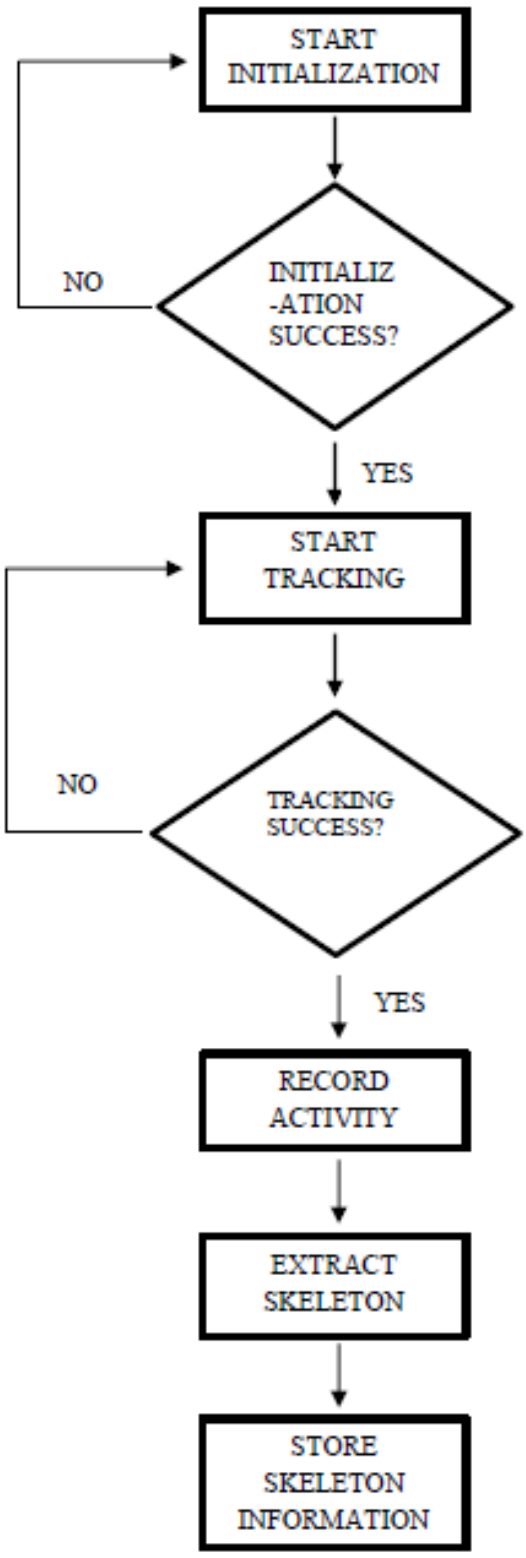

Fig 10. Flow Diagram Of The Skeleton Extraction Method.

4.2.2 **Feature Selection:**

The feature selection stage is where we compute and select the features (points of interest, in our case the joint coordinates), which best represents the activity sequence. Before proceed to the features selection stage we first define the joints and their coordinate system. The joint information obtained from the kinect is with respect to the kinect coordinate system. This coordinate system is a right-handed system that places the kinect at the origin with the positive z-axis extending in the direction the kinect is pointed, the positive y-axis extends upward, and the positive x-axis extends to the left. The depth data from the kinect is in terms of millimeters. Figure 11, illustrates this coordinate system.

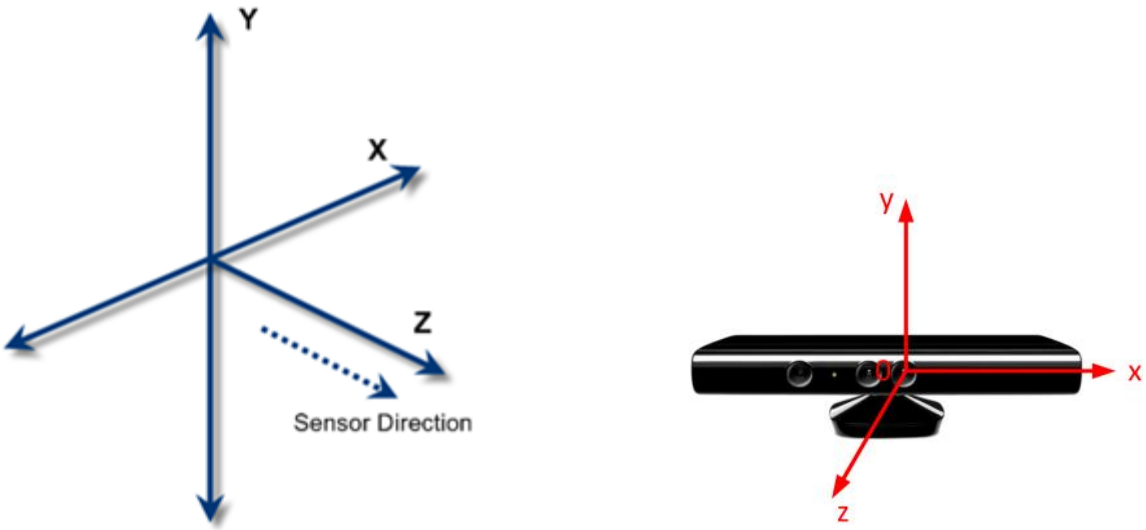

Fig 11. Illustration Of The Kinect Coordinate System.

Next, with this knowledge about the kinect coordinate system, we illustrate with a figure how the different joints appear in the kinect when using OpenNI and NITE packages. Figure 12, illustrates the skeleton obtained from using these packages and also the 15 different skeletal joints. The skeleton obtained comprises of a head, neck, torso joints along with the right and left joints for each of the shoulders, elbows, hands, hips, knee and foot. The joints are obtained as a set of $x, y, and\ z$ coordinates, the naming in the figure is only shown for illustration.

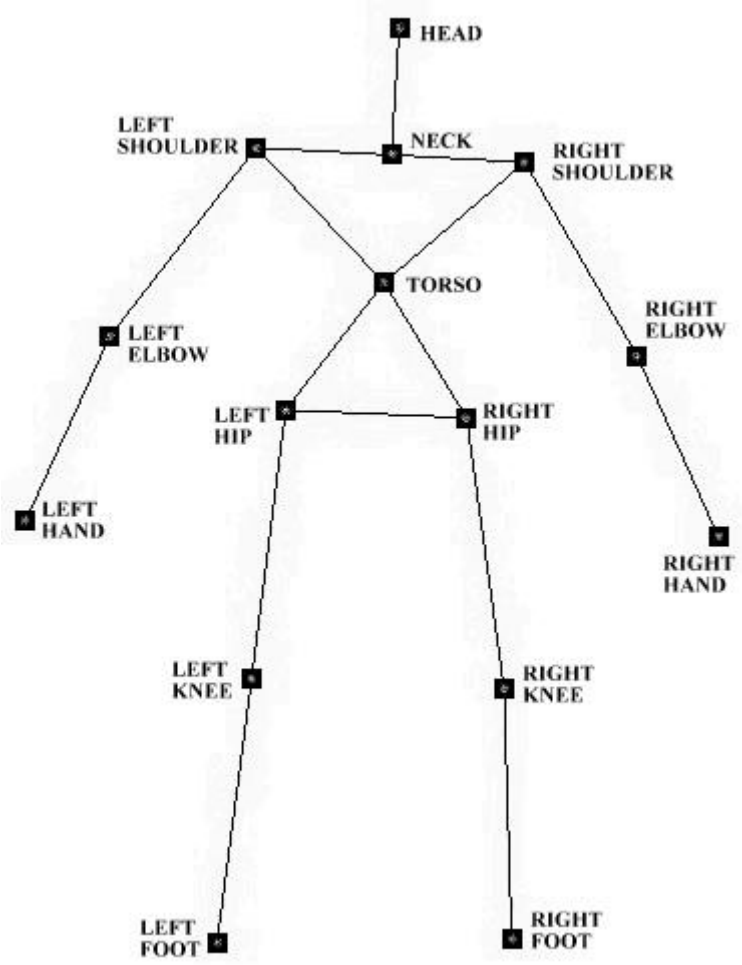

Fig 12. Illustrates The Different Skeletal Joints.

