
Masters Theses

Student Theses and Dissertations

2012

Development of Kinect^{TR} applications for assembly simulation and ergonomic analysis

Chinmay Prakash Daphalapurkar

Follow this and additional works at: https://scholarsmine.mst.edu/masters_theses



Part of the [Mechanical Engineering Commons](#)

Department:

Recommended Citation

Daphalapurkar, Chinmay Prakash, "Development of Kinect^{TR} applications for assembly simulation and ergonomic analysis" (2012). *Masters Theses*. 7287.

https://scholarsmine.mst.edu/masters_theses/7287

This thesis is brought to you by Scholars' Mine, a service of the Missouri S&T Library and Learning Resources. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

DEVELOPMENT OF KINECT^{TR} APPLICATIONS FOR ASSEMBLY SIMULATION
AND ERGONOMIC ANALYSIS

by

CHINMAY PRAKASH DAPHALAPURKAR

A THESIS

Presented to the Faculty of the Graduate School of the
MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

2012

Approved by

M. C. Leu, Advisor
F. W. Liou
X. F. Liu

© 2012

Chinmay Prakash Daphalapurkar

All Rights Reserved

ABSTRACT

Marker-less motion capture technology has been harnessed for several years to track human movements for developing various applications. Recently, with the launch of Microsoft Kinect, researchers have been keenly interested in developing applications using this device. Since Kinect is very inexpensive (only \$110 at the time of writing this thesis), it is a low-cost and a promising substitute for the comparatively expensive marker-based motion capture systems. Though it is principally designed for home entertainment, numerous applications can be developed with the capabilities of Kinect. The skeleton data of a human being tracked by a single Kinect device is enough to simulate the human movements, in some cases. However, it is highly desirable to develop a multiple Kinect system to enhance the tracking volume and to address an issue of occlusions. This thesis presents a novel approach for addressing the issue of interference of infrared light patterns while using multiple Kinect devices for human motion capture without lowering the frame rate. This research also presents a software solution to obtain skeleton data from multiple Kinect devices using Kinect for Windows SDK. It also discusses the development of an application involving auto scaling of a human model in digital human modeling software by Siemens Jack and human motion simulation using skeleton tracking data from Kinect to assist the industries with a flexible tool for ergonomic analysis. Further, the capability of this application for obtaining assembly simulations of fastening operations on an aircraft fuselage is also presented.

ACKNOWLEDGMENTS

I would like to extend my deepest gratitude and thanks to my advisor Dr. Ming Leu for his never ending support, and excellent guidance throughout the duration of this research study.

I would like to thank Dr. Wenjuan Zhu for her constant help during the course of my studies as well as research. I would also like to express my thanks to Dr. X. F. Liu and Dr. F. W. Liou for spending their time and effort as my committee members and helping me through the course of studies at Missouri S & T.

I wish to sincerely thank all my research mates in the Virtual Reality and Rapid Prototyping Lab, who have helped me greatly during my research. I would specially thank Dr. Akul Joshi and Dr. Cemil Oz for their help and insights during the project work. I would like to express my thanks to Sajeev Chirayil Puthaveetil, Diego Garcia, Raghvendra Kuber, Zafar Khan and Swanand Gadgil, both, for assisting me in the project and for being the subjects in my experiments. I would also like to express my gratitude towards Rohit Bapat, Maxwell Mulholland, Aaron Thornton, Brad Deuser, Bharat Mahajan, and the IT team in helping me with the experimentation.

I wish to extend my thanks to the Center for Aerospace Manufacturing Technology at Missouri S & T for funding and supporting my research work through the grants from Spirit Aerosystems.

Finally, I am deeply indebted to my parents for their eternal support and all my friends who stood by me during the entire duration of my studies at Missouri S & T.

TABLE OF CONTENTS

	Page
ABSTRACT.....	iii
ACKNOWLEDGMENTS	iv
LIST OF ILLUSTRATIONS.....	vii
LIST OF TABLES.....	x
SECTION	
1. INTRODUCTION.....	1
1.1. MOTIVATION AND OBJECTIVE OF RESEARCH.....	1
1.2. LITERATURE SURVEY.....	3
1.3. THESIS OVERVIEW.....	5
2. MARKERLESS MOTION CAPTURE SYSTEM BASED ON KINECT	7
2.1. OVERVIEW OF MARKER-LESS MOTION CAPTURE TECHNIQUES	7
2.1.1. Classification.....	7
2.1.1.1. Time of flight (TOF).....	8
2.1.1.2. Structured light.....	9
2.2. MICROSOFT KINECT	10
2.2.1. Kinect Sensor.....	10
2.2.2. Principle of Depth Sensing in Kinect.....	11
2.2.3. System Architecture.....	17
3. DEVELOPMENT OF A MULTIPLE KINECT MOTION CAPTURE SYSTEM. 20	
3.1. MOTIVATION FOR DEVELOPING A MULTIPLE KINECT SYSTEM.....	20
3.1.1. Determination of Kinect Sensor's Tracking Range	20
3.1.2. Interference of Multiple Infrared Beams.....	29
3.2. LITERATURE REVIEW ON USE OF MULTIPLE KINECTS	31
3.3. MULTIPLE KINECT SYSTEM	33
3.3.1. Principle.....	33
3.3.2. Software and Hardware Architecture.....	33
3.3.3. Shutter Mechanism.....	34
3.3.3.1. Shutter and motor mount.....	34

3.3.3.2. Servo motor.....	36
3.3.4. NI myDAQ.....	37
3.3.5. Device Switching Algorithm.....	38
3.4. CAMERA CALIBRATION	43
3.4.1. Calibration of Two Kinect Devices.....	44
3.4.2. Calibration of Three Kinect Devices.....	46
3.4.3. Calibration Results for Two-Kinect System.	47
4. APPLICATIONS.....	54
4.1. DIGITAL HUMAN AUTO-SCALING APPLICATION FOR PARTIAL BODY MOTION CAPTURE.....	54
4.1.1. Digital Human in Jack.....	54
4.1.2. Auto-Scaling the Digital Human Model Using Kinect Data.....	55
4.1.3. Simulation Support.....	61
4.2. HUMAN MOTION SIMULATION USING KINECT.....	68
4.2.1. Interfacing Kinect and Jack.....	68
4.2.2. Mapping Jack and Kinect Coordinates Systems.	72
4.2.3. Simulation Using Single Kinect.	74
4.2.3.1. Evaluation of Kinect skeleton tracking data for dynamic motions.....	80
4.2.3.2. Accuracy analysis of elbow angles for static postures.....	86
4.3. SIMULATION RESULTS FOR MULTIPLE KINECTS	90
4.3.1. Simulation Results for Two-Kinect System.....	90
4.3.2. Comparison of Simulation from Single and Two-Kinect Systems.	91
4.3.3. Simulation Using Three-Kinect System.....	100
4.4. ERGONOMIC ANALYSIS.....	103
5. CONCLUSION	106
BIBLIOGRAPHY.....	108
VITA	111

LIST OF ILLUSTRATIONS

	Page
Figure 2.1. Time of flight principle	8
Figure 2.2. Structured light patterns	9
Figure 2.3. Kinect sensor	11
Figure 2.4. Schematic representation of PrimeSense depth sensing technology	12
Figure 2.5. Illumination assembly and speckled pattern.....	13
Figure 2.6. Process of 3D reconstruction from depth	14
Figure 2.7. Principle of PrimeSense depth sensing technology.....	15
Figure 2.8. Pair of depth and hand labeled images	17
Figure 2.9. System architecture: Kinect for Windows SDK.....	18
Figure 2.10. System architecture: OpenNI	19
Figure 3.1. Posture for accuracy analysis	22
Figure 3.2. Kinect skeleton showing 20 body joints.....	23
Figure 3.3. Kinect Explorer application's video stream snapshot showing the skeleton superimposed on the video stream.....	24
Figure 3.4. Percentage error in the Torso segment lengths at different subject orientations for different subject postures.....	25
Figure 3.5. Percentage Errors in the Left Upper Arm segment length	26
Figure 3.6. Mean of relative errors of segment lengths over different subject postures at different subject orientations.....	28
Figure 3.7. Degradation of the depth image quality with different number of Kinects..... facing the same surface	29
Figure 3.8. Mannequin.....	30
Figure 3.9. System architecture for two-Kinect system.....	34
Figure 3.10. Shutter design	35
Figure 3.11. Shutter positions	35
Figure 3.12. Principle of servomechanism	36
Figure 3.13. Physical I/O channels of NI myDAQ.....	37
Figure 3.14. Device switching algorithm.....	39
Figure 3.15. Loss of frame rate due to delay in switching.....	40

Figure 3.16. Improved frame rate during switching due to inter-process communication.....	40
Figure 3.17. Operator angle calculation.....	41
Figure 3.18. Selection of angle between two Kinects.....	42
Figure 3.19. IR images.....	44
Figure 3.20. Calibration of two Kinect devices	45
Figure 3.21. Calibration of three Kinect devices	46
Figure 3.22. Experimental set-up.....	49
Figure 3.23. X coordinates of the right hand	51
Figure 3.24. Y coordinates of the right hand	52
Figure 3.25. Z coordinates of the right hand.....	53
Figure 4.1. Jack skeleton.....	55
Figure 4.2. System architecture for digital human auto-scaling	57
Figure 4.3. Kinect skeleton	57
Figure 4.4. Segment definitions	58
Figure 4.5. System architecture for human motion simulation.....	62
Figure 4.6. Hierarchical representation of Jack digital human model	63
Figure 4.7. Constraints.....	65
Figure 4.8. Interface in Jack.....	67
Figure 4.9. Structure of positions and orientations of marker triads and Jack skeleton before constraining.....	68
Figure 4.10. Kinect-Jack skeleton tracking interface.....	70
Figure 4.11. Mapping of real and virtual worlds	73
Figure 4.12. Definition of reference points.....	73
Figure 4.13. Evaluation of mapping, Left: real world poses, Right: virtual world poses in Jack	74
Figure 4.14. Simulation of fastening operation using a Kinect sensor	75
Figure 4.15. Filtering of skeletal tracking data	79
Figure 4.16. Dynamic motion analysis for first set of motions	81
Figure 4.17. Dynamic motion analysis for a sample from first set of motions.....	83
Figure 4.18. Dynamic motion analysis for second and third set of motions	85
Figure 4.19. Mannequin postures for angle accuracy analysis	86

Figure 4.20. Angle measurement set-up	86
Figure 4.21. Set-up for two-Kinect system.....	90
Figure 4.22. Motion simulation using a tracking system with two Kinects	91
Figure 4.23. Comparison of simulation from one-Kinect and two-Kinect systems	93
Figure 4.24. Sequence of fastening operation captured using a three-Kinect system ...	101
Figure 4.25. Simulation of fastening operation in Jack using data captured from three Kinect system, corresponding to the sequence in Figure 4.24.....	102
Figure 4.26. RULA	105
Figure 4.27. RULA: analysis summary	105

LIST OF TABLES

	Page
Table 3.1. Segments formed using Kinect skeleton data	23
Table 3.2. Torso segment accuracy analysis (250 frames)	31
Table 3.3. Mean error in the 3D joints' positions (Unit: cm)	48
Table 4.1. End points of body segments for Jack and Kinect skeletons	59
Table 4.2. Right lower arm segment measurements (Unit: cm)	61
Table 4.3. Kinect and Jack skeleton joint comparison.....	69
Table 4.4. Measured vs. actual, right elbow joint angle at different postures	88
Table 4.5. Measured vs. actual, left elbow joint angle at different postures	89
Table 4.6. Action levels in RULA	104

1. INTRODUCTION

1.1. MOTIVATION AND OBJECTIVE OF RESEARCH

In aircraft assembly, automation of assembly operations is a common approach to reduce the process time and reduce the probability of human errors. However, manufacturers tend to use automation for fabrication, and leave assembly to human operators who can adapt to changing circumstances more rapidly and easily than machines. While performing these assembly operations the operator has to perform a number of challenging and physically exhausting operations which may cause musculoskeletal injuries to the operator. Some of the potential sources of ergonomic injuries to the operator are: working in confined spaces such as aircraft fuel cells and wings, forceful exertions such as lifting and carrying, vibration during operations such as riveting, repetitive motions and human/machine interface such as tooling and awkward posture assembly, etc. [1]. Hence there is a need to inspect and analyze such tasks in order to ensure physical safety of the operator. Ergonomic analysis is the solution to the above problems which may require expertise and consultation programs. Generally, engineers use ergonomic analysis software which requires building of animations using key frame methods which might be very time-consuming and depends on the skills of the person using the software. This can be resolved by using motion capture technology to obtain these simulations in the ergonomic analysis software to assist the engineers to quickly assess the risks involved in the assembly operations. This research was motivated

by a collaborative project with Spirit AeroSystems in relation to the above requirements. A task was identified for the purpose of developing a tool for analyzing the ergonomic risk, to which a mechanic is subjected to, while performing repetitive motions during fastening operations in different awkward postures during the installation of a belly skin of an aircraft fuselage.

Optical motion tracking technology is widely used to track the human movements for developing applications in the field of surveillance, control and motion analysis [2]. The optical motion tracking systems can be categorized into marker-based systems and marker-less systems. A number of passive and active, marker-based, human motion capture systems are commercially available, e.g. those provided by Natural Point [3], Vicon [4], Motion Analysis [5] or Phase Space [6]. Although these systems are sufficiently accurate to obtain human motion data for the purpose of digital human modeling and simulation, they need the operators to wear a body suit mounted with markers. If the operator were to wear such a suit during assembly operations on the shop floor, it will cause discomfort to the operator and may cause interference in their operation, which is not acceptable. Moreover, these systems are prone to marker occlusion issues which might affect the captured motion data. Marker-less optical motion capture technologies would be a promising and inexpensive alternative to overcome some of the limitations of marker-based systems. This research concentrates on developing an application using Microsoft Kinect sensor which is based on a depth sensing technology developed by Primesense, which may prove to be an effective and a quick tool to assist the engineers in assessing the risk of ergonomic injuries to an assembly operator in manufacturing industry. The thesis concentrates on developing a multiple Kinect system

to resolve the issue of insufficient tracking volume tracked while using only a single Kinect. This thesis will also introduces a novel approach to resolve the issue of interference of multiple structured light patterns when using multiple Kinect sensors for developing an application to track an operator's movements while performing a fastening operation on an aircraft fuselage mockup and develop simulations using the captured data for performing ergonomic analysis in a digital human modeling software called Jack (Siemens Corp).

1.2. LITERATURE SURVEY

Traditional key frame animation is time-consuming and requires the development of predictive software tools. Motion capture data has a major advantage over traditional motion prediction for animation [7]. The animation can be obtained with the help of Digital Human Modeling (DHM) software, an overview of which was given in [8]. There are a number of ready-to-use marker-based human motion capture systems which have been used in the field of biomechanics. The motion capture system by Motion Analysis Corp. has a plug-in with DHMs such as Siemens' Jack and Dassault Systems' CATIA V5 to facilitate a simulation environment for ergonomic analysis. Similarly, Vicon also has a plug-in with Jack. Jie et al. [9] provided an overview of developing an interface between a wireless optical motion capture system called ShapeWarpIII from Measurand Corp. and DELMIA Human software package. Zhu et al. [10] described how a motion capture technology developed using Nintendo Wii Remotes was used as an inexpensive tool for the purpose of capturing the operator movements while performing fastening operation. The motion capture data was further used to develop simulations in Jack, which can be

used to perform ergonomic analysis. However, a Wii Remote cannot track more than 4 IR hotspots and hence this system can track a very limited number of human body segments. Dua et al. [11] presented a methodology developed to integrate Motion Analysis Corp's motion capture software with Jack to conduct ergonomic analysis of an operator performing a lifting job on the factory floor. Although it is evident that the marker-based technology is a boon to the field of biomechanics, it has numerous limitations and disadvantages such as marker occlusion, discomfort caused on the subject due to the wearable suit with markers, which make it difficult to implement on a shop floor. Moreover, with the rapidly decreasing cost of computer vision based systems, marker-less technology is an attractive alternative to overcome these issues.

Marker-less optical motion capture technology has been a topic of research for the last two decades. Stereo vision, time of flight, and structured light are some of the marker-less technologies used for optical 3D mapping to reconstruct 3D objects. The marker-less motion tracking capability of Microsoft Kinect™ [12], which is based on the structured light technique, has caught the attention of numerous software professionals and hobbyists who seek to develop applications other than gaming. Lindstrom et al. [13] described how, with the combination of video, audio and depth capabilities of a Kinect sensor can be used to develop different applications in the fields of entertainment, medical, fashion, education, surveillance, sports and biomechanics. In this research, we concentrate on developing applications in the animation and biomechanics fields. There are some studies carried out to do ergonomic analysis with the help of a single Kinect sensor. Ray et al. [14] presented a framework for NIOSH suggested, rule-based ergonomics evaluation of a construction worker performing overhead lifting tasks in a

training environment using the skeleton tracking capability of Kinect. Martin et al. [15] presented a real-time ergonomic analysis system using a single Kinect to analyze lifting jobs performed by human operators, recommended to be used in a training scenario. The use of Kinect for fastening operation in an aircraft manufacturing industry has not been reported in the literature. The objective of this research is to evaluate the different ways for using Kinect sensors for ergonomic analysis of fastening operation for assessing the risks involved in performing different assembly operations in an aircraft assembly line. Moreover, multiple Kinect devices have not been used for this purpose, thus development of multiple motion capture systems for human motion tracking is a major contribution of this research. The literature review on the multiple Kinect devices will be described in Section 3.2.

1.3. THESIS OVERVIEW

The goal of this thesis was to discuss the use of Microsoft Kinect to develop a marker-less motion capture system to track human motions of an operator performing a fastening operation on a belly section of an aircraft fuselage. This was achieved by developing an application to use the skeleton tracking data provided by the Kinect API's (Application Programming Interface) to generate digital human simulations for ergonomic assessment using the virtual human prototyping system Siemens Jack. These simulations are a direct input for the Task Analysis Toolkit (TAT) available with Jack for assessing the risks involved in the fastening operation. This thesis attempts to provide a tool for engineers to evaluate work-related injuries, particularly while performing fastening operations.

Section 2 provides an overview of existing depth sensing technologies used for human motion capture. It also provides an overview of the depth sensing technology developed by PrimeSense, which is the basis of the Kinect sensor. Section 3 describes the development of a motion capture systems using multiple Kinect devices. This work proposes a novel approach to address the issue of interference between multiple infrared patterns, which is caused due to the simultaneous use of multiple Kinect devices which affects the quality of the depth image and hence, the skeleton data. Finally, Section 4 discusses the applications developed using the skeleton tracking capability of the Kinect for ergonomic analysis.

2. MARKERLESS MOTION CAPTURE SYSTEM BASED ON KINECT

2.1. OVERVIEW OF MARKER-LESS MOTION CAPTURE TECHNIQUES

This section briefly discusses the different optical and marker-less technologies currently used for human motion capture and introduces Microsoft Kinect which was used in this research.

2.1.1. Classification. Classification of optical, marker-less motion capture approaches depends on various factors and, numerous surveys categorize them into dissimilar classes. One such survey is [16], which is structured on the functional taxonomy of the approaches such as Initialization, Tracking, Pose Estimation and Recognition of human figures. Another approach [17] described the taxonomy based on three levels, viz. human detection (low-level), human tracking (intermediate-level), and human behavior understanding (high level). Mündermann et al. [2] discussed the need for marker-less motion capture systems in view of biomechanical applications and divided vision based motion capture systems into two groups, namely, active and passive. While passive systems are those which make use of images, active systems are those which make use of light (mostly in the infrared spectrum) information to generate human motion data. Additionally, as discussed in [16] the tracking stage is further divided into two stages: (1) figure-foreground segmentation and (2) temporal correspondences. Figure-foreground segmentation is the process of separating the humans from the rest of the scene. Segmentation is further categorized based on motion, appearance, shape or depth data. In this research we concentrate on the systems which use depth data for the segmentation process. Depth sensing technology has been a topic of research for the past

decade. This technology has three types: (1) Microwaves, (2) Light waves and (3) Ultrasonic waves. In the light wave domain, though traditional stereo vision cameras can be used for depth sensing, such as the system described by Thang et al. [18], where Bumblebee, a stereo camera by Point Grey Research, was used to fit an articulated human model to the 3-D data (upper body only), they are computationally expensive. The depth sensing using light waves which have proven to be effective for full body human motion capture can be classified mainly into two categories: time of flight (TOF) and structured light.