The joint coordinates obtained from the previous step is in terms of the kinect coordinate system, however to make our system invariant to the location of the kinect in the real world we transform these coordinates to new system called the body centered coordinate system. In the body centered coordinate system we consider the center of the coordinate system to coincide with the torso joint of the body and all other joints are expressed relative to this center. Here the positive $x - axis$ passes through the right side part of the body, the positive $y - axis$ projects orthogonally upward from the positive $x - axis$ (in the direction of the head joint), the positive $z - axis$ projects orthogonally outwards from the torso. Figure 13, illustrates this coordinate system, with the positive $x, y, and\ z\ axes$ marked in *green, blue, and red* colors.

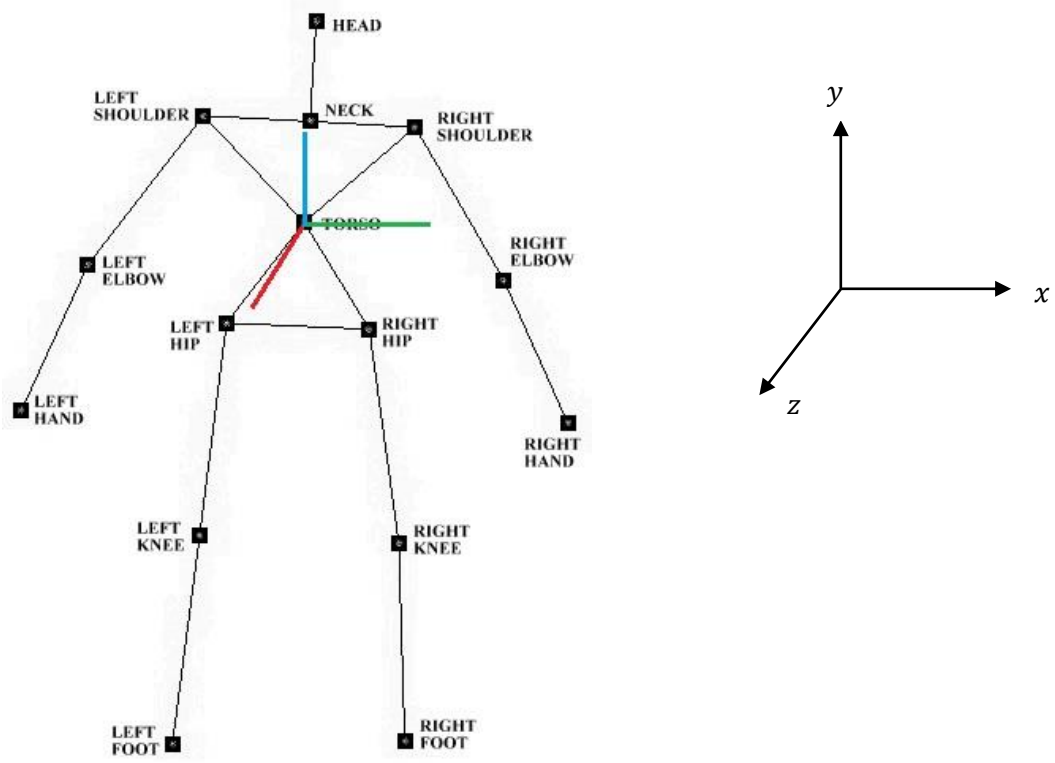

Fig 13. Illustration Of The Body Centered Coordinate System.

To transform the joint coordinates from the kinect coordinate system to the body centered coordinate system we perform two major operations: Translation and Rotation, theoretical aspects about these operations and how they are performed in general are explained in chapter 3. Here we explain how we implement these operations in our work.

Translation is the process of moving every point a constant distance in a specified direction. This can also be interpreted as an addition of a constant vector to every point, which can also be interpreted as shifting the origin of a coordinate system. In our approach we translate the coordinates relative to the kinect coordinate system to the body centered coordinate system, this can also be interpreted as translating the origin of the kinect coordinate system to the origin of the body centered coordinate system (the torso). We achieve this by multiplying the $x, y, and\ z$ coordinates of the body joints (relative to the kinect coordinate system) by a translation matrix that contains in its last column the negation of the $x, y, and\ z$ coordinates of the torso joint (also relative to the kinect coordinate system).

Suppose $T_x, T_y, and\ T_z$ are the $x, y, and\ z$ coordinates of the torso joint respectively, and $H_x, H_y, and\ H_z$ are the $x, y, and\ z$ coordinates of the head joint respectively, both of which are relative to the kinect coordinate system and for one frame of the activity sequence. Then to translate the head joint from the kinect coordinate system to the body centered coordinate system, we multiply the coordinates of the head joint with a translation matrix that contains the negation of the coordinates of the torso joint in its last column, i.e. we substitute these in equation 3,

$$Head\ joint\ coordinates\ = \begin{bmatrix} 1 & 0 & 0 & -T_x \\ 0 & 1 & 0 & -T_y \\ 0 & 0 & 1 & -T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} H_x \\ H_y \\ H_z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} H_x - T_x \\ H_y - T_y \\ H_z - T_z \\ 1 \end{bmatrix} = \overline{\boldsymbol{H}} - \overline{\boldsymbol{V}}.$$

The resultant vector obtained from the above operation is the coordinates of the head joint which is relative to the body centered coordinate system. Similarly we perform the above operation to all the joint coordinates, and in all frames of the activity sequence to obtain the same activity sequence in which all the coordinates of the body joints are relative to the body centered coordinate system.

Rotation is the next operation which is performed on all the joint coordinates, to achieve this we use the axis angle representation of a rotation matrix. Representing a rotation matrix in terms of axis angle requires a rotation axis and a rotation angle, details about these are explained in chapter 3. We consider the upper part of the torso, i.e. the vectors joining the two shoulders and the torso, which is in the form of a triangle, see Figure 12. Using this we calculate the normal at the torso as the cross product between the vectors joining the two shoulders and the torso (the resultant vector projects outwards from the skeleton). Using this normal and the actual $z - axis$ $(0, 0, 1)$ we calculate the rotation axis as the cross-product between these vectors and the rotation angle as the dot-product between these two vectors. Once we have the rotation axis and angle we substitute this in the equation for rotation matrix representation using axis angle given by equation 4 as:

$$R = \begin{bmatrix} x \cdot x \cdot C + c & x \cdot y \cdot C - z \cdot s & x \cdot z \cdot C + y \cdot s \\ y \cdot x \cdot C + z \cdot s & y \cdot y \cdot C + c & y \cdot z \cdot C - x \cdot s \\ z \cdot x \cdot C - y \cdot s & z \cdot y \cdot C + x \cdot s & z \cdot z \cdot C + c \end{bmatrix}$$

Where,

Rotation axis = $[x \quad y \quad z]^T$ and rotation angle = θ

$c = \cos\theta, s = \sin\theta, and\ C = 1 - c.$

This rotation matrix *R* is multiplied to all the joint coordinates which have been translated to the body centered coordinate system. The resultant set of coordinates which are obtained after these combined operations are the coordinates which are relative to the body centered coordinate system and invariant to the kinect's location in the real world.

Scaling is an optional operation performed mainly to maintain consistency of axes and skeletal figure during plotting and illustration of the skeleton. To achieve this we multiply all the joint coordinates with a scaling matrix *S*. The resultant coordinates is a scaled version of the original set. A scaling matrix *S* with uniform scale factor is represented as:

$$S = \begin{bmatrix} Scale\_Factor & 0 & 0 \\ 0 & Scale\_Factor & 0 \\ 0 & 0 & Scale\_Factor \end{bmatrix} * \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} \qquad (5)$$

$V_x, V_y, and\ V_z$ are the $x, y, and\ z$ vector coordinates of the different joints. The resulting vector from the above operation is the scaled version of the original set of coordinates. $Scale\_Factor$ in our case is the resultant of a division operation between a constant and the distance between the two shoulder joints.