2.1.1.1. Time of flight (TOF). Knowing the speed of light, the principle of TOF cameras for depth sensing is to measure the absolute time required by a source light pulse to travel to an object in the scene and back to the detector. The phase shift between a continuously modulated sine wave from the emitter and the detector is shown in Figure 2.1. This phase shift is proportional to the distance from the sensor. Ganpathi et al. [19] gave an overview of different papers where TOF cameras were used for human motion tracking and described in detail how to use Swissranger SR4000 Time-of-Flight camera for real-time tracking of humans.

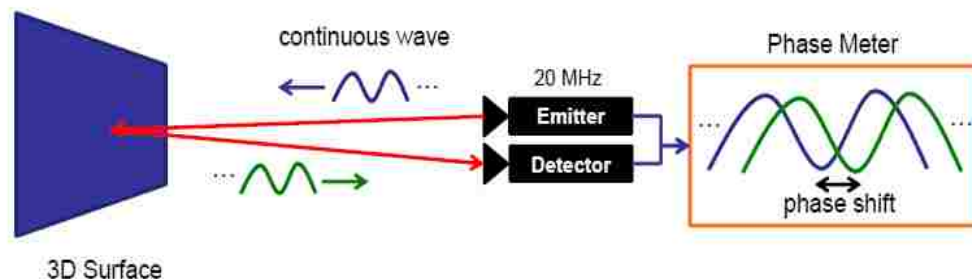


Figure 2.1. Time of flight principle [37]

The advantages of TOF cameras for human motion tracking are that these cameras possess high frame rates up to 90 fps (PMD [vision] ® CamBoard Nano reference design) making them suitable for real-time applications. Also, they can obtain range and amplitude images simultaneously. However, TOF cameras have to assume that the person being tracked is facing the cameras. Moreover, interference issues are caused due to use of multiple TOF cameras and calibration errors are caused due to low resolution of TOF cameras. Another disadvantage is that the number of subjects tracked by a single camera is limited to one.

2.1.1.2. Structured light. Fofi et al. [20] categorized invisible structured light technology (suitable for human motion capture) depending upon the type of light into three types: (1) Infrared Structured Light (IRSL), (2) Imperceptible Structured Light (ISL) and (3) Filtered Structured Light (FSL). In the IRSL depth sensing systems the projector emits a near-infrared light which is scattered by a pattern generator onto the scene which is then detected by either CCD (Charge-Coupled Device) or CMOS (Complementary Metal–Oxide–Semiconductor) sensors. Figure 2.2 shows different types of structured light patterns.

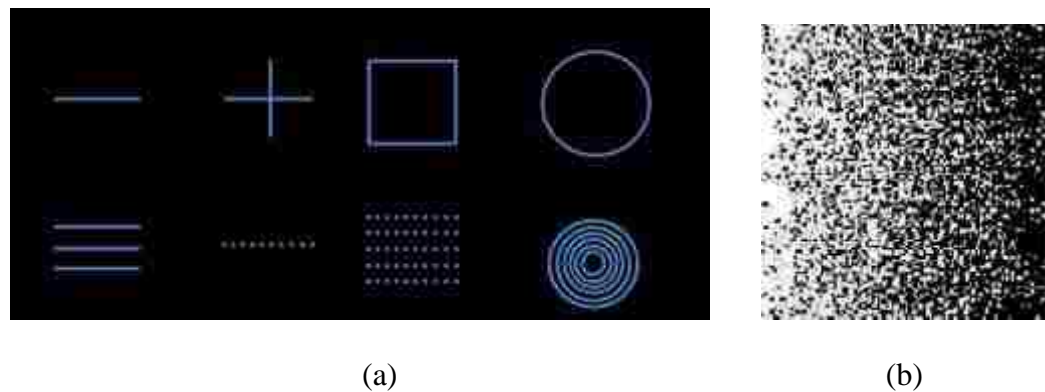


Figure 2.2. Structured light patterns: (a) Simple structured light patterns, (b) Coded structured light pattern [22]

It is important to note though, that the real-time motion capture with traditional IR pattern based structured light is a difficult task. Hence, different patterns are used to reduce the processing time of the structured light depth sensing system and to improve the resolution and accuracy of the data processed. One such pattern is shown in Figure 2.2 (b). The following section will discuss such a pattern which is the defining technology of the Prime Sensor.

2.2. MICROSOFT KINECT

Microsoft Kinect and Asus Xtion Pro Live [21] are the two depth sensing devices currently commercialized for human motion capture. Although they are based on the same technology developed by PrimeSense, they have certain differences which should be taken into consideration while selecting a device for motion capture. In the current research Microsoft Kinect is used, since its popularity has given birth to vast forum space which makes its use easy. In this section we will discuss the Kinect sensor and its skeleton tracking capabilities.

2.2.1. Kinect Sensor. Microsoft Kinect is a gaming device based on a depth sensor developed by PrimeSense [22]. The Kinect sensor box as shown in Figure 2.3, comprises of an infrared (IR) emitter and a depth sensor (CMOS) associated with the depth sensing arrangement suggested by PrimeSense. Microsoft adds an RGB camera as a color sensor (CMOS) for video support, and microphone arrays (not shown in the figure) for audio support. Kinect device is also provided with a tilt motor (not shown in the figure) to adjust the angle ($\pm 27^\circ$) of the Kinect box about the horizontal axis with respect to the base. The emitter emits infrared light beams which, when reflected back

from an object in the scene, are detected by the depth sensor, which converts the light beams into a depth image where each pixel carries the information about the distance between an object and the depth sensor. The next section describes this principle of depth sensing in detail.



Figure 2.3. Kinect sensor

Primesense's PS1080 SoC (System-on-Chip) is the brains behind the Kinect device. Its functions include controlling the projection of IR pattern on the scene, controlling data acquisition from a depth CMOS sensor which detects the IR light pattern from the scene, image processing to develop the depth image of the scene, map every pixel in the color image to a pixel in the depth image and transfer depth, color and audio data to the host controller via a USB2.0 interface.

2.2.2. Principle of Depth Sensing in Kinect. PrimeSense [23, 24, 25] described their technology and a system for 3D reconstruction of objects using projected speckled patterns. Figure 2.4 shows a schematic diagram of the basic principle of the PrimeSense depth sensing technology.

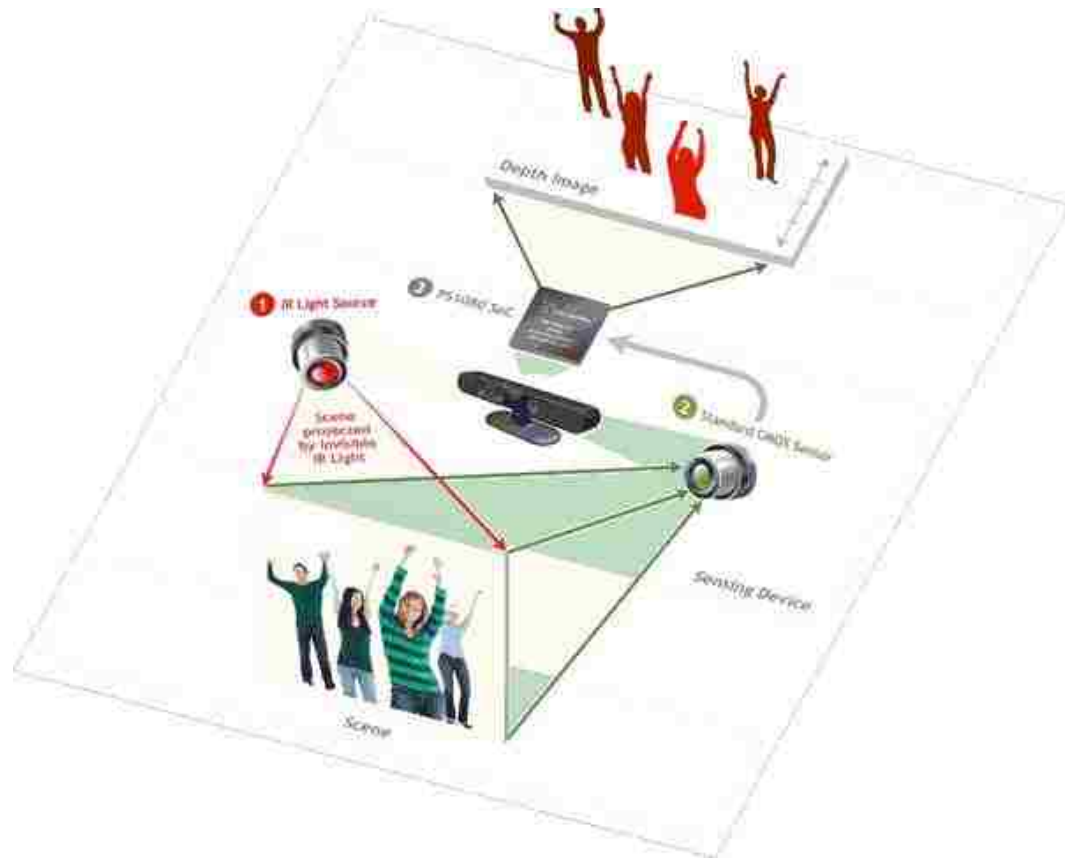


Figure 2.4. Schematic representation of PrimeSense depth sensing technology [22]

The system comprises of an Illumination Assembly and an Image Capture Assembly. A basic type of illumination assembly is shown in Figure 2.5 (a). A light source (infrared, ultraviolet or visible) trans-illuminates a transparency and the light is projected onto the scene with suitable optics. The transparency contains a pattern on which an incident beam gets diffracted to project an uncorrelated speckled pattern on the scene, an example of which is shown in Figure 2.5(b). Uncorrelated patterns are the projected patterns of spots whose positions are uncorrelated in the planes transverse to the projection beam axis (If the beam is assumed to be in the Z-axis then the uncorrelated

pattern are in the X-Y plane). The motive to do this is to simplify the image mapping algorithm. The image capture assembly consists of an objective optic which focuses the image onto an image sensor (CCD or a CMOS).

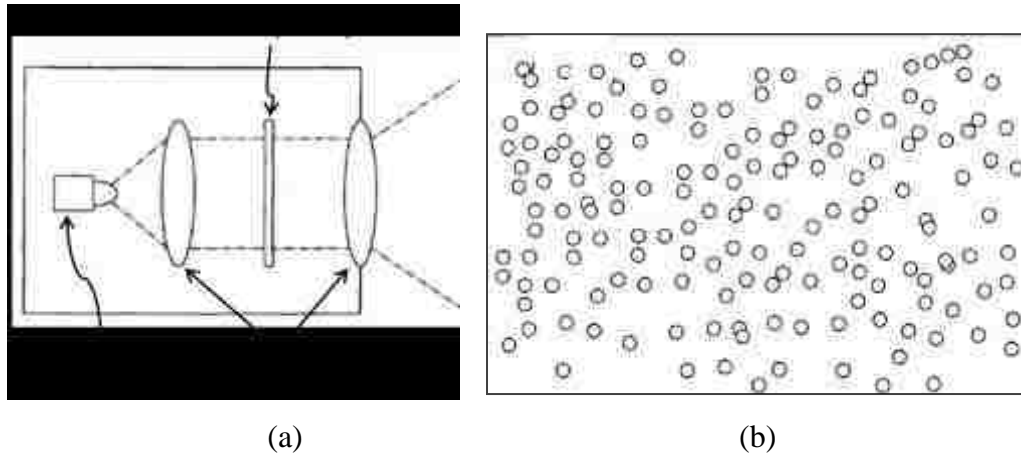


Figure 2.5. Illumination assembly and speckled pattern: (a) Illumination assembly, (b) Speckled pattern [23]

Figure 2.6 shows the process followed by the system to achieve real-time 3D reconstruction of an object scene. Initially a reference image is obtained by projecting the uncorrelated pattern on the reference plane at a known distance. This image is stored in the processor for mapping with future images.

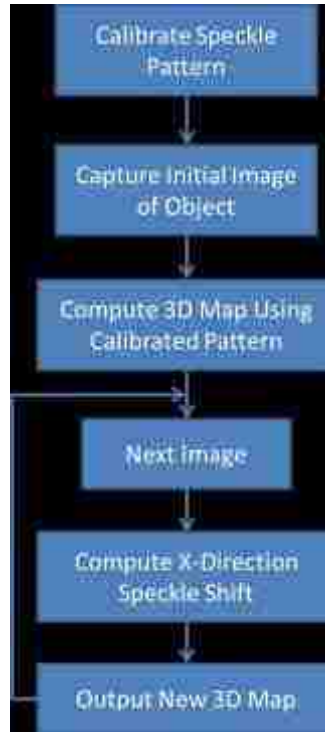


Figure 2.6. Process of 3D reconstruction from depth [23]

For tracking, the constant speckled pattern is projected into the object scene and the images obtained are mapped to the reference image. The mapping is based on a correlation between a pixel window of the object image and the reference image [24]. A small window of say 16X16 pixels on the object image is taken around an inspection point $P_{obj}(X_{obj}, Y_{obj}, Z_{obj})$, whose depth needs to be computed with respect to the reference point $P_{ref}(X_{ref}, Y_{ref}, Z_{ref})$ as shown in the Figure 2.7. The pattern of the speckles is uncorrelated which means that within a pixel window the geometric arrangement of a speckle with respect to its neighboring speckles will be dissimilar to any other pixel window in any transverse plane. Hence, there will be a match of this small window of the

pattern projected on the object image to a window on the reference image. A search is carried out to find the best match of the image pixels in the object plane to those in the reference plane to find the reference points. The X-direction shift of a pixel point in the object plane with respect to the reference plane can be found out once the mapping is performed. The Z-direction shift of the object point (P_{obj}) from the corresponding reference point (P_{ref}) is then found out using a triangulation principle. This is explained with the help of Figure 2.7.

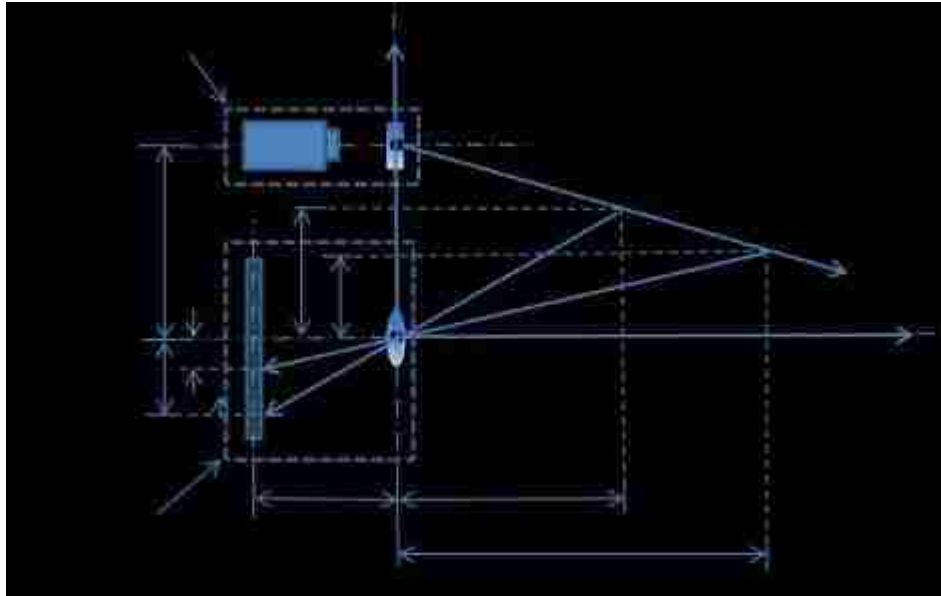


Figure 2.7. Principle of PrimeSense depth sensing technology

O_1 is the depth sensor which detects the reflected infrared beams projected from the source O_2 . By the triangulation principle, equations (1) and (2) give the relation of the x coordinate of the points $P_{ref}(X_{ref}, Y_{ref}, Z_{ref})$ and $P_{obj}(X_{obj}, Y_{obj}, Z_{obj})$, respectively, in the object space and the image plane.

$$\frac{F}{x_{ref}} = \frac{Z_{ref}}{X_{ref}} \quad (1)$$

$$\frac{F}{x_{obj}} = \frac{Z_{obj}}{X_{obj}} \quad (2)$$

Also,

$$\frac{S - X_{ref}}{Z_{ref}} = \frac{S - X_{obj}}{Z_{obj}} \quad (3)$$

By combining equations (1), (2) and (3), we can obtain an equation correlating the shift in the X-direction and the shift in the Z-direction as follows:

$$\delta_x = \frac{SF\delta_z}{Z_{ref}(Z_{ref} + \delta_z)}$$

where,

δ_z – Shift of the depth in Z-direction

δ_x – Shift of the pixel in X-direction

S – Distance between depth sensor and light source

F – Focal length of depth-camera

Z_{ref} – Depth of calibration plate from X-axis (in this case P_{ref})

Thus a 3D depth image with depth value associated with every pixel in the image plane is obtained. The skeletal tracking algorithm of Microsoft Kinect for Windows SDK is based on research carried out on real-time human pose detection in parts, based on these depth images [28]. A database consisting of a large number of pairs of depth and manually labeled body part images as shown in the Figure 2.8 is developed.



Figure 2.8. Pair of depth and hand labeled images [28]

This data base which covers 500K of such pairs of images, cover a variety of postures and is used to learn a classifier to understand the process of recognition of different body parts. After the learning of the classifier it accepts a single depth image as an input which is segmented into body part labels, with the parts defined to be spatially localized near skeletal joints of interest. Every pixel in the image is associated with a body part recognized by the classifier. The density of the pixels associated with the body parts is then used to propose the global 3D centers of these parts which are re-projected into the world space to generate confidence-weighted proposals of the 3D locations of each skeletal joint.

2.2.3. System Architecture. Kinect uses the Natural User Interface (NUI) library or the Application Programming Interface (API) to act as a bridge between the human-computer interactions. There are five known NUI libraries available, viz. OpenNI/NITE, Libfreenect, CL NUI, Microsoft Kinect for Windows SDK and Evolve SDK. Out of these, the Microsoft Kinect for Windows SDK [26] and the OpenNI/NITE [27] were used in developing applications in this research, where Kinect SDK was mainly used for developing the applications while OpenNI was used for calibration purpose only. Figure 2.9 shows the system architecture for Kinect SDK.



Figure 2.9. System architecture: Kinect for Windows SDK

Kinect sensor provides three data streams, namely, depth, color and audio. The Natural User Interface (NUI) for Kinect SDK consists of a software runtime which utilizes the depth data stream and converts it into skeleton joint data in the form of 3D joint positions of 20 body joints and the corresponding bone orientations. The runtime also provides support for speech recognition using the audio data stream and face tracking SDK for tracking human faces [29].

Figure 2.10 shows the three layered architecture for OpenNI. The bottom layer shows the hardware device, a Kinect sensor, which captures the visual scene. The middle layer represents the OpenNI open-source libraries for producing the body joint position data. It is a group of middleware components such as hand or gesture recognition, audio and skeleton tracking.

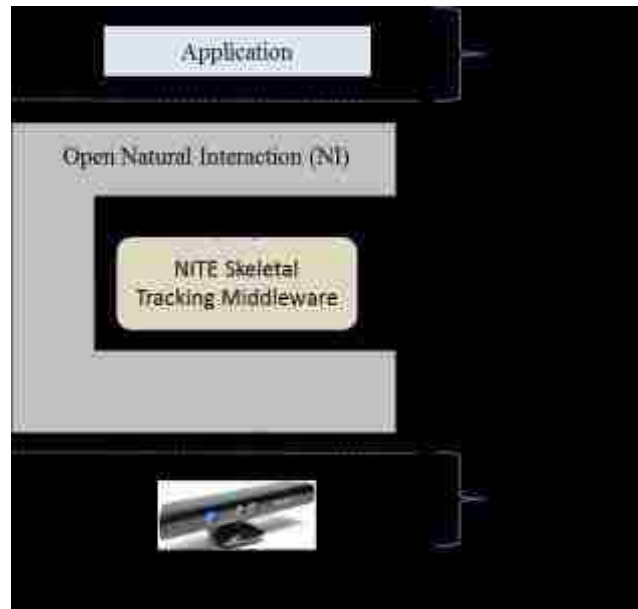


Figure 2.10. System architecture: OpenNI

PrimeSense's NITE (Natural Interaction Technology for End User) middleware offers a full-body skeleton-tracking algorithm based on the PrimeSensor depth images and translates these perceptions into meaningful data. Based on the scene segmentation output, the body of a user is tracked to output the current user pose, a set of locations of body joints. OpenNI supplies two sets of APIs, one to be implemented by the sensor devices and one by the middleware components. The APIs provide the 3D positions of different human body joints of the person being tracked.