The pseudo code given below explains the different operations and the manner in which they are executed:

1: $\{N = length\ of\ the\ activity\ or\ number\ of\ frames\}$

2: $for\ 1\ to\ N\ do$:

3: $plot\ skeleton$

4: $calculate\ translation\ matrix\ T\ from\ equation\ 3$

5: $multiply\ all\ coordinates\ with\ matrix\ transformation\ matirx\ T$

6: $calculate\ rotation\ matrix\ R\ from\ equation\ 4$

7: $multiply\ all\ coordinates\ with\ rotation\ matirx\ R$

8: $calculate\ scaling\ matrix\ S\ from\ equation\ 5$

9: $multiply\ all\ coordinates\ with\ matrix\ S$

10: $plot\ transformed\ skeleton$

11: $Store\ all\ the\ coordinates\ in\ a\ matrix$

12: $end\ for$

Figure 14 and Figure 15 illustrates the original and change in the skeleton after the transformations. These calculations were carried out and tested on MatLab R2013b.

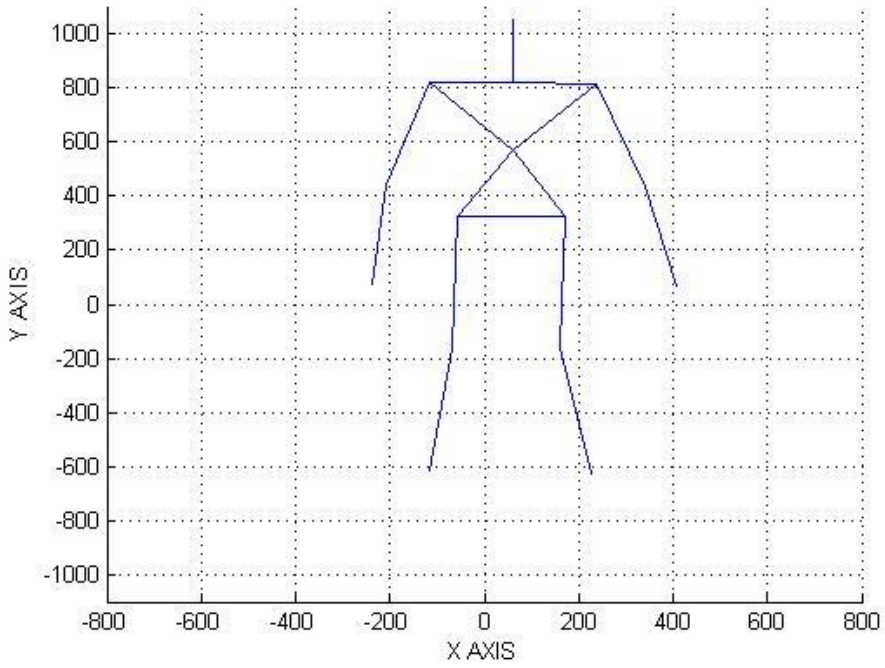

Fig 14. Illustrated Is The Skeleton Obtained Directly From The Kinect.

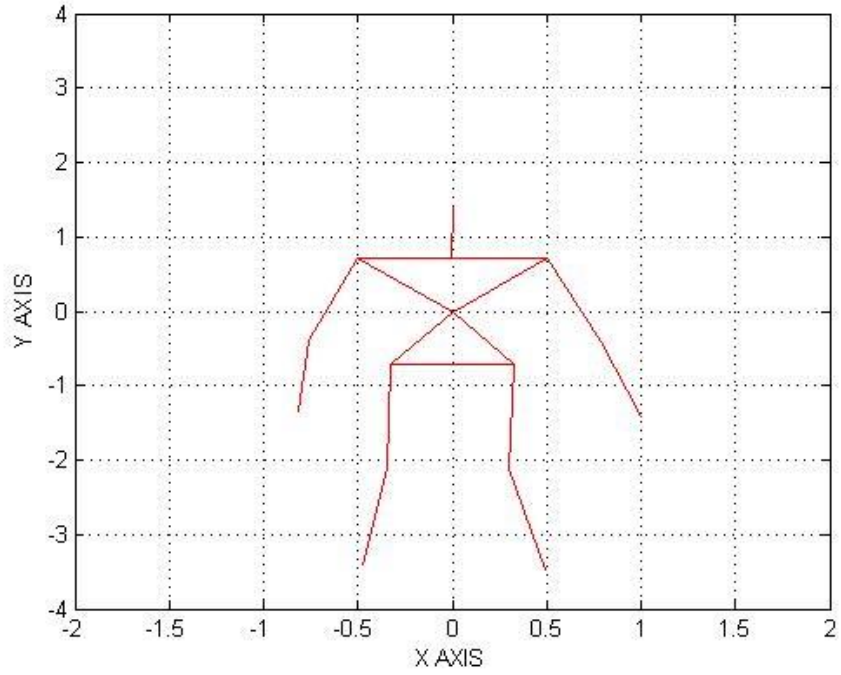

Fig 15. Illustrates The Skeleton After Transformations.

The set of coordinates obtained after the translation and rotation operations is stored as a $(m \times 45)$ matrix, where m is the length of the activity sequence and the number 45 represents the 15 joint coordinates consisting of 3 dimensions $x, y, and\ z$. We use this matrix as the feature input to our recognition algorithm. The features lengths that are used for training generally span 3 to 4 seconds. The features used for testing have generally larger lengths than the lengths of the training samples. These features are used later on for the recognition of the test sequence which is explained in the next section.

4.2.3 Training

The training procedure for our algorithm is very simple. It consists of storing all the features of the training sequences in an underlying data structure. In this method we use an extension of the binary tree to multiple dimensions, the kd-tree. The kd-tree is an efficient data structure for the retrieval of nearest-neighbors in multiple dimensions. A simple kd-tree can be built in $O(n)$ space and $O(n * log(n))$ time. Figure 16, illustrates a kd-tree in two dimensions. We organize all the data points in the tree by splitting the tree on different dimensions at each level of the tree.

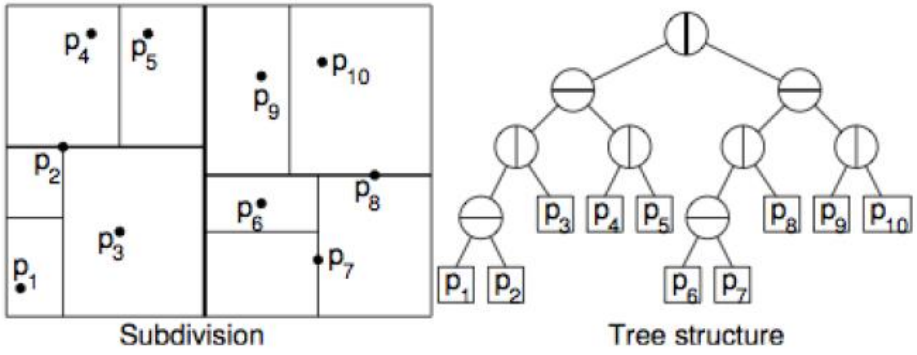

Fig 16. Illustration Of A Two Dimensional KD-Tree.

The computational complexity for inserting and deleting a point in a balanced k-d tree where n points are stored will be $O(n \log n)$. In a balanced k-d tree if n points are stored, the computational complexity for a k-nearest neighbor or range search is $O\left(n^{1-\frac{1}{d}} + p\right)$, where p represents the number of points returned [8]. This result is true if $n \gg 2^D$, where D is the number of dimensions, if not this will be no better than an exhaustive search where most of the tree is to be searched. The information stored in kd-tree data structure contains the following information:

Node Entry	**Description**
Feature Vectors	The normalized vector sequence representing the activity.
Model Index	The index of the model activity. Another table[1] is constructed to determine which class (walking, standing, sitting, etc.) the input sequence belongs to.

Table I. Data Corresponding To Each Node Of The Kd-Tree.