3. DEVELOPMENT OF A MULTIPLE KINECT MOTION CAPTURE SYSTEM

3.1. MOTIVATION FOR DEVELOPING A MULTIPLE KINECT SYSTEM

The capture volume that can be covered by a single Kinect sensor for capturing human motions is limited. Moreover, the quality of data provided by a single Kinect depends on the orientation of the subject being tracked with respect to the Kinect. To achieve a larger capture volume and to improve the quality of motion tracking data, multiple Kinect devices have to be used simultaneously, with each Kinect covering a part of the desired capture volume. There are various challenges involved in developing a motion tracking system using multiple Kinects. While using multiple Kinect sensors simultaneously, it was observed that the quality of the data was reduced due to the interference of IR structured light patterns. Hence, the two main issues to be addressed while developing a motion tracking system with multiple Kinects are camera placement to maximize the capture volume and avoiding interference of IR light pattern from multiple Kinect cameras. In this section a new method to address these issues will be discussed.

3.1.1. Determination of Kinect sensor's tracking range. Kinect SDK's Natural User Interface (NUI) Skeletal Tracking (ST) system provides the position data of 20 body joints of a person being tracked. It has three tracking states, namely: *Tracked* (provides position and orientation of 20 body joints), *Not Tracked* and *Position Only* (provides only the position of the skeleton). The tracked state of each joint may be one of the following tracking states: *Tracked* (the joint is tracked and the data can be trusted), *Not Tracked* (the joint is not tracked and no data is available for this joint) and *Inferred* (confidence in

the position data is very low). An “*Inferred*” state of a joint means that the position of this joint is calculated from the surrounding joint data rather than directly captured by the camera. Inferred joint data is less accurate: it is more likely to have temporary spike noise and the random noise levels are usually higher [30].

In practice, there is noise associated with the raw joint-position data returned by the ST system. These characteristics and noise levels are dependent upon numerous parameters such as room lighting, person’s body size, person’s distance from the sensor, person’s position in the field of view of the camera, the orientation of the person with respect to Kinect sensor and the person’s postures. There are two types of noise present in joint positions from the ST system: (1) white noise that is always present for all joints due to imprecision and, (2) temporary spikes caused by inaccuracy, which happens when the joint has an inferred tracking state [30]. The joint positions are generally inferred when the ST system lacks information from the captured frame to determine the individual joint position. This might occur in cases where these joints are self-occluded (occlusion due to interference of body parts in the field of view of the camera) and occlusion by certain objects or moving a joint out of the sensor’s field of view. Self-occlusions are greatly dependent upon the orientation of the person being tracked with respect to the Kinect sensor. Therefore, the Human Interface Guidelines provided by Kinect for Windows SDK suggest that the user should face the Kinect sensor for best performance. It is necessary to understand the quality of the joint predictions with the variations in the orientation and posture of a person being tracked. Hence, an experiment was carried out to evaluate these factors.

A single subject was asked to pose in three different postures in front of a Kinect sensor at a distance of 2.2 m (range suggested by Windows SDK being 0.4 m (1.3 ft.) to 3 m (9.8 ft.)) in front of the sensor. The Kinect sensor was mounted on a tripod at 1m from the ground with 0° tilt angle and the lighting was neither too bright and nor too dim. The subject's clothing was neither reflective nor absorptive. The subject was asked to pose in each of the three different postures as shown in Figures 3.1 (a), (b) and (c).



Figure 3.1. Posture for accuracy analysis: (a) T-Pose, (b) Standing Relaxed Pose, and (c) Right Forearm Up Pose

For each posture, the subject was asked to increase the angle of orientation by 10° in the range from - 90° to 90° about the vertical axis. The 3D joint positions of each of the 20 skeletal joints as shown in the Figure 3.2 were recorded (500 frames) for each set. The lengths of fourteen different segments as shown in Table 3.1 were calculated by using the 3D positions of these joints.

Table 3.1. Segments formed using Kinect skeleton data

Segment Name	End Point Site 1	End Point Site 2
Torso	Hip Center	Shoulder Center
Head	Shoulder Center	Head
(R/L) Upper Arm	Shoulder	Elbow
(R/L) Lower Arm	Elbow	Wrist
(R/L) Hand	Wrist	Hand
(R/L) Upper Leg	Hip	Knee
(R/L) Lower Leg	Knee	Ankle
(R/L) Foot	Ankle	Foot

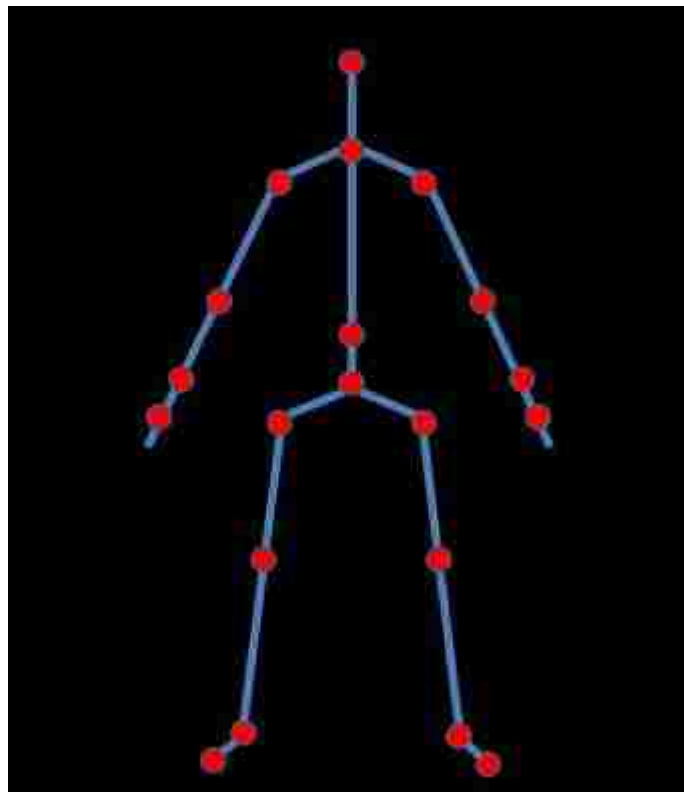


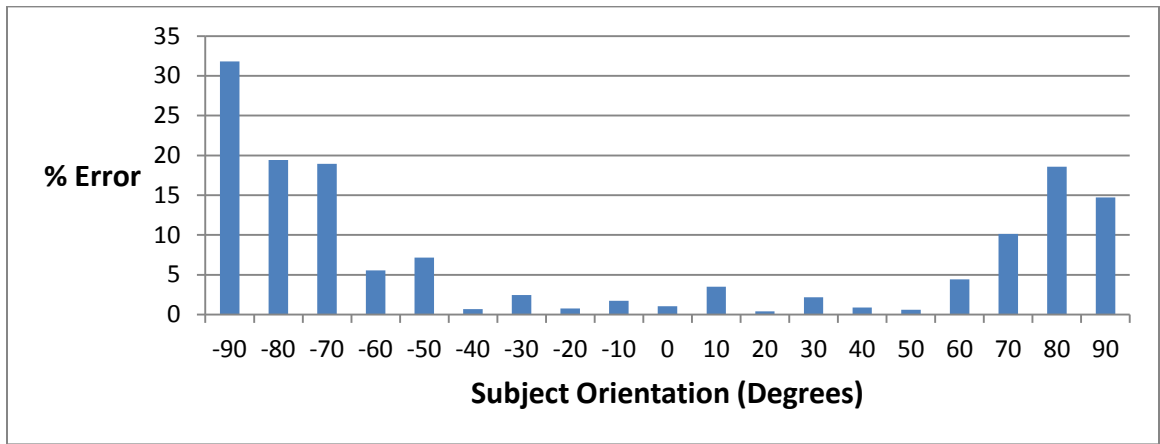
Figure 3.2. Kinect skeleton showing 20 body joints

Also, the corresponding body segment lengths of the operator were manually measured using a measurement tape with the help of key body joint positions. These positions were obtained with the help of an application called “Kinect Explorer”, available with the Windows SDK. This application draws a skeleton using the 3D positions of the body joints provided by the ST system and superimposes it on the human figure in the video stream as shown in the Figure 3.3. Another person used a probe to identify the physical position of these joints. Once the probe point coincided with the necessary skeletal joint on the video stream, it was marked. All the joints were marked in this way and further used for manual measurement.

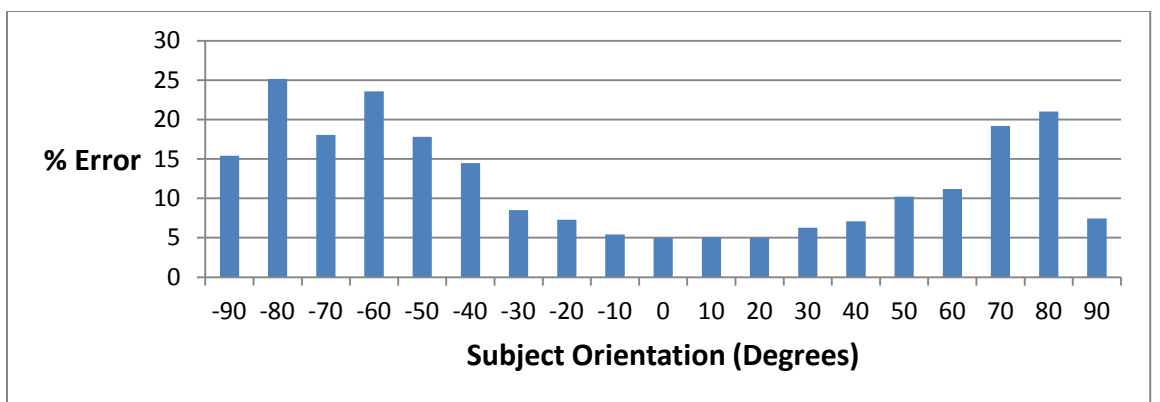


Figure 3.3. Kinect Explorer application’s video stream snapshot showing the skeleton superimposed on the video stream

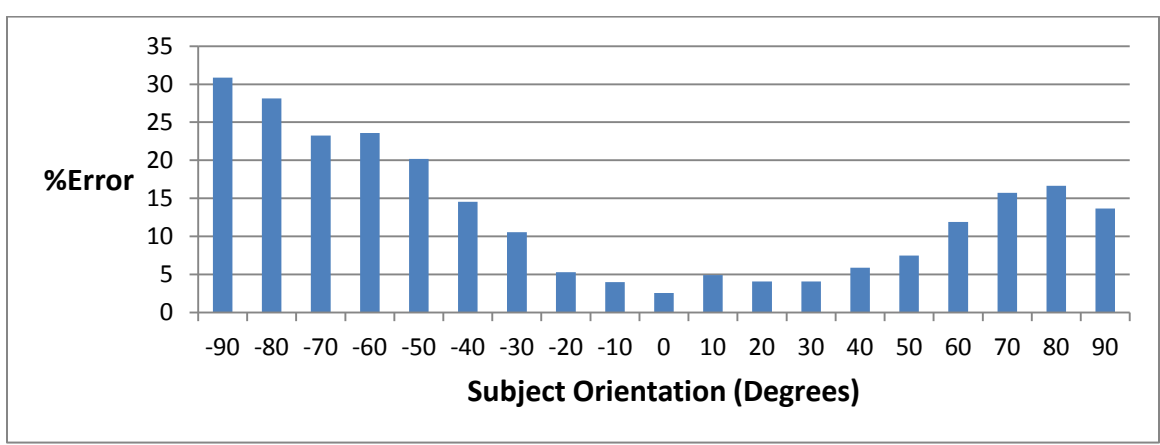
Mean percentage errors in these segment lengths were calculated. Figure 3.4 (a), (b) and (c) show relative errors in the torso segment lengths for T-Pose, Standing Relaxed, and Fore Arm Up postures, respectively.



(a)

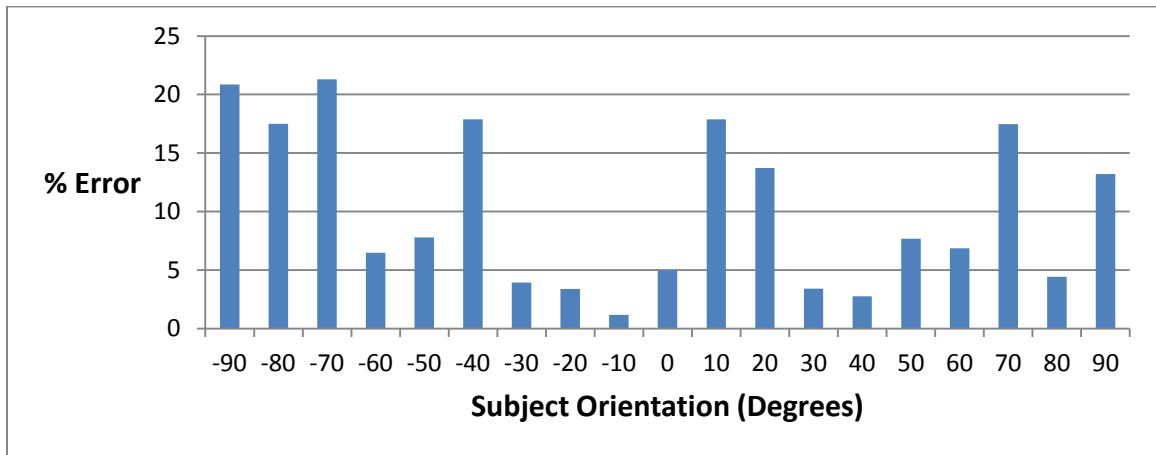


(b)

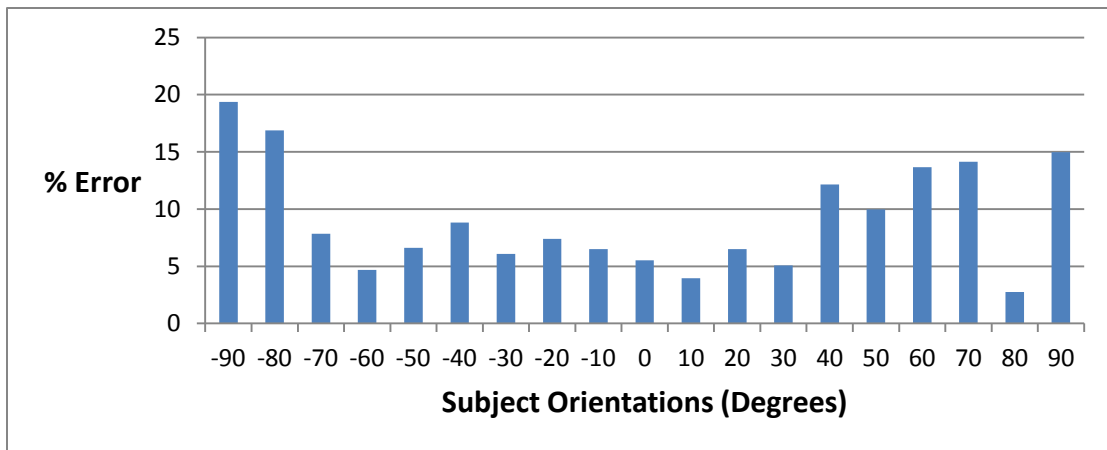


(c)

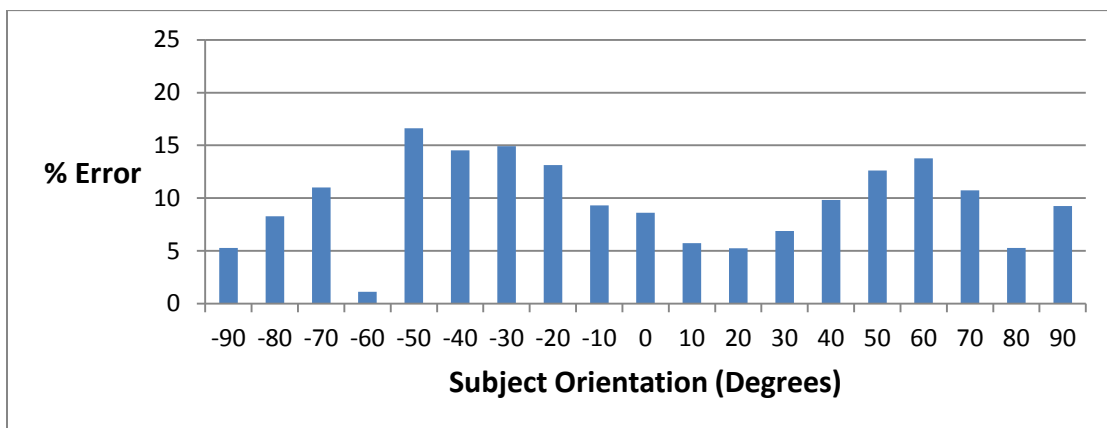
Figure 3.4. Percentage error in the Torso segment lengths at different subject orientations for different subject postures: (a) T-Pose, (b) Standing Relaxed Pose and (c) Right Forearm Up Pose



(a)



(b)



(c)

Figure 3.5. Percentage Errors in the Left Upper Arm segment length: (a) T-Pose (b) Standing Relaxed Pose and (c) Right Forearm Up Pose

As shown in Figure 3.4, the relative percentage error in the torso segment length varied from within 5% for low orientation angles to more than 30% for high angles for all the three postures. Similar trend (data not reported here) was observed for the Head (both max. 30%), Right Upper Leg and Left Upper Leg (max.10%). For Left and Right Lower Legs and Feet the errors were within 10 % between $\pm 70^\circ$ of orientation but did not show any trend for all the three postures. The relative error for the Upper Arms, Lower Arms and Hands varied with both posture and orientation. One example is shown in Figure 3.5.

It can be concluded from this experiment that the calculated segment length depends on both the orientation and the posture of the subject being tracked. Moreover, barring some cases the segment length measured, tends to be more accurate at lower orientations, as shown in Figure 3.6. The numbers in this figure were calculated as follows. First, the averages of the mean percentage errors of all the segment lengths per subject orientation were calculated. The averages of these percentage errors for all the three postures (T-Pose, Standing Relaxed and Fore Arm Up) were then calculated.

Also, it was observed during the experiment that most of the times the ST system was unable to recognize any static pose for subject orientations over $\pm 60^\circ$ for any posture. Often the subject had to perform certain motions to trigger the tracking in some other pose and then stand in the desired pose. Hence it is advisable to track the humans mostly at lower angles say $\pm 40^\circ$ to keep the overall good performance of the ST system as compared to that at higher angles. This was also a main motivation of our research for developing a multiple Kinects tracking system.

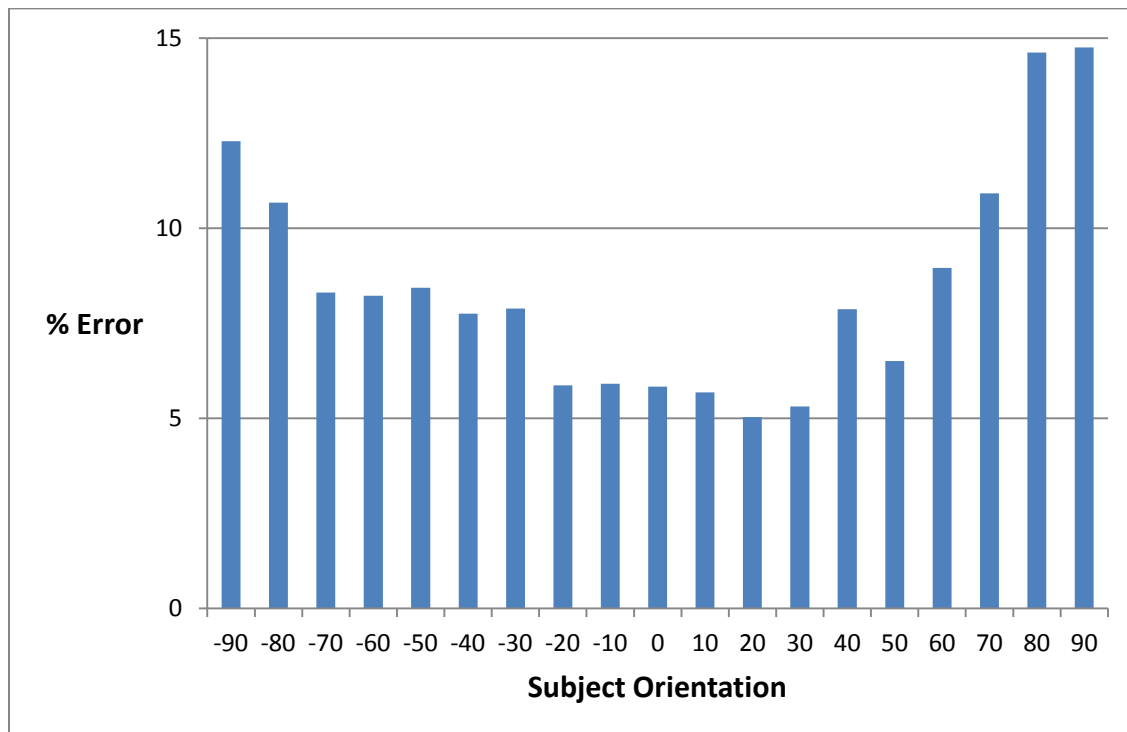


Figure 3.6. Mean of relative errors of segment lengths over different subject postures at different subject orientations

3.1.2. Interference of Multiple Infrared Beams. An experiment was carried out to investigate the loss of accuracy in the depth data from a Kinect due to interference of IR beams when used in sync with two, three or four Kinects. As shown in the Figure 3.7, it was observed that the quality of the depth image of a calibration plate decreased with the increase in the number of Kinect sensors. Note that the white spots developed in the image are invalid pixel coordinates.

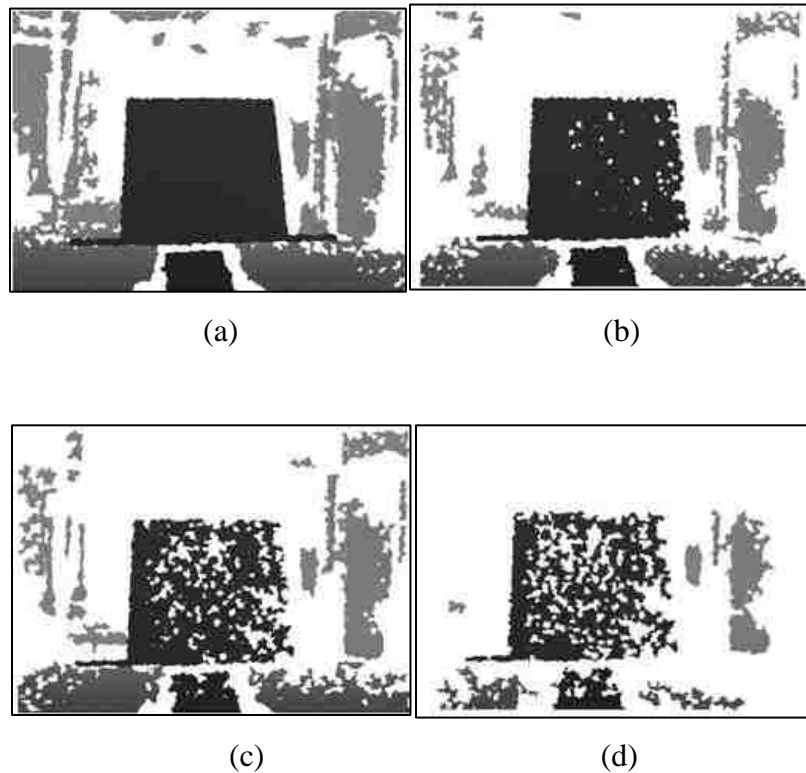


Figure 3.7. Degradation of the depth image quality with different number of Kinects facing the same surface: (a) 1 Kinect, (b) 2 Kinects, (c) 3 Kinects, and (d) 4 Kinects. Each image is of a calibration board recorded by one of the four Kinects

Another experiment was carried out to study the effect of interference on the skeletal tracking data. As shown in the Figure 3.8, a mannequin was posed in a *standing relaxed* posture in front of four Kinect devices. Different segment lengths were manually measured and recorded as discussed earlier. Then, these segment lengths were calculated from the joint position data obtained from the Kinect ST system. The other three Kinect sensors were switched on one by one and the segment lengths were calculated. The torso segment, which is the distance between the Hip Center and the Shoulder Center joints, was used as the test segment.



Figure 3.8. Mannequin

Table 3.2 shows the results of accuracy analysis for the torso segment. It was observed that the accuracy of the data reduced with increase in the number of Kinect

sensors. This is due to the interference of IR beams from multiple Kinect devices. Hence it is necessary to address this issue to develop a multiple Kinect system.

Table 3.2. Torso segment accuracy analysis (250 frames)

Segment	Torso			
Measured Length (cm)	55.12			
No. of Kinect sensors	1	2	3	4
Calculated Length from Kinect(s) (cm)	55.30	54.48	55.91	59.48
Mean Error (cm)	0.18	0.64	0.79	4.36
Standard deviation (cm)	0.05	0.19	0.24	2.07

3.2. LITERATURE REVIEW ON USE OF MULTIPLE KINECTS

As discussed in the previous section the fundamental issue of interference needs to be resolved to develop a multiple Kinect system. Several approaches have been investigated previously to address this issue of interference, including those from software and hardware sides. One approach is the time division multiple access (TDMA), which was used by Berger et al. [31] to develop a four-Kinect motion capture system. They mounted a fast rotating disk with well-defined slots in front of each of the four Kinect devices. These discs were controlled by stepper motors, which rotated at the same speed but with a different phase. These motors were synchronized such that only one laser could project at any given time. With four Kinects, the slots on the rotating discs were made such that each Kinect captures only two frames out of eight frames, with the

other six left for the remaining three Kinect devices. However, the shutter of the depth sensor would be opened and closed somewhere in the middle of the frame, resulting in less than one full frame exposed by the laser, i.e. less than a full depth image. The two depth images were then stitched together to obtain a full-frame image. This reduced the frame rate of the motion capture to one eighth of the original frame-rate (30 fps), i.e. 3-4 fps for each Kinect, which produce ambiguities for faster motions. This loss of frame rate is not suitable for obtaining good simulations of human movements. In another approach, Maimone et al. [32] demonstrated the use of external motors fitted on the each of the Kinect devices applying a small vibration. This resulted in each Kinect sensor detecting its own pattern sharply but seeing blurred patterns from other Kinects. Thus each sensor sees its own pattern with higher contrast than the patterns projected by other Kinects, thus reducing the interference. However, they also stated a noted increase in measurement instability. This instability in the measurement data may not be suitable for obtaining a stable simulation in Jack. Moreover, the RGB image quality was reduced and required image de-blurring techniques to restore its quality. Thus, there was a need to develop a system which would resolve the issue of interference without reducing the frame rate of the depth cameras as well as keep the data stable in order to obtain good human motion simulation. In this thesis, a novel approach is described to address the interference issue by developing a multiple Kinect system without reducing the frame rate or affecting the data stability.

3.3. MULTIPLE KINECT SYSTEM

This section discusses in detail the development of a motion capture system using multiple Kinect sensors.

3.3.1. Principle. The basic idea for addressing the issue of interference was to start the multiple Kinect devices simultaneously and let only one device project its light pattern (active state) at a time, depending upon the orientation of the subject being tracked. Considering that the skeletal tracking system works best when the person faces the Kinect device, the orientation of the subject with respect to the Kinect should be within $\pm 40^\circ$ as recommended in Section 3.1.1. The system must activate another Kinect device and deactivate the current Kinect if a pre-specified operator orientation is surpassed. Since only one Kinect will track the person at a time, the interference issue can be resolved. A simple approach is to activate and deactivate the light sources of the Kinect devices software-wise. However, there was a data loss for 1 to 2 seconds, the time duration required to switch the activation states between the Kinect devices. The next approach was to use external electro-mechanical shutter system to activate (uncover) and deactivate (cover) the Kinect IR emitters. This approach was used to develop the multiple-Kinect system and will be discussed in detail in the following sections.

3.3.2. Software and Hardware Architecture. The system architecture is shown in Figure 3.9. A computer application was developed using the Windows Presentation Foundation (WPF) of the C# .NET 4.0 Framework. Kinect for Windows Software Development Kit (SDK) and National Instruments (NI) NI-DAQmx Library for .Net 4.0 were used to develop the application. The hardware of the system consisted of two GWS PARK HPX F Mini Servo motors which were used to position shutter fins mounted in

front of the Kinect IR emitter. The motor mounts and the shutter fins were printed out using Stratasys Fortus 400mc, a 3D printing machine. The details of the individual components will be discussed next.

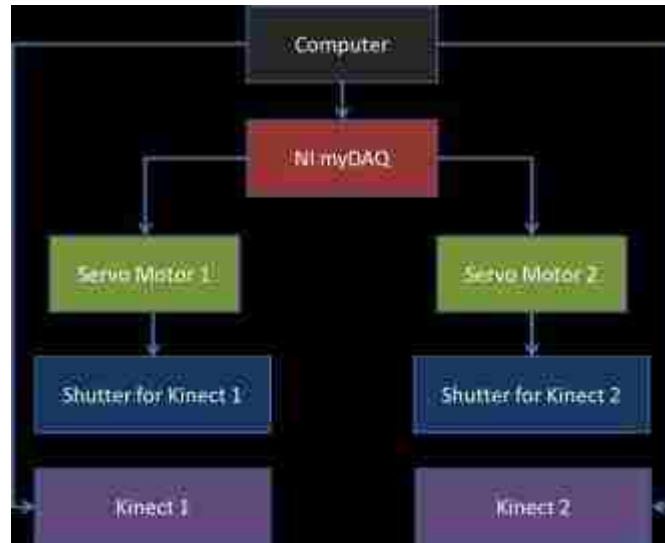


Figure 3.9. System architecture for two-Kinect system

3.3.3. Shutter Mechanism. This section describes the development of an external shutter mechanism to control the projection of the IR pattern from a Kinect sensor to avoid interference.

3.3.3.1. Shutter and motor mount. Figure 3.10(a) shows a Kinect device mounted on a tripod stand. The motor mount was designed in such a way that it could be sandwiched between the Kinect mount and the tripod top. The motor was fixed to the motor mount so that the shutter attached to the motor was in front of the IR emitter of the Kinect devices. A fin was attached to the motor output shaft which acted as a shutter which was designed with a slot to make it light-weight. Figure 3.10 (b) shows the exploded view of this assembly. Figures 3.11 (a) and (b) show the open and closed positions of the shutter, respectively.

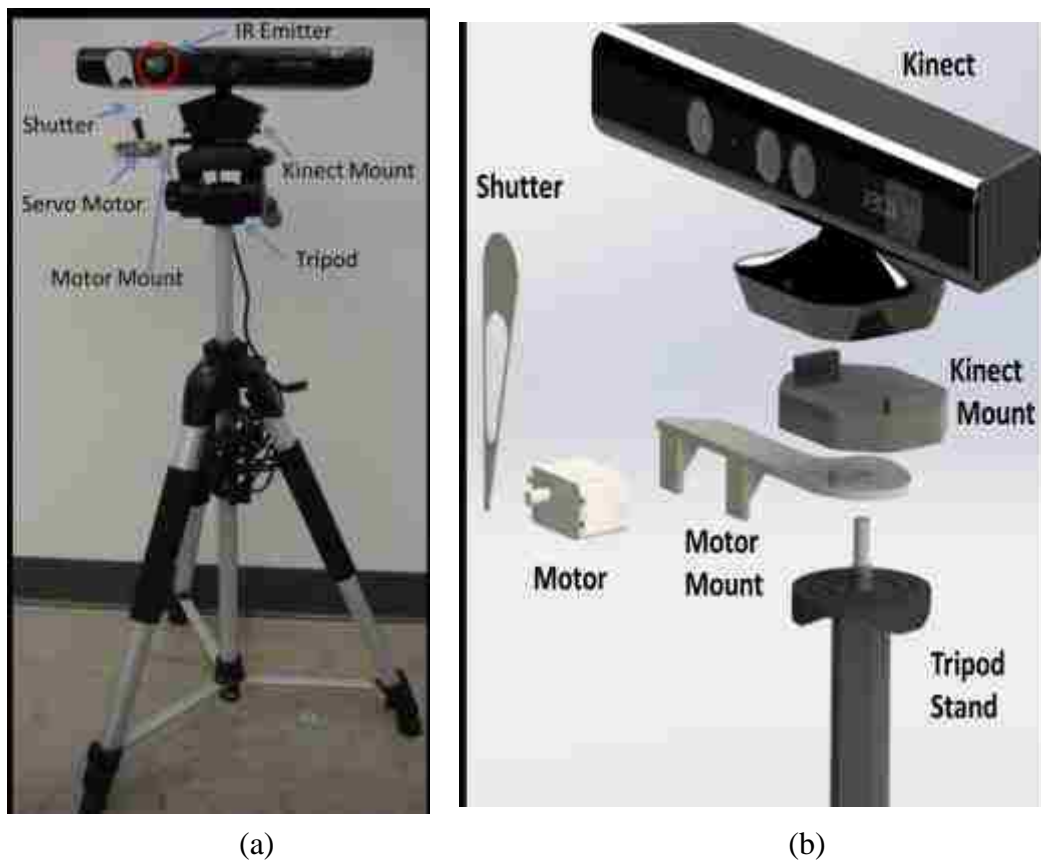


Figure 3.10. Shutter design: (a) assembly and (b) exploded view



Figure 3.11. Shutter positions (a) open and (b) closed

3.3.3.2. Servo motor. The basic criterion for selecting the motor is to ensure that there is no delay during the switching from one Kinect to another to avoid any loss of frames. Since the Kinect grabs the data at 30 frames per second (fps) it is desirable that the shutter position moves from *open* to *close* state in less than 1/30th of a second to avoid loss of frame rate. This can be achieved by using laser shutters. But these shutters were found to be very expensive [31]. Hence, our solution was to design a shutter and use servo motors to control the position of these shutters. GWS PARK HPX F Mini Servo motors are light-weight, quick-response RC servo motors with a speed of 0.05 sec/60° sec at 6 V. The servo motors work on the principle of negative feedback, where the control input is compared to the actual position. The control input is a pulse of varying length every 20 milliseconds. The pulse is normally between 1 and 2 milliseconds long. The length of the pulse decides the angular position of the motor output shaft. Figure 3.12 (a) (not to scale) shows the Pulse Width Modulation (PWM) diagram for 0° angular positions of the motor output shaft, and Figure 3.12 (b) shows the different angular position of the motor output shaft analogous to their respective pulse widths.

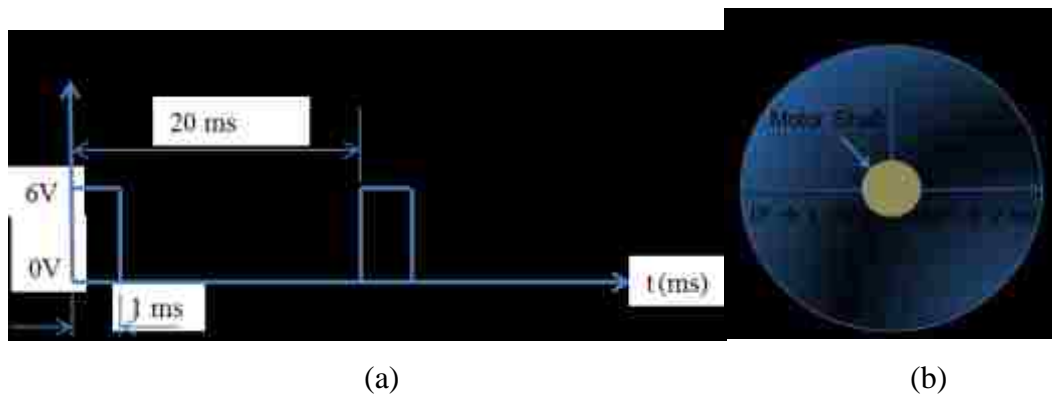


Figure 3.12. Principle of servomechanism: (a) PWM for 0° motor shaft position (b) motor output shaft positions

3.3.4. NI myDAQ. The control pulse required by the servo motor was generated with the use of a NI myDAQ [33], which is a portable data acquisition (DAQ) device. The myDAQ provides analog input (AI), analog output (AO), digital input and output (DIO), audio input and output, DC power supplies, and Digital Multi Meter (DMM) functions in a compact USB device. The PWM can be generated by using the counter/timer function which takes the frequency (Hz) and duty cycle (the width of the pulse divided by the pulse period). NI-DAQmx uses this ratio, combined with frequency, to determine pulse width and the interval between pulses as input parameters. There are 8 DIO channels, out of which 4 are allotted to the counter/timer functions; DIO 3 is allotted to counter output. Any of the remaining 7 channels can be connected to the counter output and hence the device can be used to control a maximum of 8 different servo motors. Figure 3.13 shows the output channels of NI myDAQ. Channels DIO 1 and DIO 2 are internally connected to channel DIO 3 and they are connected to motors 1 and 2 respectively.

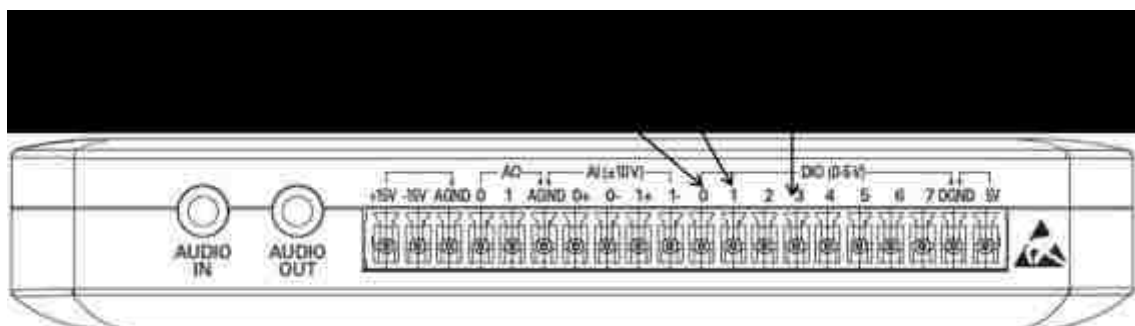


Figure 3.13. Physical I/O channels of NI myDAQ

3.3.5. Device Switching Algorithm. The objective of this algorithm was to switch the tracking states from one Kinect device to another depending upon the orientation of the tracked person. The first task was to start both Kinect devices and obtain their respective skeleton data simultaneously. Kinect SDK does not allow a single application to obtain skeletal data from more than one Kinect devices simultaneously. Hence, a multi-process application was designed to overcome this limitation. Figure 3.14 shows the design of a new approach to achieve the device switching for two Kinect devices.

The application consisted of a parent process which enumerated the Kinect devices and spawned two child processes in parallel, one for each Kinect device. However, since only one Kinect device should track at a time, it was assumed that the person was facing one of the Kinect devices (K_1) initially. This time the shutter in front of the IR emitter of K_1 was kept in the open state while that for the second Kinect device (K_2) was in the closed state. Thus the skeleton data from K_1 was obtained and stored. Once the person started turning towards K_2 which was placed at an angle of β° with respect to K_1 and surpassed a certain angle α° , switching took place and the shutter positions were toggled, opening the shutter for K_2 and closing the shutter for K_1 simultaneously, and let K_2 track the subject. However, when the shutters were toggled simultaneously, it was observed that there was 33% loss in the skeleton data during the switch as shown in Figure 3.15 where the blue and red line denote the number of frames for K_1 and K_2 , respectively, against a time instant and the green line denotes the total number of frames from K_1 and K_2 . The average frame rate during the switch was 21 fps. This issue was addressed by using the Inter Process Communication (IPC). Whenever a

switching took place, for instance from K_1 to K_2 , the shutter for K_2 was opened to activate skeleton tracking from K_2 , but the shutter for K_1 was closed only when the first set of skeleton data for K_2 was stored and a message was sent from K_2 to K_1 , thus assuring that there was no loss during the switch. The improved frame rate can be observed in Figure 3.16. The frame rate during the switch improved from 21 fps to 29 fps.