[1] This table acts as a look up table for recognizing the activities after RISq, using the index number the activity class can be determined using this table.

In our database we will then have all the activity sequences represented as a matrix for each of that activity. In our work we have 10 different activities: walking, sitting on a chair, standing, jumping, chopping, drinking, stirring, talking over the phone, working on the computer, and writing on a whiteboard. The first four activities were the activities recorded in our lab using the kinect by following the steps explained in section 4.2.1. This was performed using 3 male subjects: the lab data contains on an average 30 activities per class. The remaining set of activities were obtained from the Cornell University's robot learning lab [27]. The dataset used from them is the CAD-60 dataset which contains 12 activities: rinsing mouth, brushing teeth, wearing contact lens, talking on the phone, drinking water, opening pill container, cooking (chopping), cooking (stirring), talking on couch, relaxing on couch, writing on whiteboard, and working on computer. These were performed by 4 subjects: two males and two females.

5. RECOGNITION BY INDEXING AND SEQUENCING (RISq)

This section details the recognition stage that is used in our approach. Indexing and sequencing are the two stages that constitute our recognition algorithm. The next sections explains how we work through these stages, the thresholding stage is also presented here.

5.1 Indexing and Sequencing

The indexing stage is the first step in our recognition framework. In this step each of the input samples to be recognized as a part of the activity are indexed into the underlying data structure from the previous stage. A vote is assigned to the k-nearest neighbors that are retrieved for each of the input samples based on a distance function. The k-nearest neighbors search was chosen as it returns a fixed number of neighbors, thus leading to a constant factor in the time complexity of the algorithm. The distance functions used can be Euclidean, Mahalanobis distance or cosine similarity, for our approach the Euclidean and cosine similarity were used, tests done using either of them yielded similar results. This stage is parallel since it considers all samples from the input sequence and all the stored training activity exemplars at the same time. The set of votes obtained from indexing gives us the similarity score, this is a matrix containing the similarity scores between the input sample and the training exemplars. The labels associated with these scores are not taken into consideration during this stage but at the end of the sequencing stage.

The next step is to find a maximum weighted bipartite graphs matching the samples of the input sequence to that of the samples retrieved from the training sequence from one exemplar each time. A thresholding stage is followed at the end of the sequencing, this is set to a fraction of the highest possible score which the sequencing stage can output. This ensures the weak scores which are below the threshold can be omitted and we only retain ones that are higher or equal to the

threshold. The labels of the scores which pass thresholding are then used to identify which class the unknown input activity belongs to. The score obtained from sequencing is for an exemplar of a single class, when multiple exemplars of the same class are present we have two options: choose the exemplar with the best score or merge the scores of the different exemplars (belonging to the same class). Dynamic programming approach is implemented to solve the problem of finding the maximum weighted bipartite graph, dynamic programming is chosen as the problem exhibits an optimal substructure. There are spatial constraints that are placed during sequencing:

a) There is strict one-to-one matching that is applied.

b) No two matching's can lie across each other.

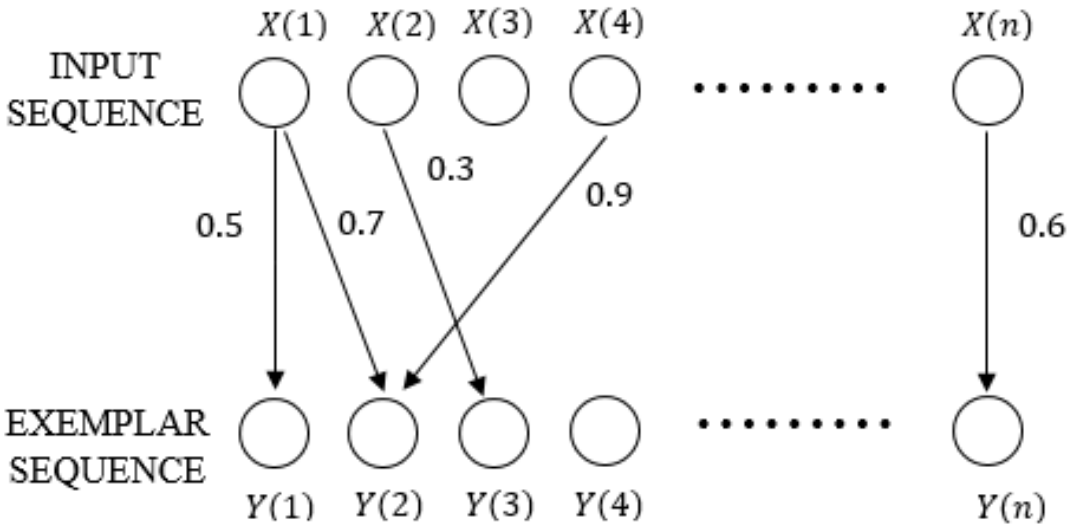

Fig 17. Illustration Of The Sequencing Stage.

The illustration of the sequencing stage shown in Figure 17, has sample from the test X and exemplar Y sequences on the top and bottom respectively. The arrows and the numbers indicate the matching's and the similarity votes associated with these matching's. Let us calculate the matching score for the above example. The maximum matching is {X(1), Y(2)}, {X(2), Y(3)}, {X(4), Y(2)} and {X(n), Y(n)} with a total score of 2.5, but this is not allowed as it violates the constraints that are placed during matching, i.e.: a strict one-to-one matching that is applied, and no two matching's can lie across each other. Hence observing the constraints we get the matching as {X(1), Y(2)}, {X(2), Y(3)} and {X(n), Y(n)} with a total score of 1.6.

The algorithm for recognition can be summarized as: index the input sequence into the database and retrieve the nearest training samples to given input sample. At each training sample we save the optimal sequence ending with that training sample which is voted by the input sample. We apply dynamic programming to efficiently find the optimal sequence. The final step is to sum up the votes for a given class in the optimal sequence to get a total similarity score of the input sequence to one of the training exemplars. After computing all the total scores we select the activity with the highest score as the recognized activity of the input sequence.

The database used for our tests is limited to activities such as sitting, walking, talking over the phone, etc. Even when the input and exemplar sequences are of different lengths RISq efficiently recognizes the input sequence. We also consider the case when there is a combination of these activities like walking and sitting in the same input sequence. In this case:

1) We segment the test sequence into over-lapping segments.
2) Use RISq to obtain a score for each of these segments.
3) Based on the score, we obtain the class label.

Segmenting the input sequence will reduce our problem to the case when we have just a single action. Since RISq can easily match parts of an activity without having the entire activity for classification, the performance is not degraded when we use segmentation. The scores obtained for each of the parts are stored, and depending on the weight of the scores the activity label associated with that score is generated. The segmenting and scoring can be viewed as "sliding" along the length of the input activity, we obtain a high score when part of the input and exemplar sequences overlap with high similarities, and a lower score at places where there are no significant similarities between the input and exemplar sequences.

5.2 Initial Tests:

The simplicity in training the RISq and in computing the similarity scores yields us in high recognition rates. Figure 18, shows us the Receiver Operating Characteristic (ROC) curve for a walk activity which was captured at the lab, from the figure we see that true positive rate is almost a 100%. An ROC is a graphical plot which illustrates the performance of a classification system as its discrimination threshold is varied. The curve is obtained by plotting the true positive rate against the false positive rate at various thresholds. A true positive is when a test result indicates that the result from the classifier is positive when the actual result is also positive. A false positive is when the test indicates that the result from the classifier is positive, but the actual result being negative (i.e. the classifier incorrectly classified the test as a positive when the test was actually a negative).