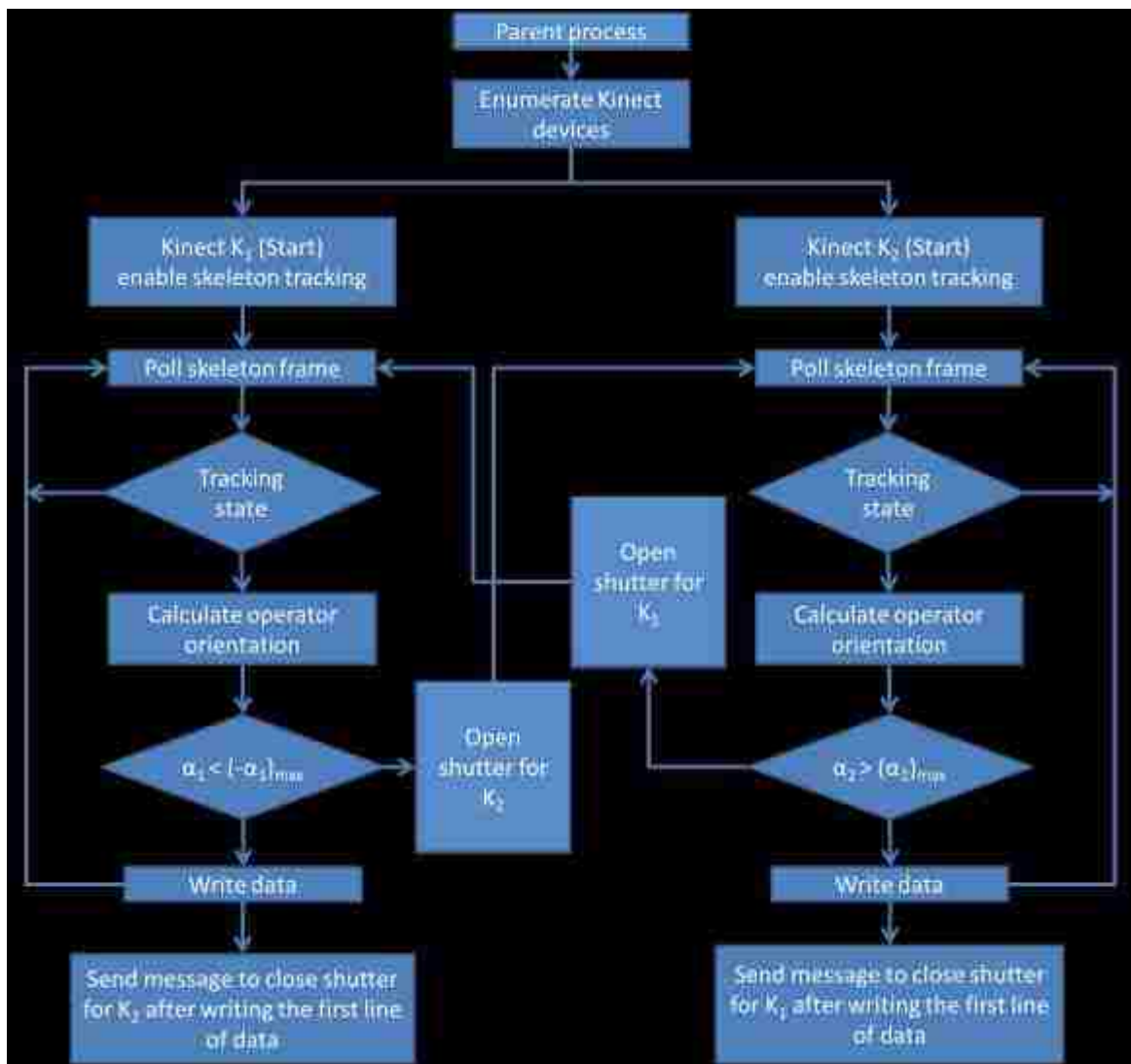


Figure 3.14. Device switching algorithm

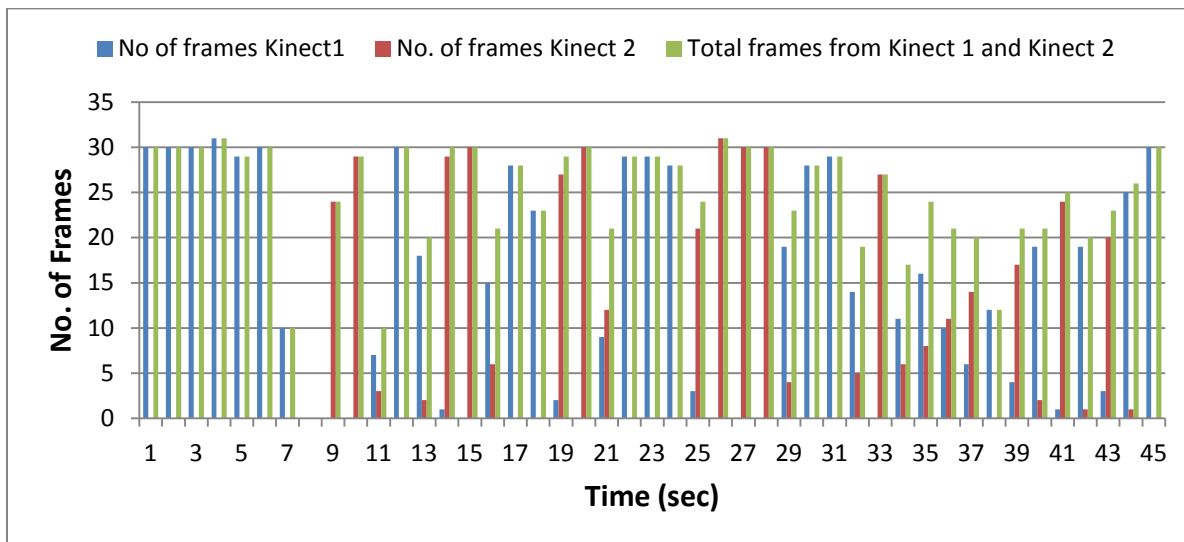


Figure 3.15. Loss of frame rate due to delay in switching

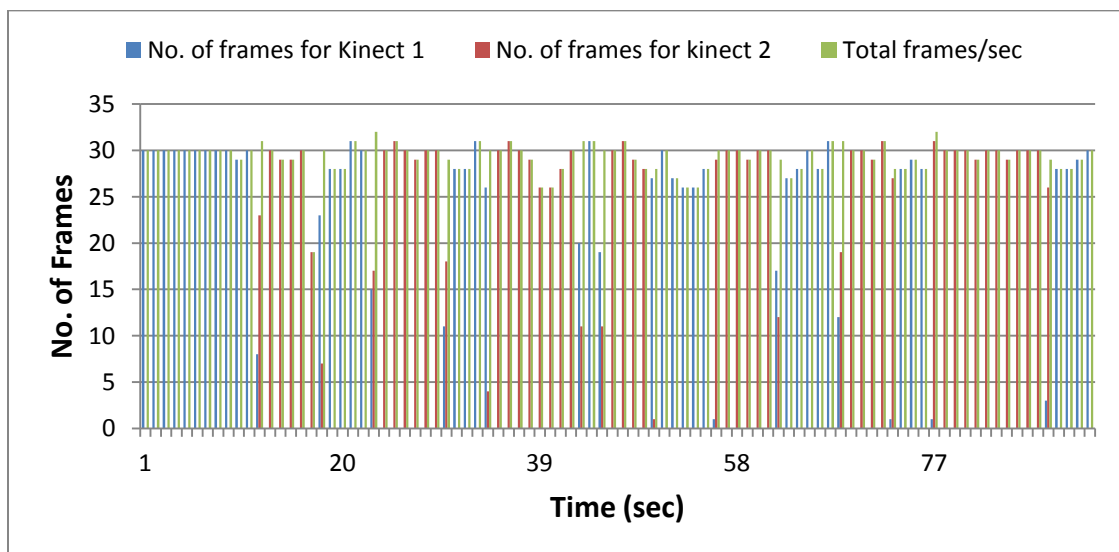


Figure 3.16. Improved frame rate during switching due to inter-process communication

Note that if the operator turned to the opposite sides than those discussed above for each Kinect device, and surpasses the maximum operator angle (α_{max}), a message was prompted on the screen showing a warning that the subject has moved to out of recommended tracking range, However, the respective Kinect will still record the data.

To cover these ranges more Kinects should be used. The use of three Kinect devices will be discussed in Section 4.

The placement of the Kinect devices and the operator orientation angle α , play a very important role in the switching algorithm. The calculation of this angle will be explained using Figure 3.17. The 3D coordinates of the left and the right hip joint of the operator are used to form a vector in the object space whose coordinate system is shown in Figure 3.17. The angle α is the angle between this vector and the unit vector along the X-axis. The angle β is the angle between the Kinect devices.

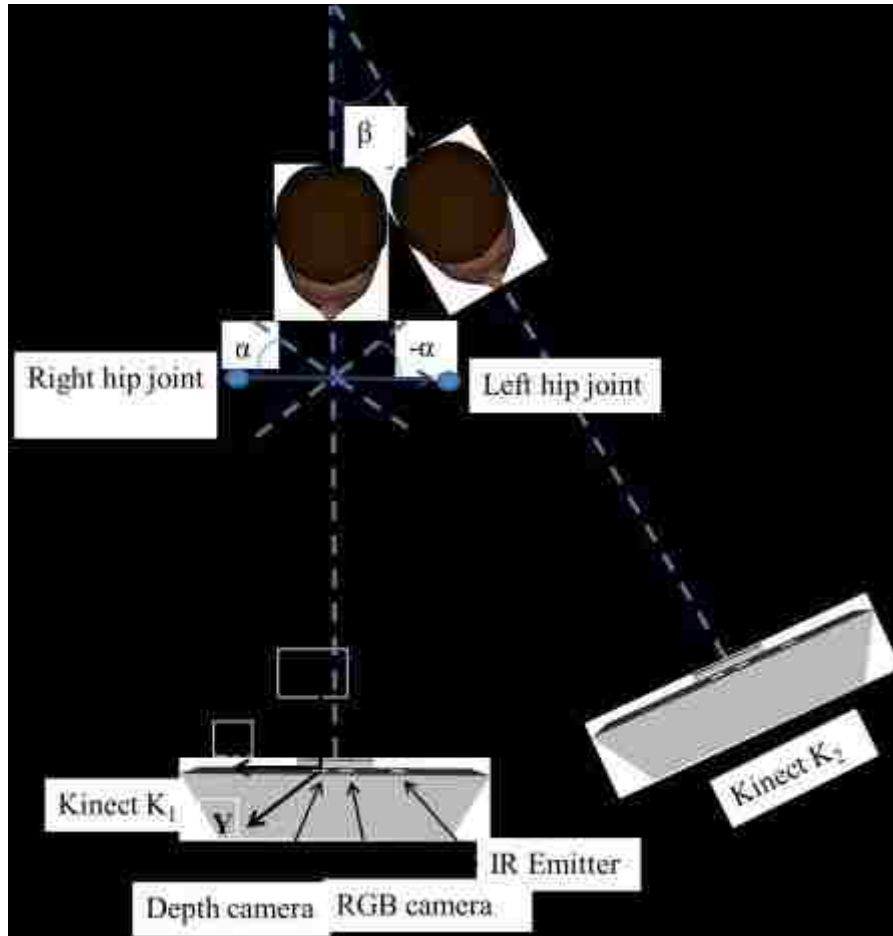


Figure 3.17. Operator angle calculation

Let $\alpha_{1, \max}$ be the maximum value of the operator angle α_1 with respect to the Kinect K_1 and $\alpha_{2, \max}$ be the maximum value of the operator angle α_2 with respect to Kinect K_2 before the switching takes place as shown in Figure 3.18. Note that these angles have the same value (α_{\max}) but, they are suffixed for distinguishing them with respect to their respective Kinects. The values of angle β and angle α_{\max} are interdependent as discussed below.

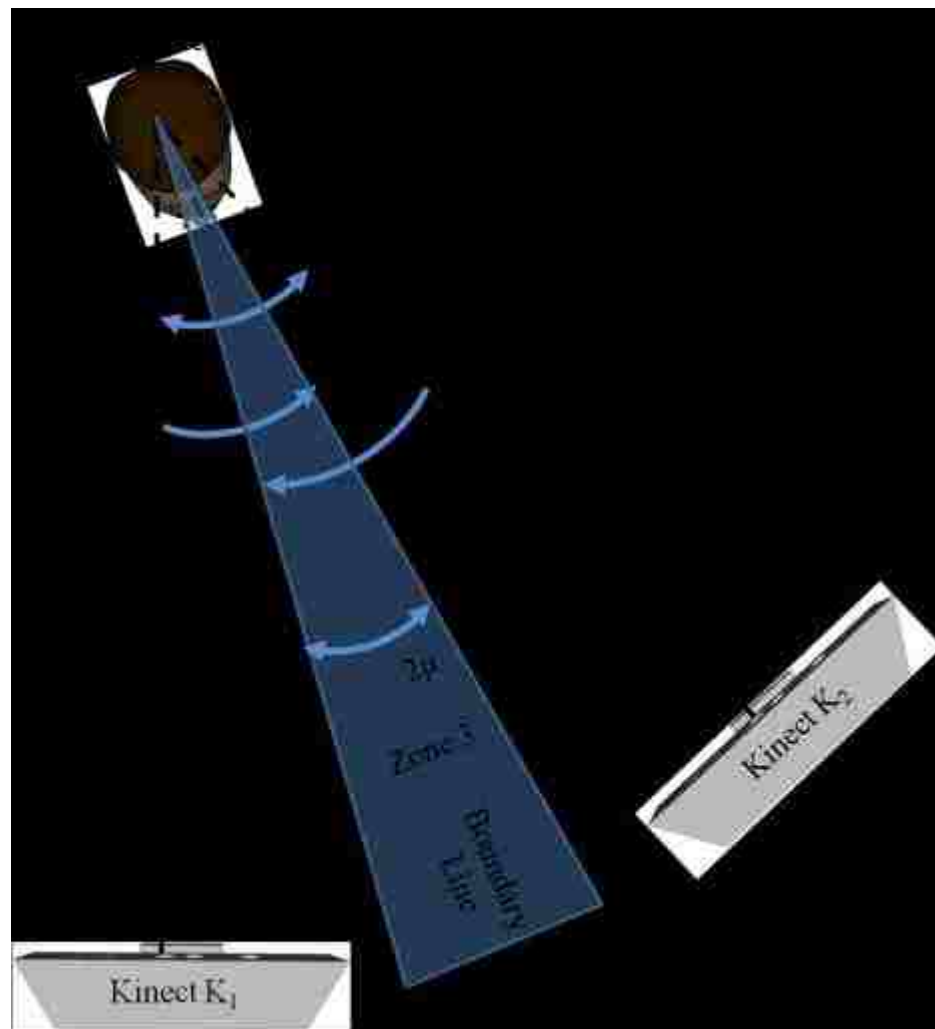


Figure 3.18. Selection of angle between two Kinects

As recommended in Section 3.1.1, the absolute value of α_1 or α_2 should not be greater than 40° and hence, the value of β cannot be greater than 80° . Now, consider a case where the value of α_{\max} is equal to $\beta/2$ and the switching happens at the boundary line as shown in the Figure 3.18. In reality there exists an error in calculating the operator angle due to the imprecision in Kinect data. If α_{\max} is set to $\beta/2$, then at the instance of switching at the boundary line, the error will result into misinterpretation of the switching states. Hence, an allowance angle μ needs to be provided to account for this error. This creates three zones in the tracking range of the two Kinects. Zone 1 is where only Kinect K_1 is tracking while Zone 2 was where only Kinect K_2 is tracking. Zone 3 is the overlapping area due to the allowance provided. So the following equation can be derived from the above observations,

$$\alpha_{\max} = \frac{\beta}{2} + \mu$$

However, if the value of angle β were to be equal to 80° , the value of α_{\max} would be larger than 40° as per the above equation. Hence the value of angle β was reduced by 2μ to accommodate the value of the allowance angle μ and to keep the value of α_{\max} equal to 40° . In our implementation to be discussed later, we let $\mu = 4^\circ$. So, if an area of 360° has to be covered, 5 Kinects would need to be placed in a circular fashion, each kept at 72° with respect to each other.

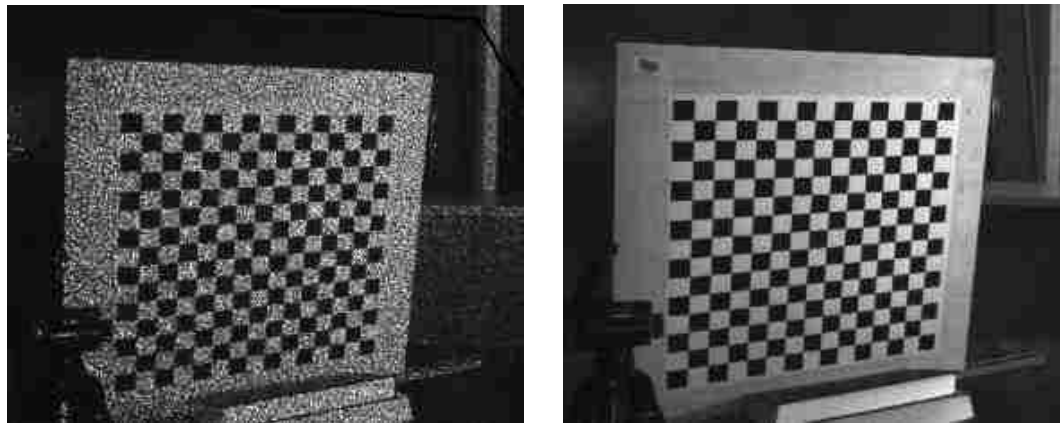
3.4. CAMERA CALIBRATION

A very simple approach was used to calibrate the depth cameras of the multiple Kinect devices. Stereo calibration of the depth cameras was performed with the help of a commonly used MATLAB camera calibration toolbox [34]. The IR images from the

depth cameras of multiple Kinect sensors can be used as an input to this toolbox.

However, Kinect SDK does not have the capability to use the IR stream from the Kinect depth camera and hence, the IR stream from OpenNI was used store the IR images.

Nonetheless, the IR pattern from Kinect is insufficient for obtaining good quality images which can be observed in the Figure 3.19 (a), so a halogen lamp was used to illuminate the environment to improve the image quality. Figure 3.19 (b) shows the IR image after using halogen lamp and covering the IR beam from the Kinect IR source.



(a)

(b)

Figure 3.19. IR images (a) with IR beam from Kinect sensor only and (b) with halogen lamp and Kinect IR source covered

The calibration process for two and three camera systems will be explained in the further sections.

3.4.1. Calibration of Two Kinect Devices. Once the IR images of the checkered board are stored from both the Kinect devices they are used as an input to the toolbox which provides the extrinsic parameters, which are the orientation and translation matrices for the two Kinect devices. The transformations for two-Kinect coordinate systems can be explained with the help of Figure 3.20.