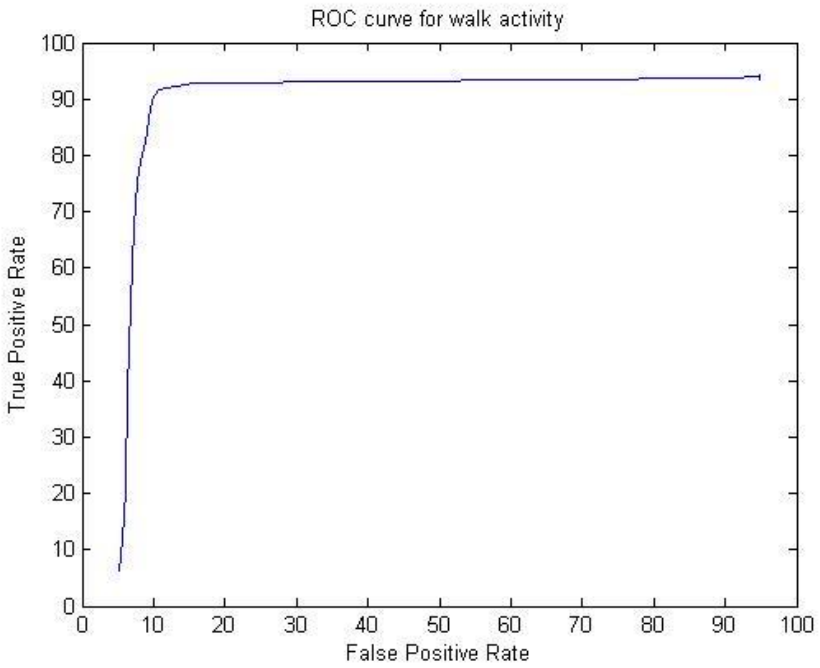

Fig 18. ROC Curve For The Activity: Walk.

A confusion matrix is a specific table layout which is used for the visualization of the performance of an algorithm. A confusion matrix is generally used for two or more classes. For 2 classes the confusion matrix contains two rows and two columns where each cell contains either the true positives, false positives, false negatives, or true negatives. For two classes the table can be viewed as:

	CLASS I	CLASS II	CLASS III
CLASS I	TPR_1	FPR_{21}	FPR_{31}
CLASS II	FPR_{12}	TPR_2	FPR_{32}
CLASS III	FPR_{13}	FPR_{23}	TPR_3

Table II. Illustration Of A Confusion Matrix

In the above illustration the rows of the matrix indicate the input classes and the columns indicate the actual classes. The scores in the other cells indicate the *true positives (TP)* and *false positives (FP)*. A true positive is when a test result indicates that the result from the classifier is positive when the actual result is also positive. A false positive is when the test indicates that the result from the classifier is positive, but the actual result being negative. False negatives (FN) is when the test indicates that the result from the classifier is negative, but the actual result being positive. A true negative (TN) is when the test indicates that the result from the classifier is negative, and the actual result also being negative.

In the matrix we enter the true positive rate and false positive rate. The true positive rate (TPR) is calculated as the fraction of the number of true positives to the total number of positives (P). The false positive rate (FPR) is calculated as the fraction of the number of false positives to the total number of positives.

$$TPR_i = \frac{TP_i}{P_i} = \frac{TP_i}{TP_i + FP_i} \qquad FPR_{ji} = \frac{FP_{ji}}{P_i} = \frac{FP_{ji}}{TP_i + FP_i}$$

$i - \ true\ object/class.$

$j\ = 1,2,3, \dots \dots i-1, i+1, \dots \dots, m\ are\ the\ flase\ objects/class.$

$FP_i\ is\ the\ total\ number\ of\ false\ positves\ when\ the\ true\ class\ is\ i.$

$FP_i = \ FP_{1i} + FP_{2i} + \cdots + FP_{i-1,i} + FP_{i+1,i} + \cdots + FP_{mi}.$

$TP_i\ is\ the\ number\ of\ true\ positives\ classified\ as\ i, when\ the\ true\ class\ is\ i.$

$FP_{ji}\ is\ the\ number\ of\ flase\ positives\ mistakenly\ calssified\ as\ class\ j, when\ the\ true\ class\ is\ i.$

The confusion matrix shown in the illustration is for three classes but the concept can be extended to higher classes and scores obtained is filled in the respective cells. The scores are generally filled as percentage values, where the sum of each row or column add up to 1.

The table below shows a confusion matrix for 3 activities: Sit, Stand, and Walk. The cells in the confusion matrix corresponding to the same class in both row and column contains the true positive, and the cells corresponding to different classes contain the false positive.

	SIT	STAND	WALK
SIT	0.93	0.03	0.04
STAND	0.02	0.94	0.04
WALK	0.01	0.05	0.93

Table III. Confusion Matrix For Sit, Stand And Walk Activities.

In the table we see that the system efficiently recognized the correct activities with a high recognition rate while successfully rejected the other activities. For example when we consider the scores of the input *sit* class, we see that our system recognized the input correctly as sit with a score of 93% and misclassified it as belonging to stand and walk class with scores of 3% and 4% respectively, each row adds up to 100%. The scores indicate that we achieve a very high true positive rate and very low false positive rate, which is critical for a recognition system. In our next chapter we present further results that were obtained when tests were done on our database as well as the database from Cornell University's Robot learning lab [27].

6. RESULTS AND CONCLUSION

In this chapter we provide the experimental results of the application of RISq to human activity recognition on two databases: one comprising of 4 activities: walking, sitting, standing, jumping, these were captured at our lab. The other one comprising of 6 activities: chopping, drinking, stirring, talking over the phone, working on the computer, and writing on a whiteboard, these were obtained from the Cornell University's robot learning lab. It is difficult to validate the performance of RISq with other state-of-the art methods due to various reasons, a few being: there is a very limited number of human activity datasets available, the format of these datasets are very different, since many of them use HMM training the algorithm is very time consuming, etc. Here we provide ROC curves for some of the activities and confusion matrices for both the datasets. In the last part of this chapter we have the conclusion, where we summarize our work and also mention the highlights about our approach and also mention the drawbacks while suggesting the improvements that can be done to overcome these drawbacks and also achieve higher recognition rates.

The dataset of activities which was captured at our lab is of a total of 4 activities, these were captured using 3 male subjects and each of them performed the activities several times and in various pose and locations (confined to the same room however). On an average each activity is about 10 to 12 seconds long and our dataset contains average of 30 activities per class. The dataset of Cornell we have used is the CAD-60 dataset, and this is much larger than our dataset and comprises a total of 12 activities of which we have used 6 activities. These were performed by a total of 4 subjects: two male and two female.

6.1 Results:

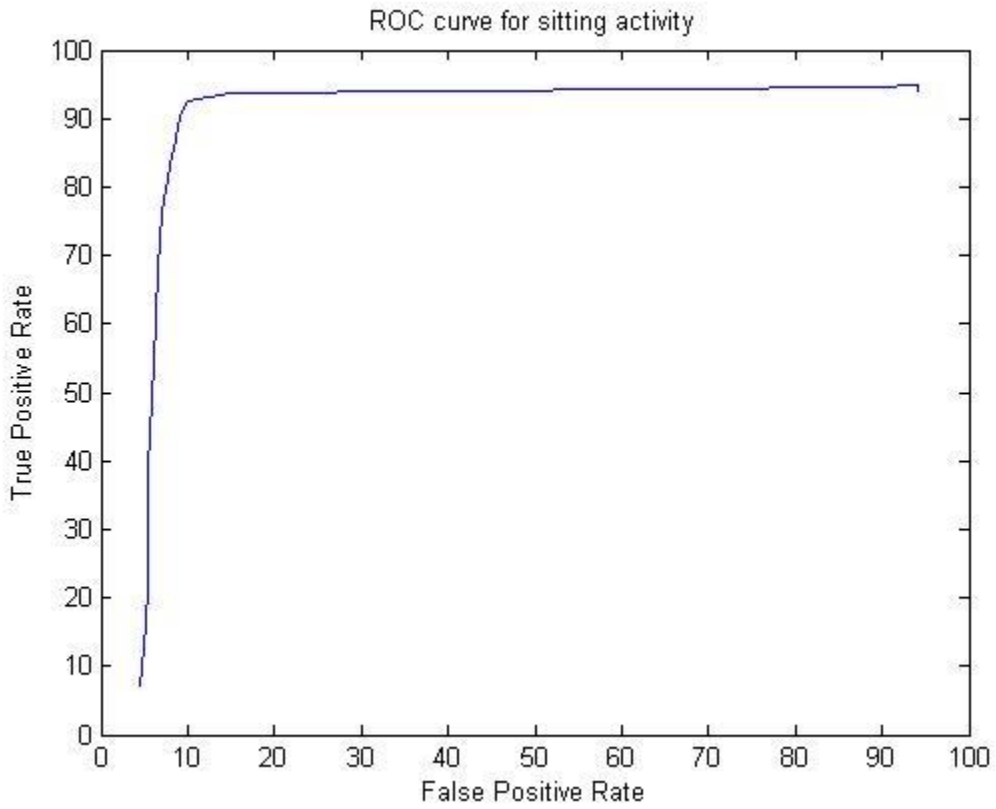

Fig 19. ROC Curve For The Activity: Sit.