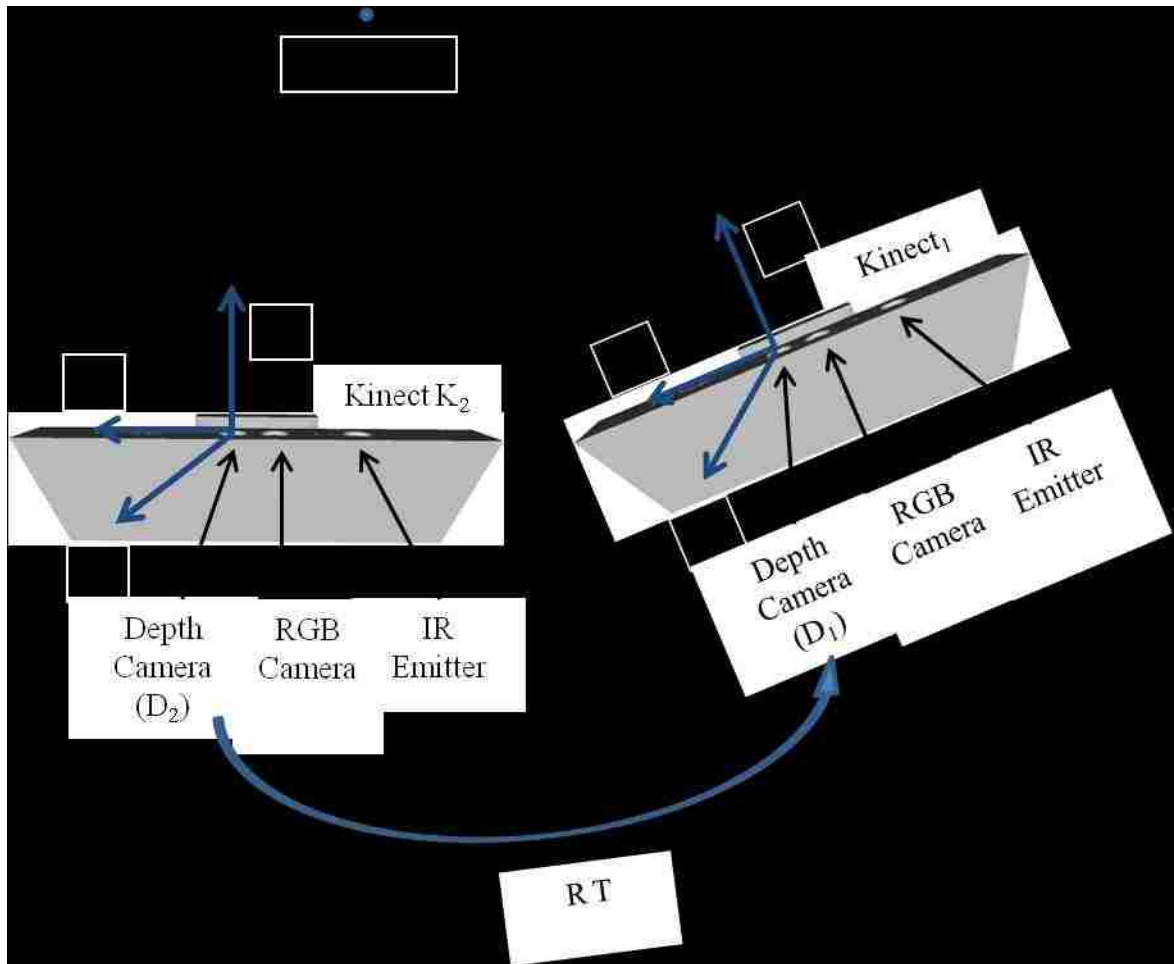


Figure 3.20. Calibration of two Kinect devices

Let D denote a 3D point in the depth camera coordinate system, R denote a rotation matrix and T denote a translation vector (Suffix 1 and 2 denote the camera number). The data obtained from Kinect K_2 is transformed to the coordinate system of Kinect K_1 using the below equation.

$$D_1 = RD_2 + T$$

3.4.2. Calibration of Three Kinect Devices. The calibration technique for three Kinect devices can be explained with the help of Figure 3.21. The depth camera of Kinect K_2 was considered as the master camera while the depth cameras of Kinects K_1 and K_3 were considered as the slave cameras. Thus K_1 , K_2 and K_2 , K_3 were calibrated separately as described previously and the master (K_2) coordinate system is considered as the world coordinate system.

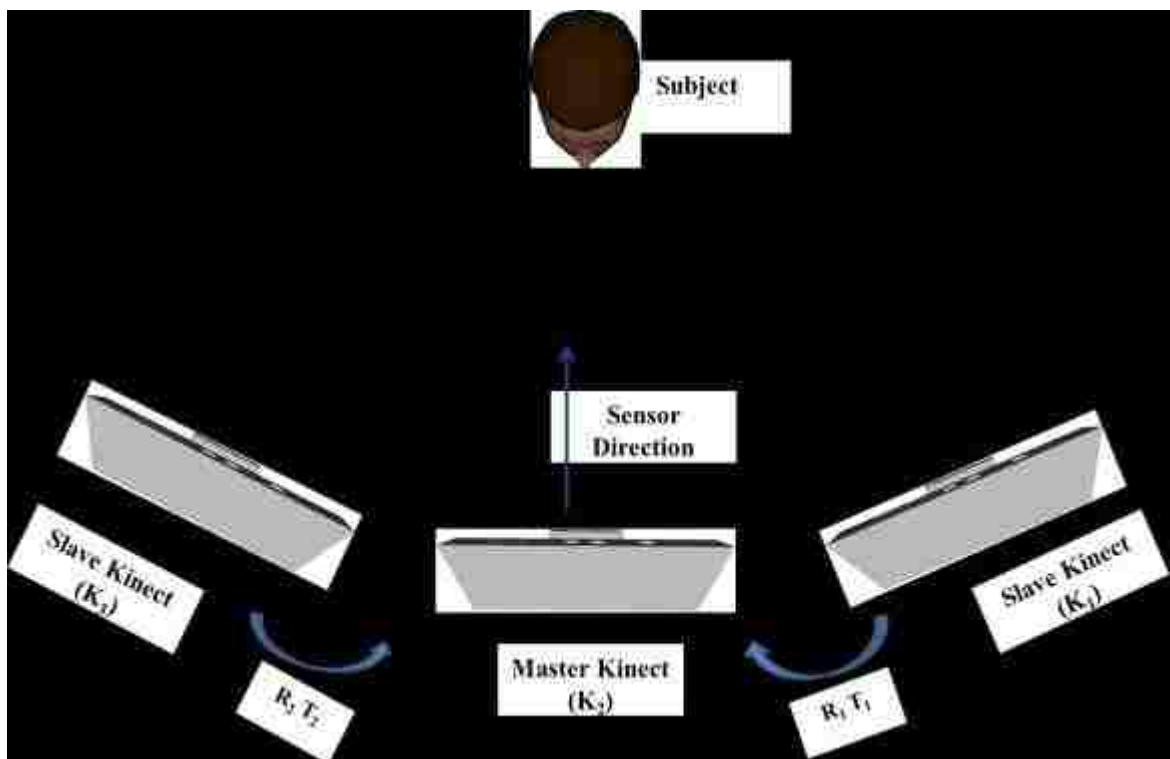


Figure 3.21. Calibration of three Kinect devices

The transformations for K_1 , K_2 and K_3 , K_2 camera coordinate systems can be given by following equations:

$$D_2 = R_1^{-1}(D_1 - T_1)$$

$$D_2 = R_2 D_3 + T_2$$

3.4.3. Calibration Results for Two-Kinect System. After calibrating the two Kinect devices an experiment was carried out to evaluate this calibration process. A mannequin was placed in front of the two Kinect devices in a standing relaxed posture. The skeletal data was recorded first using K_1 and then using K_2 . Calibrations results were applied to the K_2 and the data was transferred to the K_1 to make the coordinate systems consistent. Average absolute errors over five hundred frames of data of each of the 20 body joints was obtained for the mannequin's static posture. It was observed that the mean errors for 20 skeletal joints were -2.45 cm, 1.59 cm, and -1.78 cm for X, Y and Z coordinates, respectively, which might introduce a slight glitch in the simulation during a switch between Kinect devices which hardly affects the simulation. These errors might be introduced as a combined effect of calibration error and the difference between the device-centric intrinsic calibration results. Table 3.3 shows the mean errors in the X, Y and Z coordinates of some of the body joints for three hundred frames of data. In spite of these errors it will be proved that the simulation obtained using a two-Kinect system is better than the simulation obtained from a single Kinect system.

Table 3.3. Mean error in the 3D joints' positions (Unit: cm)

Body Segment	Mean Error		
	X	Y	Z
Root	-1.35	0.56	0.66
Neck	-2.18	1.33	1.07
Head	-3.32	1.43	1.08
Right Elbow	-2.81	1.71	1.43
Right Hand	-2.82	2.18	-1.31
Right Hip	-1.18	0.81	2.19
Left Elbow	-1.59	0.75	1.67
Left Hand	-0.66	2.24	1.73
Left Hip	-1.51	0.321	-0.86

An experiment was conducted to demonstrate the advantage of two-Kinect system over a single Kinect system using the above calibration results. Two Kinects K_1 and K_2 were calibrated with each other and placed at an angle $\beta = 40^\circ$ with respect to each other as shown in the Figure 3.22.

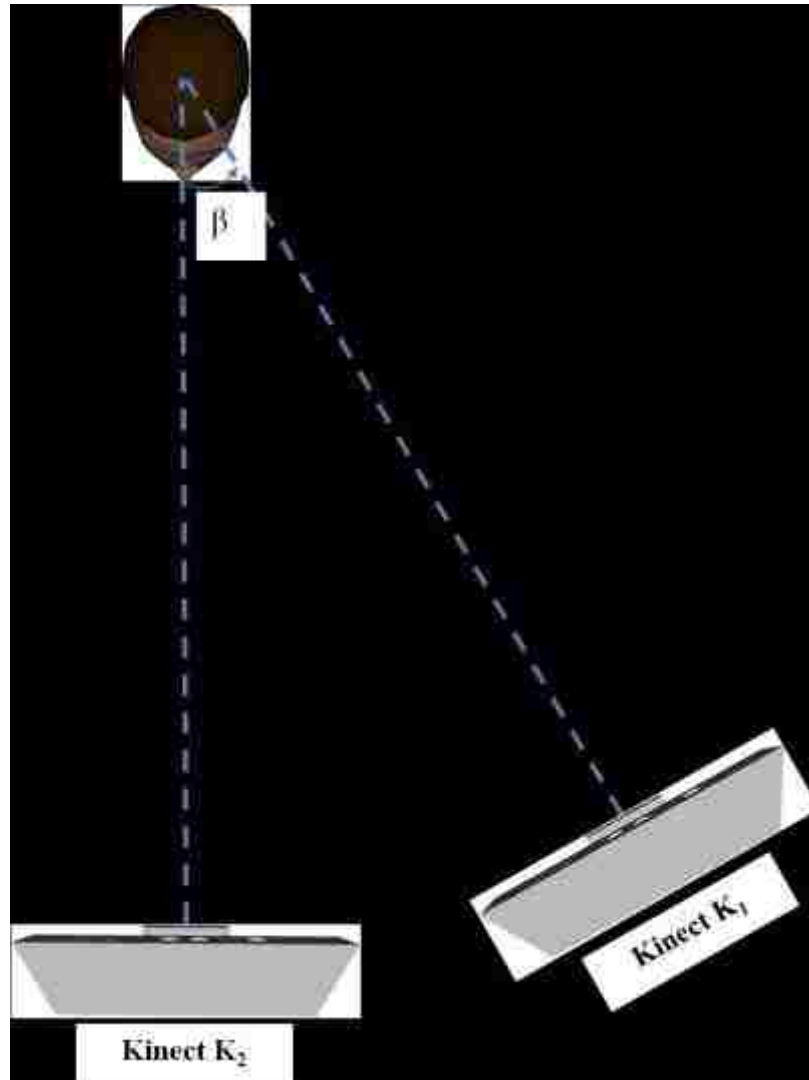
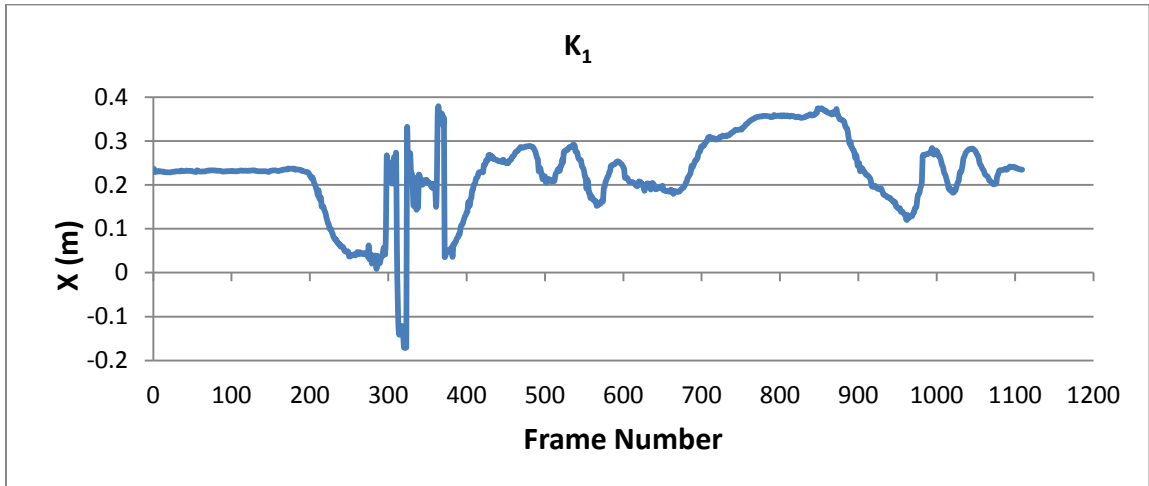


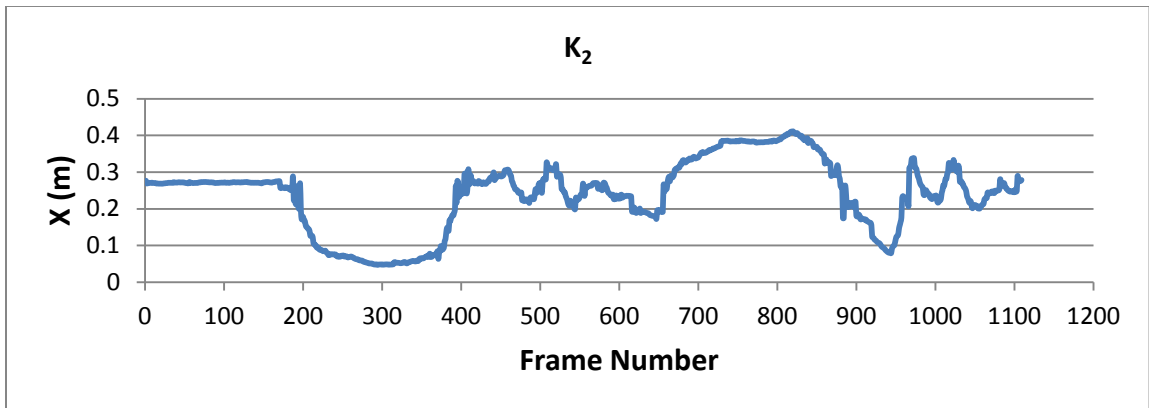
Figure 3.22. Experimental set-up

The calibration results were applied to K_2 to map its coordinate system to that of K_1 . A person performed a set of motions facing K_2 , and then turning towards K_1 to perform similar movements. The 3D positions of the right hand of the subject were recorded and plotted using the data obtained from both Kinects. To compare the data for the same set of motions, recorded using the two Kinect devices simultaneously, a switching algorithm was not applied for this scenario and the shutters were always kept

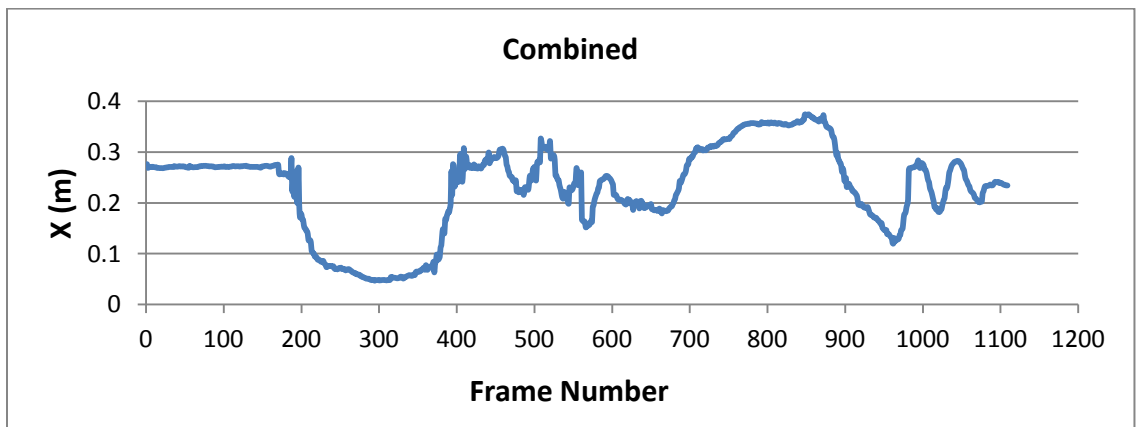
open for both Kinects. Since only two Kinects were used, interference was minimal. The data was separated based on the maximum operator angle (24°) calculated using the equation derived in Section 3.3.5, and then combined together. This combined data was then compared to the data obtained from the individual Kinects. It could be observed from Figures 3.23, 3.24 and 3.25 that since initially the subject being tracked was facing K_2 (standing at 40° with respect to K_1), the quality of the data obtained from K_2 for all X, Y and Z coordinates of the right hand was better as compared to that obtained from K_1 . Later, when the person turned to face K_1 , the data obtained from K_1 showed a better trend than the data obtained from K_2 . Hence, it could be concluded that the two-Kinect system would perform better than any of the single Kinects.