This is the ROC curve for the sitting activity from our dataset. The curve is obtained by varying the threshold at every step and noting the true positive rate and false positive rate. From the figure it is evident that the recognition rate is high, by using the RISq. The ROC curve in Figure 20 and Figure 21 are for the chopping and writing activities, taken from the Cornell CAD-60 dataset. RISq is used to obtain this curve, the recognition rates are high and a confusion matrix is also presented to illustrate the results.

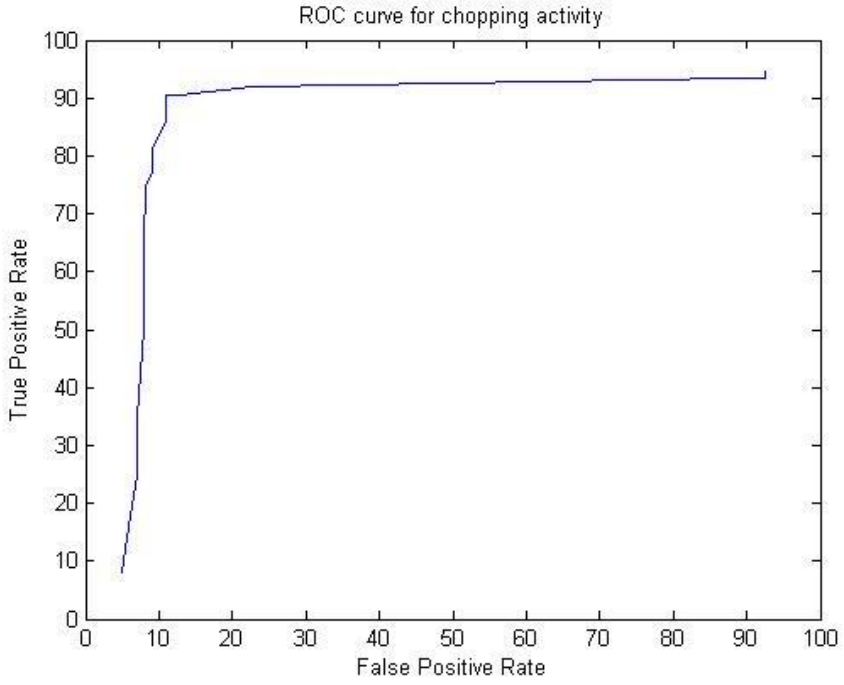

Fig 20. ROC Curve For The Activity: Chopping.

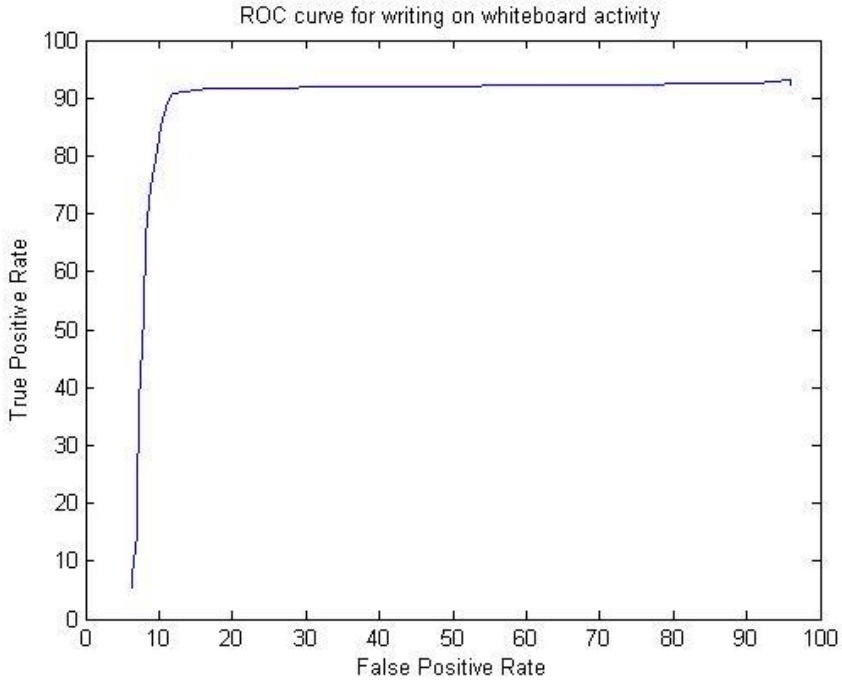

Fig 21. ROC Curve For The Activity: Writing On A Whiteboard.

The confusion matrix for the set activities that were obtained at our lab is presented below.

	SIT	STAND	WALK	JUMP
SIT	0.93	0.04	0.02	0.01
STAND	0.02	0.94	0.01	0.02
WALK	0.02	0.03	0.93	0.01
JUMP	0.01	0.04	0.04	0.92

Table IV. Confusion Matrix For Sit, Stand, Walk And Jump Activities.

It is clear from the table that we are able to achieve high recognition rates. This however is not sufficient to validate our approach, for this purpose we use the dataset from Cornell. Next we present the ROC curve for the chopping activity, the confusion matrix obtained from our method as well as the Maximum-entropy Markov model method described by Sung et al. [27].

	CHOPPING	STIRRING	DRINKING	TALKING OVER PHONE	WORKING ON LAPTOP	WRITING ON WHITEBOARD
CHOPPING	0.88	0.1	0	0	0.01	0
STIRRING	0.11	0.89	0	0	0	0
DRINKING	0	0	0.91	0.09	0	0
TALKING OVER PHONE	0	0	0.07	0.93	0	0
WORKING ON LAPTOP	0.04	0.03	0	0	0.93	0
WRITING ON WHITEBOARD	0	0	0.01	0.05	0	0.94

Table V. Confusion Matrix For Cornell CAD-60 Dataset Using RISq.

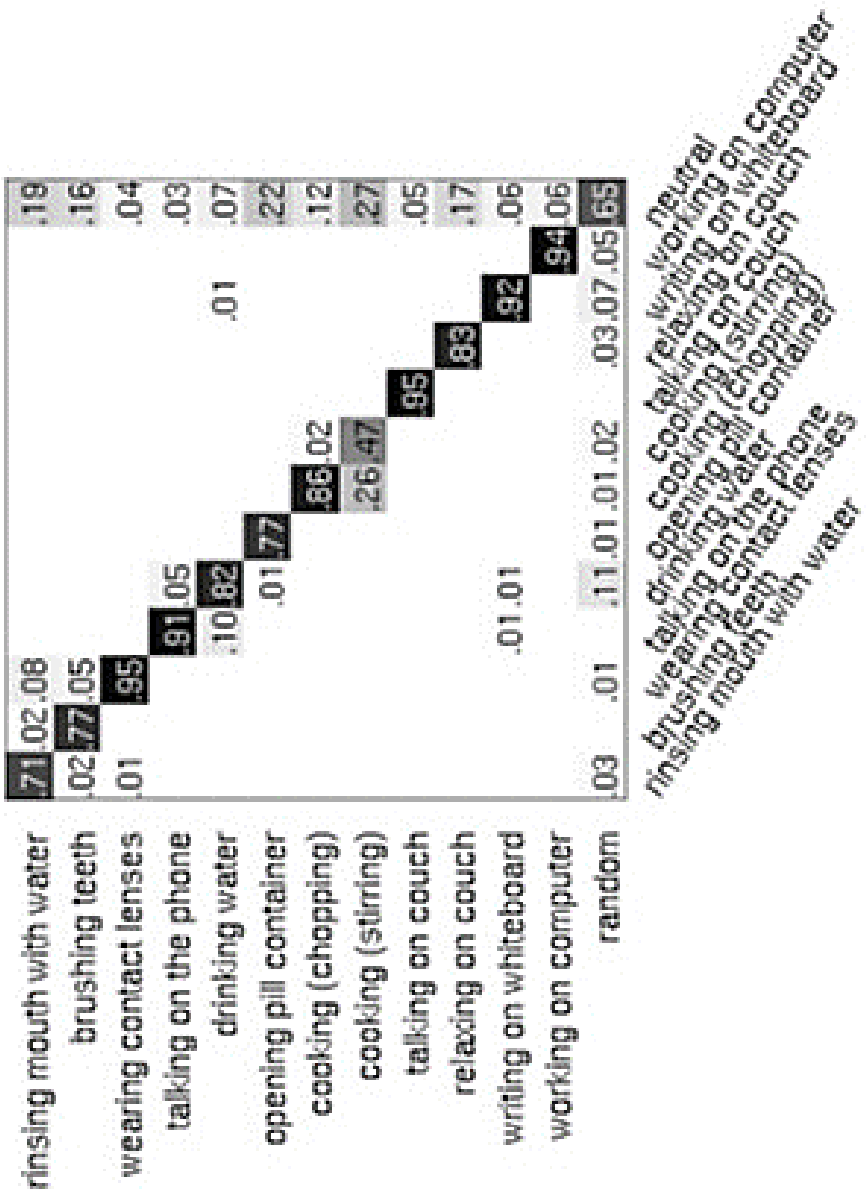

Fig 22. Confusion Matrix For All The Activities By Cornell University [1]

[1] Copyright, [27] Jaeyong Sung, Colin Ponce, Bart Selman and Ashutosh Saxena. Unstructured Human Activity detection from RGBD Images. In *ICRA*, 2012.

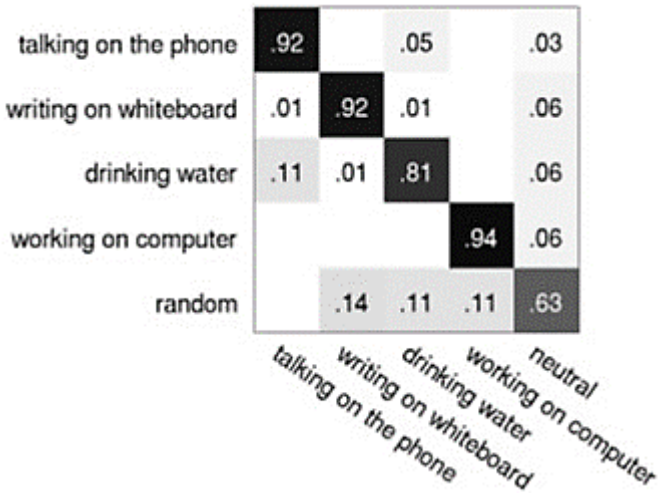

Fig 23. Confusion Matrix For Part of The Activities By Cornell University [2]

The above table and figures are the confusion matrices that are obtained from the same dataset using different methods: our approach, RISq and the other described by Sung et al. [27]. The matrices reveal that RISq performs better, and in some cases the results are very high. The tests conducted were in the '*have seen*' setting according to [27], this is the same as cross validation. In this setting out of the entire subject's data, half is given as testing and the other half is used for training. In a sense this is letting the model take a peek into the training once and then testing out the data. The results obtained for some cases like working on computer is similar, this can be attributed to the method used for data capture and its limitations. Comparing the results we can say that our approach using RISq performs better.

[2] Copyright, [27] Jaeyong Sung, Colin Ponce, Bart Selman and Ashutosh Saxena. Unstructured Human Activity detection from RGBD Images. In *ICRA*, 2012.

6.2 Conclusion:

Regardless of all the state-of-the-art algorithms, human activity recognition (HAR) is still an open problem mainly due to various constraints such as limitations of the acquisition systems, occlusions, large training models, etc. Early research used markers placed at various joints on the human to track movements, due to various constraints of the marker based motion capture, markless motion capture was developed. Many algorithms like Dynamic Time Warping (DTW), Hidden Markov Models (HMM), and Maximum Entropy Markov Models (MEMM) is used for the recognition framework. These algorithms have their own advantages and drawbacks.

HMM is a very good method for recognition of 1D sequences such as speech and human activity Rahman et all. [26], Moselund et all. [22], Rabiner et all [24][25]. Therefore HMM is very popular. HMM performs quite well if training exemplars in sufficient number are available. HMM is a generative probabilistic model which is designed to generate a specific class of activity sequences i.e. activity exemplars. The training of HMM requires a lot of training exemplars because the parameters of the hidden models are usually computed by Expectation Maximization (EM), which needs a sizeable amount of training data. Training of a complete HMM system is even longer because each activity class requires a separate model, which needs separate training. In the recognition phase all the models have to be examined for each input. This may slow the recognition process if the number of classes is large. Both in training and in recognition one has to determine the timing of the beginning and end of each activity exemplar. Inaccurate timing may result in substantial errors. In many applications including HAR getting accurate timing is quite difficult.

Recognition by Indexing and Sequencing (RISq) is a non-parametric approach used to classify temporal sequences, this technique has been successfully implemented for human activity recognition in the past by Ben-Arie et al. [2][3]. Although there are few similarities between DTW Rabiner et all. [24], Holt at all. [13] and RISq there are significant differences between the two. The two methods are superficially similar because both methods match vector sequences while allowing warping and both methods use Dynamic Programming (DP) to find the optimal matching score. But here the similarity ends. Firstly, DTW regards the absolute vector distances as an inverse similarity measure and tries to find the match between the sequences which has the minimal cumulative distance using DP, which is constrained by monotonic sequencing requirement. But the sequences must be matched according to a *connected and monotonic* path. On the other hand, RISq performs bipartite graph matching, which allows partial matching of both input and exemplar sequences in *any sequential configuration* i.e. the path could be disconnected. This provides much more flexibility in warping and improves RISq's recognition rates noticeably. Such a process is feasible because RISq is accumulating *vector similarities not distances.* RISq also penalize dissimilarities while DTW uses only distance penalties. The similarity-dissimilarity scoring in RISq provide a substantial improvement in performance. Similarity and distance have inverse relations. Mathematically, there is a non linear and inverse relation between the two. However, *minimizing accumulated distances is not the same as maximizing accumulated similarities.* In addition, DTW must determine the endpoints of both sequences (also HMM). RISq does not have to. This discrepancy alone could make a lot of difference in the recognition rates because DTW must apply pre-segmentation to locate these end points, a process which is very much error prone. Another substantial difference: with DTW one has to match all the exemplars serially one by one, in order to find the best match. This makes DTW much slower especially if it has many exemplars

(HMM has the same problem). In contrast, RISq is a *parallel* method that employs indexing and matches all the exemplars at once. To conclude, all these differences result in much better performance of RISq in recognition rates, false alarm rates and in computation time requirements.

RISq can efficiently achieve robust recognition even with few training exemplars from each class. The training of the RISq is simple and involves storing all the exemplars in an underlying data structure for efficient data retrieval. The voting and sequencing stages are the two key stages the make up the classification stage. The voting involves retrieval of the k nearest neighbors for each input and assigning a vote. In the sequencing stage, dynamic programming algorithm is used to find the optimal sequence of votes for each of the input sequence.

The work presented here is based on the markless motion capture, utilizing the Kinect manufactured by Microsoft and uses Infra-Red (IR) structured light to compute depth images with relatively high resolution in all 3 dimensions. Using RISq, a non-parametric approach used to classify temporal sequences, we successfully recognize an unknown activity sequence. Experiments were conducted on two datasets: one acquired at our Machine Vision Lab and the other is the CAD-60 dataset from Cornell University's Robot Learning Lab [27]. Results from these experiments show that high recognition rates is achieved when using RISq.