(a)

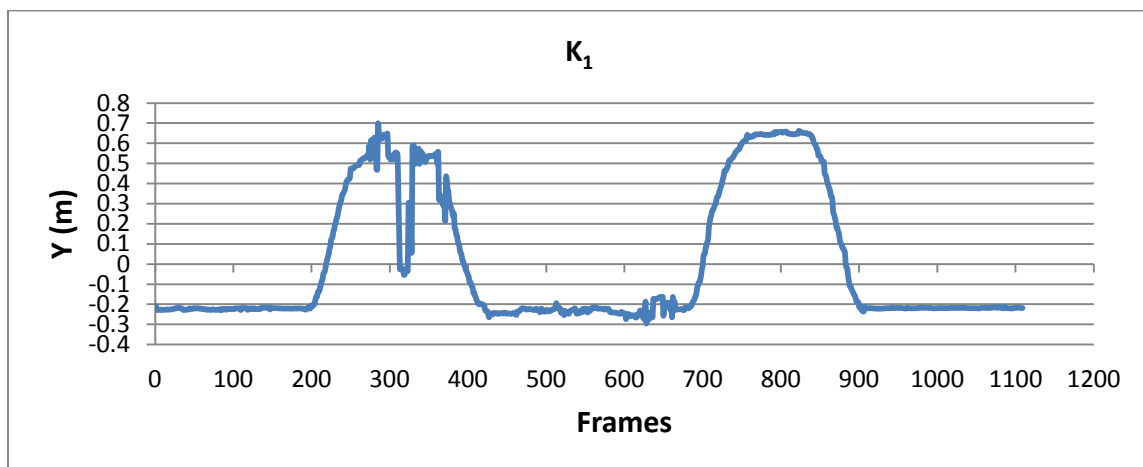


(b)

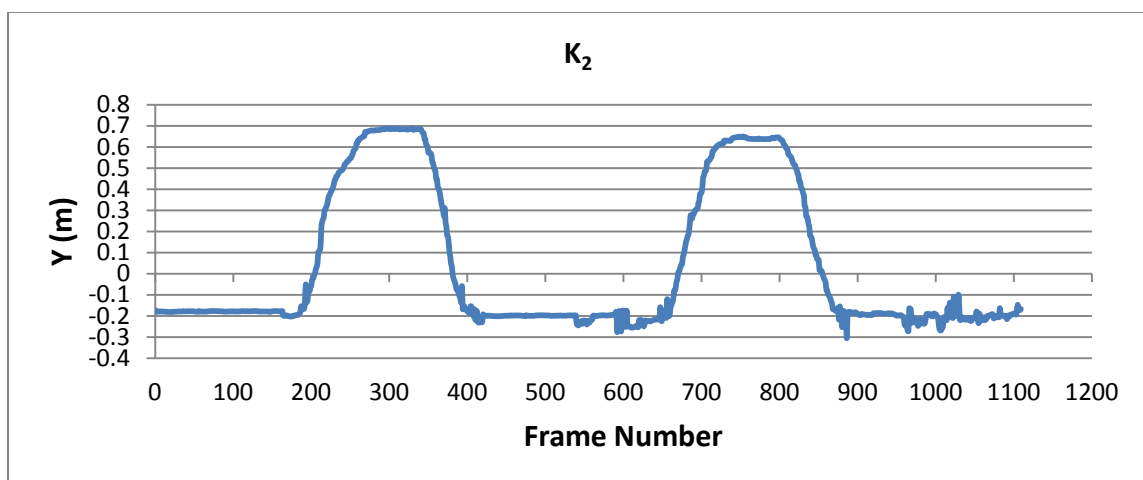


(c)

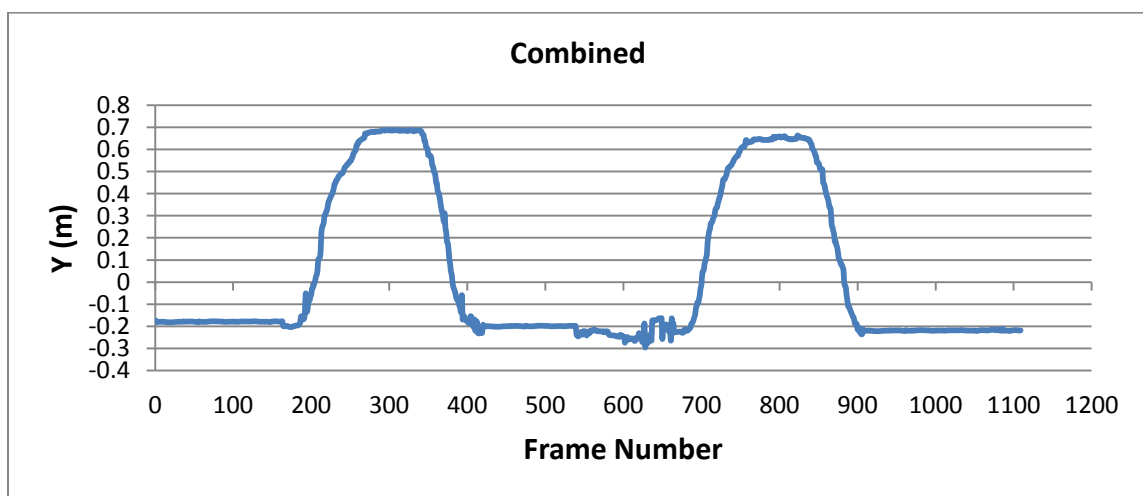
Figure 3.23. X coordinates of the right hand for: (a) K_1 , (b) K_2 , and (c) $K_1 + K_2$



(a)

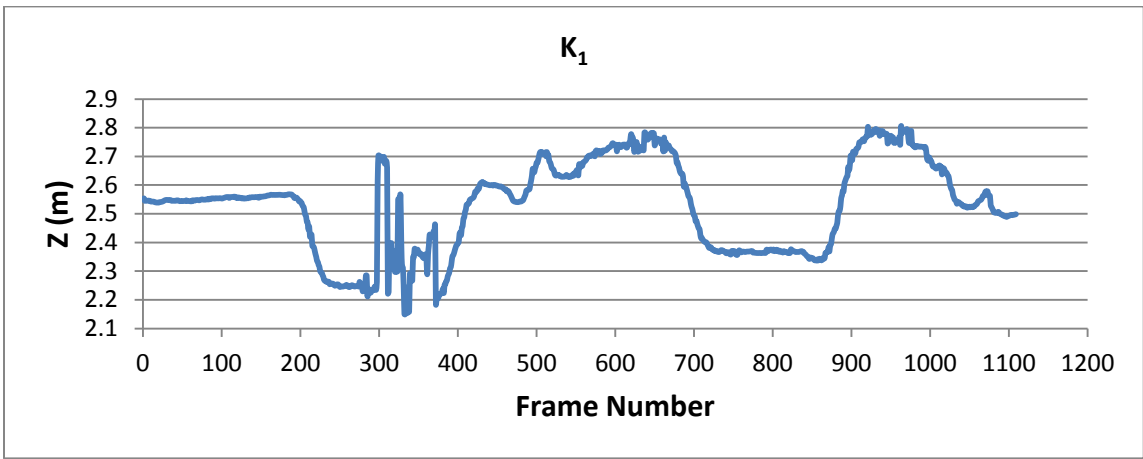


(b)

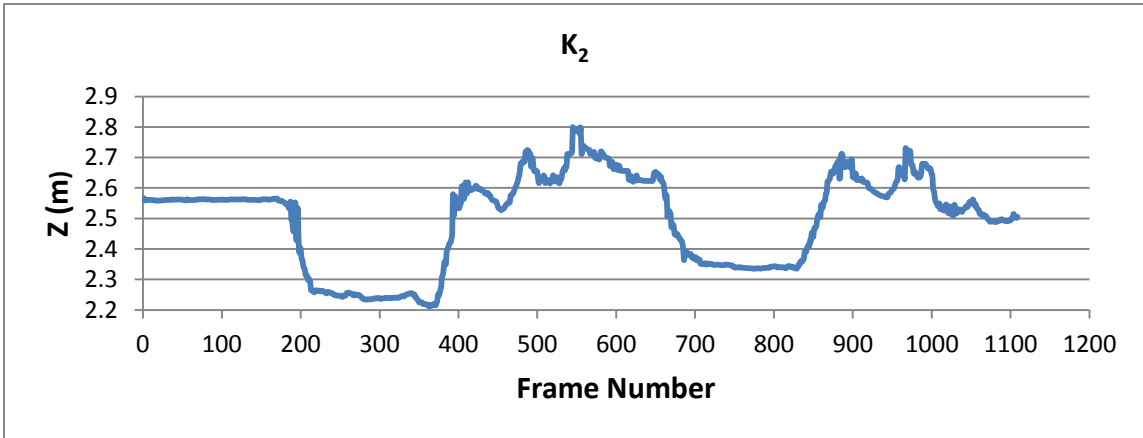


(c)

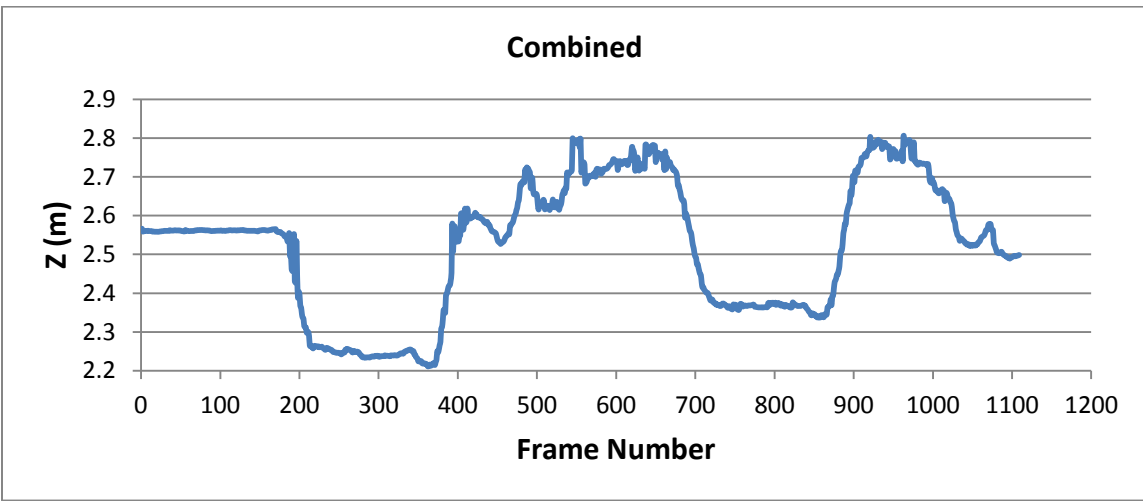
Figure 3.24. Y coordinates of the right hand for: (a) K_1 , (b) K_2 , and (c) $K_1 + K_2$



(a)



(b)



(c)

Figure 3.25. Z coordinates of the right hand for: (a) K_1 , (b) K_2 , and (c) $K_1 + K_2$

4. APPLICATIONS

4.1. DIGITAL HUMAN AUTO-SCALING APPLICATION FOR PARTIAL BODY MOTION CAPTURE

The skeleton tracking capability of Kinect could be utilized to scale a digital human model in different human modeling software packages. This section describes the development of an application for scaling the digital human model in a software package called Jack to assist simulations using partial body motion capture systems.

Technomatix Jack is a Siemens PLM software package for digital human modeling and simulation for Human Factors and Ergonomic Analysis. It provides different toolkits such as the 3D Body Scan to create humans using body scans (such as the SAE CAESAR Scans) and the Task Analysis Toolkit (TAT) to design better workplaces and maximize the safety of workers [40]. Apart from this, Jack also provides a MoCap module which supports human motion capture systems such as Vicon and Motion Analysis. With the release of Jack 7.1, Third Party Motion Capture Systems are also supported with a new User Interface. To understand how Jack is used for human motion simulation it is very important to understand the digital human model, which is discussed in the next section.

4.1.1. Digital Human in Jack. Badler et al. [36] described the human model used in the Jack software as a polyhedral model composed of 69 segments associated with 68 body joints. The human model in Jack is a very complex kinematic figure consisting of segments, joints and sites. Figure 4.1 shows the different body joints which are connected to form the Jack skeleton, which can be controlled by interactively manipulating the joints by specifying joint angles (forward kinematics) for one or more

degrees of freedom or, by manipulating a specified group of segments and joints (a "kinematic chain") via specifying a desired end-effector location and orientation (inverse kinematics). The later approach was chosen to manipulate the human model with the skeleton data from Kinect and will be discussed later.

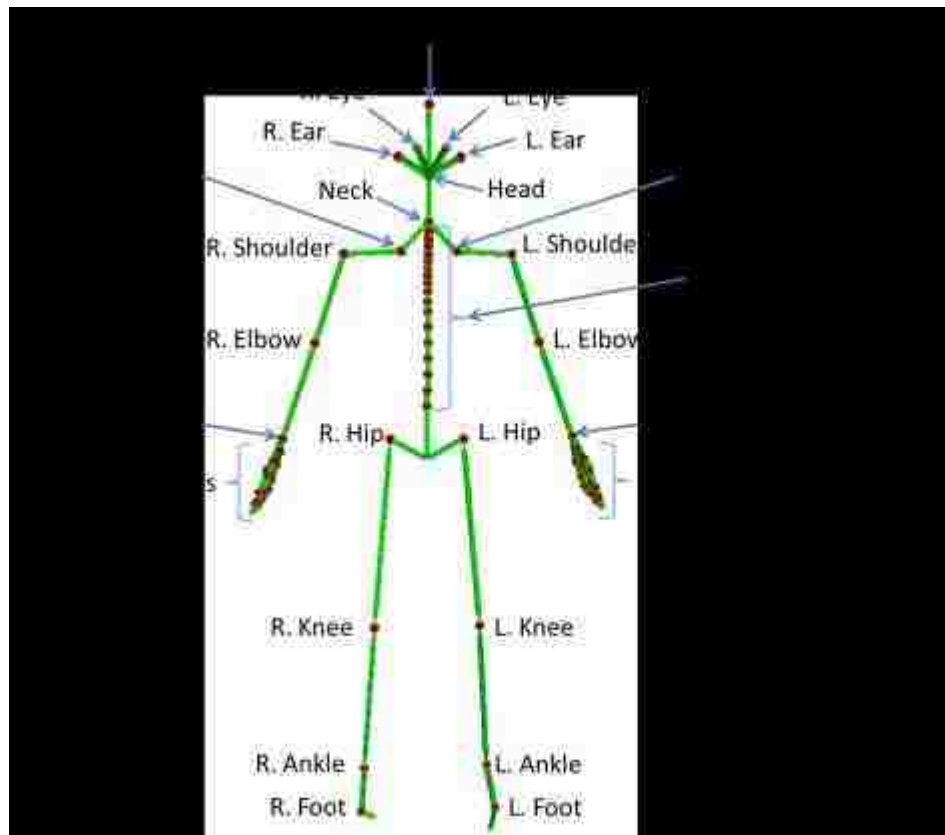


Figure 4.1. Jack skeleton [40]

4.1.2 Auto-Scaling the Digital Human Model Using Kinect Data. As discussed in the previous section, the digital human model in Jack is formed by different segments. Many default models are available in the Jack human library. Every human figure has some default segment lengths depending upon a chosen percentile population (e.g., 95th

percentile or 5th percentile). An accurate subject-specific scaling of the digital human model in Jack is desirable to obtain realistic postures during the simulation [35].

The user interface available with the Third Party Motion Capture feature of Jack enables the user to integrate the third party motion capture systems with Jack using a Jack MoCap Communication Protocol to obtain mostly real-time simulations. Its *Tracking Setup* consists of an *Auto-Scale* function to scale a digital human model using the motion capture data. However it requires the motion capture system to provide the 3D positions of 20 body joints which will be discussed later. Most of the full body motion capture systems can provide such data and can directly use this feature. However, motion capture systems that can track only a limited number of body joints do not provide sufficient data for using the Third Party Motion Capture feature. This issue of partial human body motion capture could be addressed by developing an application using Kinect that would calculate the lengths of the body segments to scale the human model in Jack. This was achieved by us through the digital human auto-scaling application. The architecture of this application is shown in Figure 4.2, and can be explained as follows.

The first step was to record the 3D joint position of 20 skeletal joints of a subject as shown in Figure 4.3, who stood in a T-pose in front of the Kinect device. The 3D position data was obtained using an application developed in Microsoft Kinect SDK [26]. This data was used to calculate the lengths of different body segments required for scaling the digital human model in Jack.

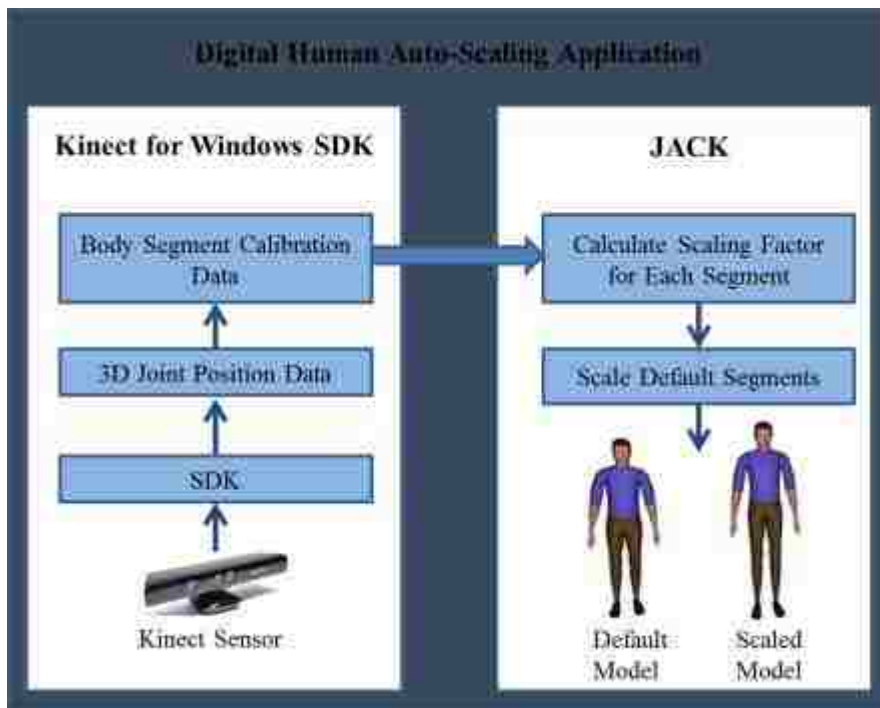


Figure 4.2. System architecture for digital human auto-scaling

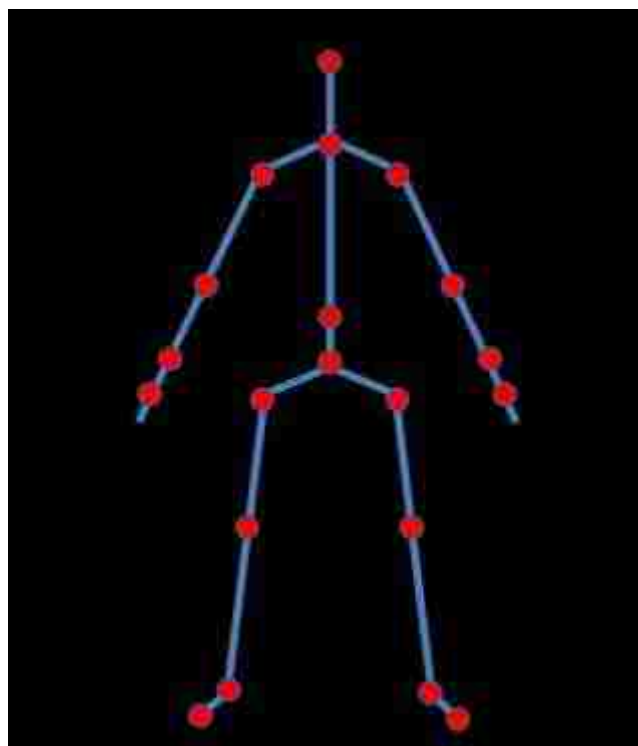


Figure 4.3. Kinect skeleton

Figure 4.4 shows the mapping between the Kinect skeleton model and the Jack skeleton model. Head (H), Upper Arm (UA), Lower Arm (LC), Hand (HA), Upper Torso (UT) (consisting of all 4 lumbar and 11 thoracic segments), Upper Leg (UL), Lower Leg (LL) are used to scale the Jack digital human model. Table 4.1 shows the end points of the above mentioned body segments. The clavicle, foot and hand segments were not scaled since the data for these segments was not available.

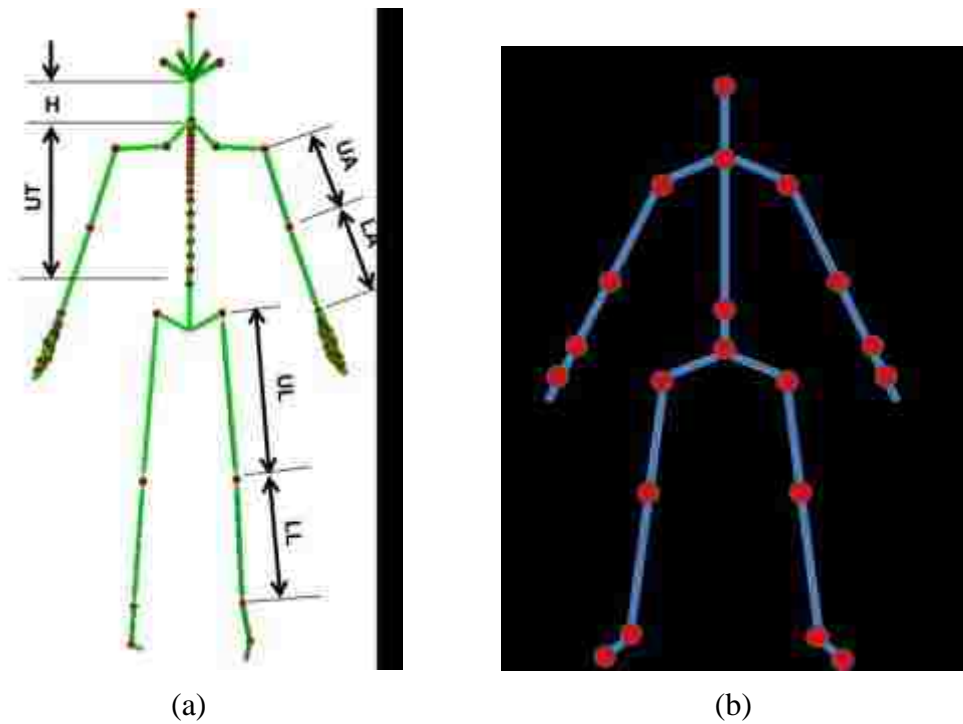


Figure 4.4. Segment definitions for: (a) Jack skeleton and (b) Kinect skeleton

Table 4.1. End points of body segments for Jack and Kinect skeletons

Segment Name	Jack		Kinect	
	Site 1	Site 2	Joint 1	Joint 2
Head	bottom_head.bottom	bottom_head.sight	Shoulder Center	Head
Upper Arm	right/left_upper_arm.proximal	right/left_upper_arm.distal	R/L Shoulder	R/L Elbow
Lower Arm	right/left_lower_arm.proximal	right/left_lower_arm.distal	R/L Elbow	R/L Wrist
Upper Torso	t1.distal	l5.proximal	Shoulder Center	Hip Center
Upper Leg	right/left_upper_leg.proximal	right/left_upper_leg.distal	R/L Hip	R/L Knee
Lower Leg	right/left_lower_leg.proximal	right/left_lower_leg.distal	R/L Knee	R/L Ankle

The second step was to calculate the scaling factor, which relates the length of each body segment obtained from the Kinect tracking system to the default length of the corresponding body segment of the digital human model in Jack. The calculation of these scaling factors were carried out with the help of an application developed using a Jack Script programming language which is an integral part of the Jack software. Scaling was done using a simple scaling factor given by the following equation,

$$SF = \frac{KL}{DL}$$

where SF is the scaling factor for the body segment, DL is the default length of body segment for the digital human model in Jack, and KL is the length of the corresponding body segment obtained from the Kinect tracking system. The scaling procedure for the upper torso and head segments needed different approaches.