6.3 Recommendations for Future Work:

There can be a few improvements done to our approach to achieve even higher recognition rates, some of the changes are for the acquisition of the human skeleton and other is for the RISq algorithm. Our work uses one kinect camera for the acquisition of the human skeleton, while this is better than using a standard 2-D camera, it has flaws of its own: susceptible to occlusion, limited coverage area. In order to overcome these using two or multiple kinect cameras for the acquisition is suggested. This overcomes the problems of occlusion and also this can cover and entire room. The improvement that can be done to the RISq algorithm is to fasten its execution time, this can be improved by implementing the same algorithm keeping parallelism in mind, i.e. rewriting the algorithm in terms of parallel programming approach. This enables the program to execute a lot faster there by reducing the execution time. These improvements are can significantly increase the speed, with minor changes to the original framework.

REFERENCES

1. Aggarwal, J.K, and Cai, Q.: Human activity analysis: a review. IEEE Proceedings on Nonrigid and Articulated Motion Workshop, pages 90-102, 1997.

2. Ben-Arie J., Zhiqian Wang, Pandit, P, and Rajaram, S.: Human activity recognition using multidimensional indexing. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1091-1104, 2002.

3. Ben-Arie J., Pandit, P, and Rajaram, S.: View-based human activity recognition by indexing and sequencing. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages II-78 – II-83 vol.2, 2001.

4. Bengalur, M.D.: Human activity recognition using body pose features and support vector machine. In Proceedings of the International Conference on Advances in Computing, Communications and Informatics, pages 1970-1975, 2013.

5. Besl, P.J. and Mckay, Neil D.: A method for registration of 3-D shapes. In IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 239-256, 1992.

6. Blackburn, J. and Ribeiro, E.: Human motion – Understanding, Modeling, Capture and Animation, pages 285-298, 2007.

7. Chen H., and Bahnu, B.: Efficient recognition of highly similar 3d objects in range images. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 172-179, 2009.

8. De Berg M., van Kreveld, M., Overmars, M., and Schwarzkopf, O.: Computational Geometry: Algorithms and Applications. Springer, 2000.

9. Feng Niu and Abdel-Mottaleb, M.: View-invariant human activity recognition based on shape and motion features. In Proceedings of the IEEE sixth International Symposium on Multimedia Software Engineering, pages 546-556, 2004.

10. Franzini S., and Ben-Arie, J.: Speech recognition by indexing and sequencing. International Conference on Soft Computing and Pattern Recognition, pages 93-98, 2010.

11. F. S. Grassia, Practical Parameterization of Rotations using the Exponential Map, Journal of Graphics Tools, vol. 3, no. 3, 1998.

12. Han J., Ling Shao, Dong Xu, and Shotton, J.: Enhanced computer vision with Microsoft kinect sensor: A review. IEEE Transactions on Cybemetics, pages 1318-1334, 2013.

13. Holt ten G.A, M. J. T. Reinders, and E. A. Hendriks. Multi-Dimensional Dynamic Time Warping for Gesture Recognition. In the Annual Conference of the Advanced School for Computing and Imaging, 2007.

REFERENCES (Continued)

14. Jia W., Won-Jae Yi, Saniie J., Oruklu, E.: 3D image reconstruction and human body tracking using stereo vision and kinect technology. IEEE International Conference on Electro/Information Technology, pages 1-4, 2012.

15. Khan, A.M., Young-Koo Lee, S.Y., and Tae-Seong Kim.: A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. IEEE Transactions on Information Technology in Biomedicine, pages 1166-1172, 2010.

16. Lester J., Tanzeem Choudhury , Nicky Kern , Gaetano Borriello , Blake Hannaford: A hybrid discriminative/generative approach for modeling human activities. In the Proceedings of the International Joint Conference on Artificial Intelligence, pages 766-772, 2005.

17. Li, Thi-Lan, Nguyen, Minh-Quoc, Nguyen, and Thi-Thanh-Mai: Human posture recognition using human skeleton provided by Kinect. International Conference on Computing, Management and Telecommunications, pages 340-345, 2013.

18. Li Q., Panlong Yang: Keep up with me: A gesture guided moving robot with Microsoft kinect. IEEE International Conference on Mobile Ad-Hoc and Sensor Systems, pages 435-436, 2013.

19. Li W., Zhengyou Zhang, Zicheng Liu: Action recognition based on a bag of 3d points. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop, pages 9-14, 2010.

20. Lim Yong-Wan, Hyuk-Zae Lee, Na-Eun Yang, and Rae-Hong Park: 3-D reconstruction using the kinect sensor and its application to a visualization system. IEEE International Conference on Systems, Man, and Cybemetics, pages 3361-3366, 2012.

21. Ma Kai, Ben-Arie, J: Vector array based multi-view face detection with compound exemplars. IEEE Conference on Computer Vision and Pattern Recognition, pages 3186-3193, 2012.

22. Moeslund Thomas B., Erik Granum: A Survey of Computer Vision-Based Human Motion Capture, 2001**.**

23. Nair, N.U., Sreenivas, T.V: Multi pattern dynamic time warping for automatic speech recognition. IEEE Region Conference TENCON, pages 1-6, 2008.

24. Rabiner, L: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, pages 257-286, 1989.

25. Rabiner, L., and Schmidt, C: Application of dynamic time warping to connected digit recognition. IEEE Transactions on Acoustics, Speech and Signal Processing, pages 377-388, 1980.

REFERENCES (Continued)

26. Rahman, A. Ahad, Tan, J.K, Kim, H.S, and Ishikawa, S: Human Activity Recognition: Various Paradigms. International Conference on Control, Automation and Systems, pages 1896-1901, 2008.

27. Sung J., Ponce, C., Selman, B, and Saxena, A.: Unstructured human activity detection from RGBD images. IEEE International Conference on Robotics and Automation, pages 842-849, 2012.

28. Uddin, M.Z, Lee, J.J., and Kim, T.S: Independent component feature-based human activity recognition via linear discriminant analysis and hidden markov model. International Conference of the IEEE Engineering in Medicine and Biology Society, pages 5168-5171, 2008.

29. Uddin, M.Z, Nguyen Duc Thang, Tae-Seong Kim: Human activity recognition via 3-d joint angle features and hidden markov models. IEEE International Conference on Image Processing, pages 713-716, 2010.

30. Walker. W, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel, Sphinx-4: A Flexible Open Source Framework for Speech Recognition. Sun Microsystems, Technical report TR-2004-139, 2004.

31. Yang Z., Liu Zicheng, and Cheng Hong: RGB-Depth feature for 3d human activity recognition. Communications, China, vol 10, pages 93-103, 2013.

32. Yun K., Honorio, J., Chattopadhyay, D, and Berg, T.L: Two-person interaction detection using body-pose features and multiple instance learning. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop, pages 28-35, 2012.

33. Zhang H., and Yoshie, O.: Improving human activity recognition using subspace clustering. International Conference on Machine Learning and Cybermetics, pages 1058-1063, 2012.

34. Zhang Mi, Sawchuk, A.A: Human daily activity recognition with sparse representation using wearable sensors. IEEE Journal of Biomedical and Health Informatics, pages 553-560, 2013.

VITA

NAME:	Sadgun Srinivas Devanahalli Shashikumar
EDUCATION:	Bachelor of Engineering in Electronics and Communication Engineering, Visvesvaraya Technological University, India, 2008. Master of Science in Electrical and Computer Engineering, University of Illinois at Chicago (UIC), Chicago, Illinois, 2015.
EXPERIENCE:	Machine Vision Laboratory, 10/2012 to 11/2014. Research involved developing a human activity recognition system using the depth sensor, kinect.
PROFESSIONAL MEMBERSHIP:	Graduate student member, Institute of Electrical and Electronics Engineers (IEEE)