The upper torso of the Jack human model comprises of 17 joints (5 lumbar and 12 thoracic) or 16 segments as discussed earlier. In order to scale the upper torso we needed to scale these intermediate segments. Firstly, the length of the upper torso segment was calculated using Kinect data. This length was then divided into 16 segments by multiplying it by the ratio of each intermediate segment length in Jack to the total length of the upper torso segment in Jack (calculated as the sum of all the intermediate segments). The lengths of each of these newly formed segments were then used to calculate the scaling factor for each segment. Lastly, these scaling factors were used to scale the individual segments of the upper torso.

The end points of the head segment of the Jack model do not match with those found with the Kinect data. Hence a different approach was used to scale the head segment. Firstly, a virtual point was considered at the top of the head for the Kinect data. A ratio of the segment length calculated from *bottom_of_head.top* to *bottom_of_head.sight*, to the segment length calculated from *bottom_of_head.top* to *bottom_of_head.base* was calculated. The actual segment length (*Shoulder Center to Head*) in Kinect was then multiplied by this ratio so that the length of the head segment from Kinect (*Shoulder Center to Top of Head*) corresponds to the length of the segment in Jack. After finding the virtual length of the head segment, it is used for scaling the head segment in Jack as discussed earlier.

For validating the digital auto-scaling application, the body segment lengths of five individuals were calculated using the skeletal joint position data recorded by a Kinect sensor. The first hundred frames were used for calculating the body segment lengths of every individual. The actual lengths were manually measured as discussed

earlier. Table 4.2 provides a comparison of the actual length of the right lower arm segment with the measurements obtained from our auto-scaling application. The segment lengths measured using the auto-scaling application showed good repeatability with a standard deviation less than 1 cm.

Table 4.2. Right lower arm segment measurements (Unit: cm)

Subject	1	2	3	4	5
Actual Length	33.00	30.00	36.00	30.00	32.00
Measured Length	33.08	30.31	36.47	29.52	31.94
	34.30	30.23	37.11	30.81	31.61
	33.04	28.70	36.29	31.09	30.73
	32.58	29.81	35.34	30.57	31.14
	32.55	31.34	36.82	30.89	30.36
Average Length	33.11	30.08	36.41	30.58	31.16
Standard deviation	0.71	0.95	0.67	0.62	0.64

4.1.3. Simulation Support. The auto-scaling application discussed in the previous sections was integrated with a human motion simulation application for partial body motion capture. This application can be described as follows.

To obtain the human motion simulations in Jack the data format required to be provided by the motion capture systems should be in the form of degrees of freedom of a joint that are the 3D position (X,Y,Z) in cm and Euler angles (x, y, z) in radians. Before transferring the motion capture data to Jack, the Jack environment must be set up. This involves creating the scene, scaling the human model, mapping the body joints, and creating constraints in Jack. These tasks are time-consuming and require substantial knowledge on Jack software. Therefore the simulation process was automated to

minimize the time required to perform these tasks. A module was developed in Jack using the Jackscript programming language to initialize and simulate human motion in Jack. It also provided the flexibility to play back the simulation of human motions for any number of body segments.

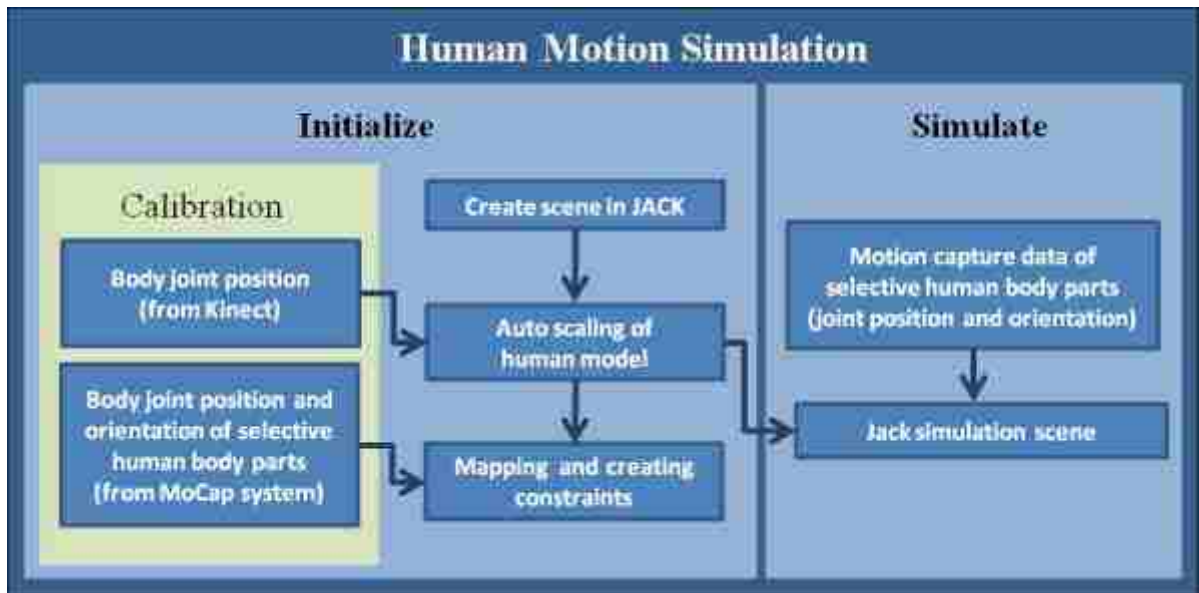


Figure 4.5. System architecture for human motion simulation

The simulation process was categorized into two domains, as shown in Figure 4.5. The first domain is the Initialize function which has three tasks: (1) Create a scene in Jack by creating different figures, such as the human model and the objects such as tools, chairs, CAD models, etc., required to simulate the virtual environment in Jack. A marker triad (a three-axis figure in Jack possessing six degrees of freedom) is created in the Jack scene for each of the 20 body joints tracked by the Kinect sensor. The 3D position and orientation of the human body joints of the subject in the real world tracked by the Kinect sensor are represented by these marker triads in the virtual scene in Jack environment. (2) Auto-scale the digital human model in Jack as discussed previously, and (3) mapping the

coordinate systems of Jack and motion capture system, and constraining the selective number of segments of the Jack human model to their respective marker triads.

The Jack digital human model is a hierarchical model having a tree-like structure, with a root site and then branching out into upper and lower body, leaf joints as shown in Figure 4.6. These joints are connected to each other by segments. In this model every top level segment will act as a parent to the low level child segment.

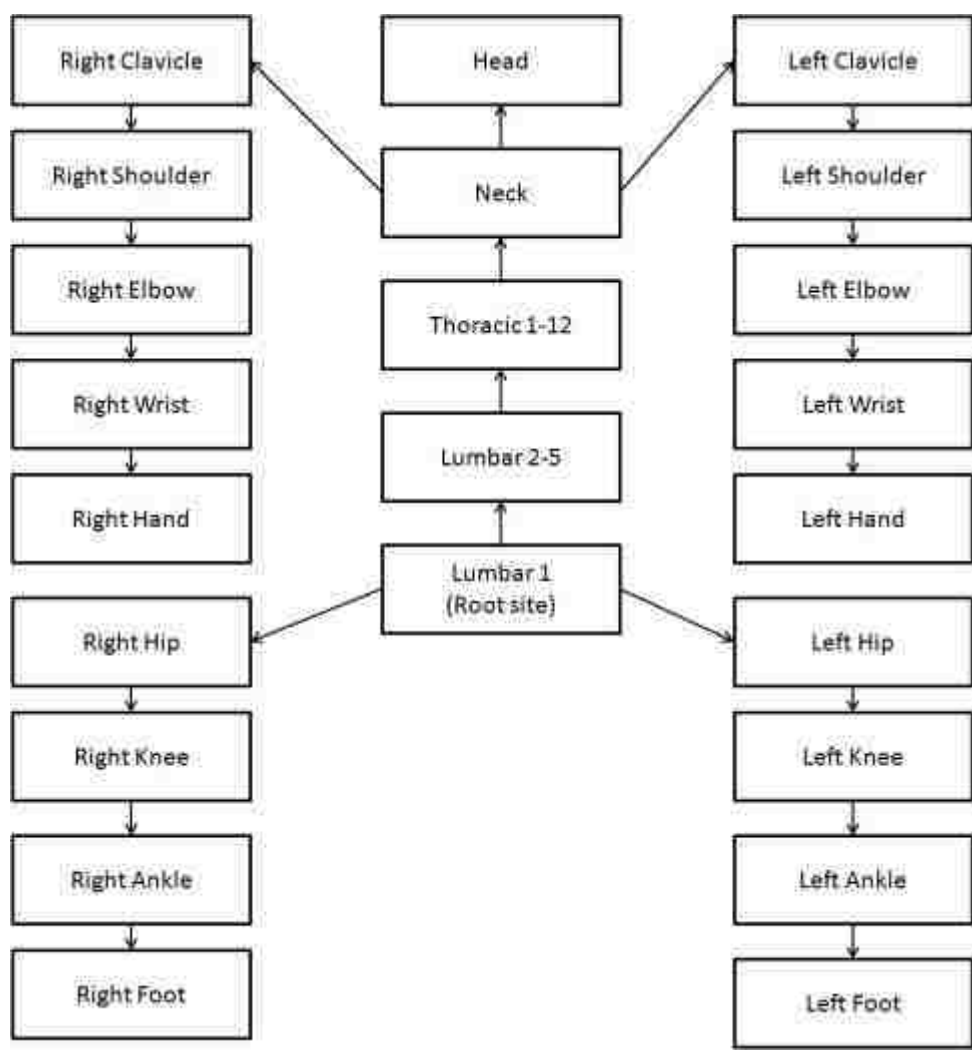


Figure 4.6. Hierarchical representation of Jack digital human model

As we go down the hierarchy, every child body segment is connected to its parent body segment at their respective joints with some geometrical relationship within the Jack environment called constraints. Constraints are of two types. (1) Modeling constraints are the constraints formed between each of the segments of a human model to develop an articulated figure. These constraints define the number of degrees of freedom of each joint and ensure that the segments are always connected to each other. The length of each of the segments always remains constant i.e., each segment is considered to be a rigid body. The modeling constraints also include the inequality constraints on the joint angles. For example if θ is a joint angle corresponding to a single DOF, and u and l are the upper and lower limits of this joint angle, respectively, then the inequality constraint for this joint angle is give by:

$$l \leq \theta \leq u$$

(2) Simulation constraints are those constraints which are desired to manipulate the posture of the human model in Jack, using the motion capture data. The locations of each of the marker triads created in the Jack environment using the motion capture data are used as reference locations or in terms of inverse kinematics they are called the goals. The joints on the human model which are called the end-effectors should be attached to these marker triads using the simulation constraints. These constraints define a spatial relationship in between all the 20 goal sites and their corresponding end-effectors, and demand that for every instance, the end-effectors be placed at the goal sites to obtain a desired posture of the human model. This relationship can be described in terms of position, orientation, or both.

The combination of modeling and simulation constraints results into a complex simulation structure. To elaborate further on constraints, consider the case of a single constraint between the goal frame of a marker triad and an end effector frame of the right-hand palm of the human model in Jack as shown in Figure 4.7. To constrain the goal and the end-effector coordinate frames, first it is important to understand the concept of a constraint chain. A constraint chain is defined as a set of joints within the hierarchical structure with a starting joint, a number of intermediate joints leading to an end-effector. For example, one constraint chain is from the shoulder joint (starting joint) with the elbow and the wrist joints as the intermediate joints and the end-effector or the right-hand is constrained to its corresponding goal or the right-hand marker-triad as shown in the Figure 4.7. A number of such constraint chains are created to define the simulation structure [41].

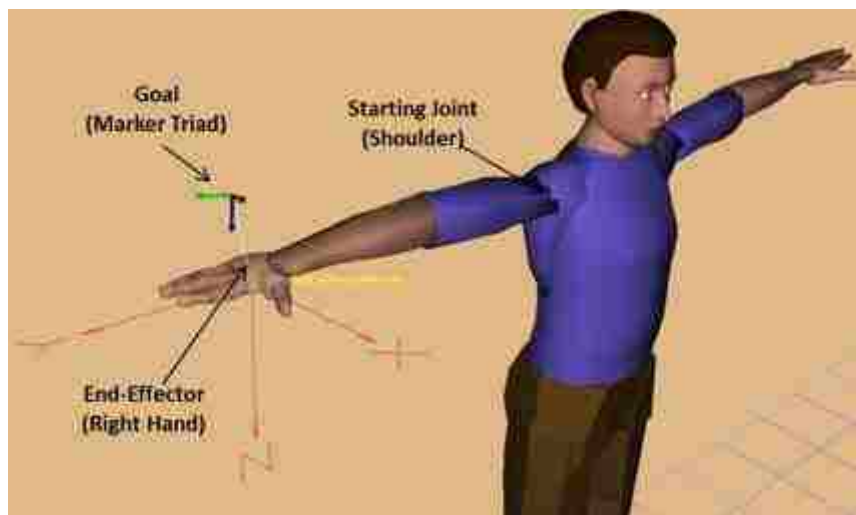


Figure 4.7. Constraints

Jack defines an inverse kinematic function called an objective or a potential function to solve constraints. This function measures the distance between the end-

effector and the goal. The constraint is satisfied when this objective function is minimized. This is achieved by calculating a set of all the joint angles which will posture the human model in such a way that the end-effectors will reach as close as possible to their respective goals. Hence, if there is a change in position, orientation or both of the goal frame, the constraint will be solved to minimize the difference in position, orientation or both between the coordinate frames of the goals and the end-effectors, by calculating a set of joints angles using the modeling and simulation constraints.

Let P denote an objective function associated with a goal, and e denote the location of the end-effector. This function is a weighted combination of position and orientation constraints. The potential function in case of a position constraint can be given by:

$$P(e) = (g - e)^2$$

where g is the goal point and e is the end-effector point in 3D space. An orientation constraint can be defined by the potential function given by:

$$P(x_e, y_e) = (x_g - x_e)^2 + (y_g - y_e)^2$$

where $\{x_g, y_g\}$ and $\{x_e, y_e\}$ are sets of two orthonormal vectors defining the orientation of the goal and the end-effector frames, respectively, in 3D space. Then, the constraint will be defined by an effective objective function, which is the weighted sum of the position and the orientation components. An individual constraint is satisfied when this function vanishes. However, a single constraint may not be sufficient to specify a pose. Hence, in addition to constraining the right-hand palm, the elbow joint is also desired to be constrained to its respective marker-triad. To pose the entire human model, Jack demands such constraints for multiple body joints. Then, the function associated with these

conjunctively combined goals is defined as a weighted sum of all the individual constraints given by the following equation:

$$G(\theta) = \sum_{i=1}^m w_i G_i(\theta)$$

where m is the number of goals combined, and

$$G_i(\theta) = P_i(e_i(\theta))$$

where subscript i refers to the i th goal, and w_i s are the weights on respective constraints G_i s. In case of full body motion capture systems, Jack demands the constraining to be applied to 20 fundamental body joints which will be discussed in Section 4. In the case of partial body motion capture, simulation constraints are applied only for a limited number of joints while the other joints are manipulated manually with the help of an interactive positioning provision in Jack.

Once the simulation environment in Jack is initialized, the *Simulate* function reads and plays back the motion capture data recorded by the motion capture system to obtain the simulation in Jack. Figure 4.8 shows the customized MoCap module developed in Jack using Jackscript.

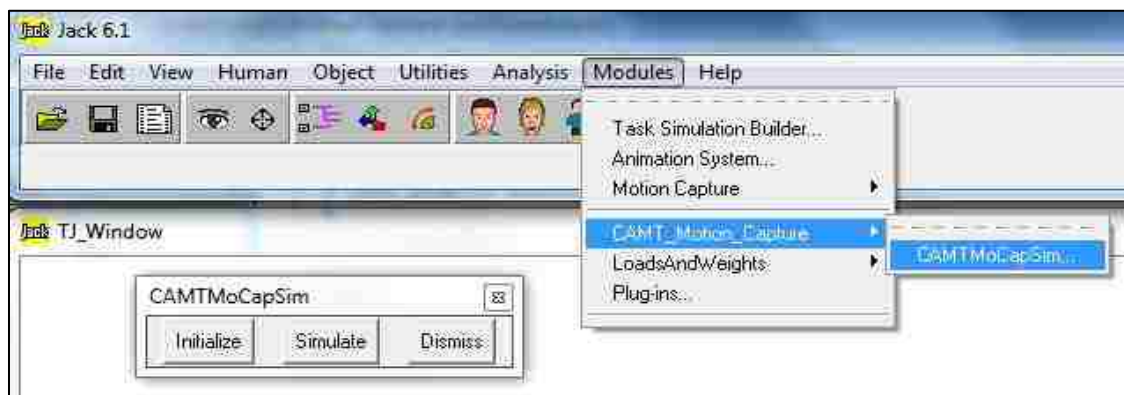


Figure 4.8. Interface in Jack

4.2. HUMAN MOTION SIMULATION USING KINECT

The Kinect for Windows SDK not only provides the 3D positions (in meters) of 20 body joints but also bone orientations (3x3 orientation matrices). This data can be used to obtain human motion capture simulations by using the Third Party Motion Capture Protocol (MoCap Protocol) as discussed previously. The development of such an application will be discussed in this section.

4.2.1. Interfacing Kinect and Jack. According to the protocol, the Third Party Motion Capture System (in this case Kinect) should create an application which can transfer the data, 3D position (X, Y, Z) in cm and (x, y, z) Euler orientations in radians of 20 triads corresponding to respective body joints in Jack by setting up a Transmission Control Protocol (TCP) socket connection. Figure 4.9 shows the 20 marker-triads to which the Jack skeleton needs to be fitted.

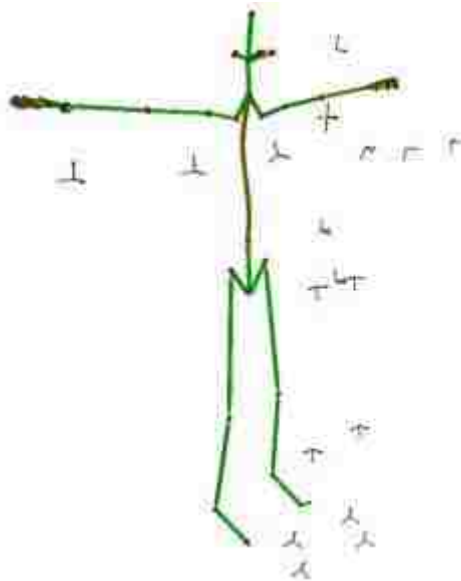


Figure 4.9. Structure of positions and orientations of marker triads and Jack skeleton before constraining

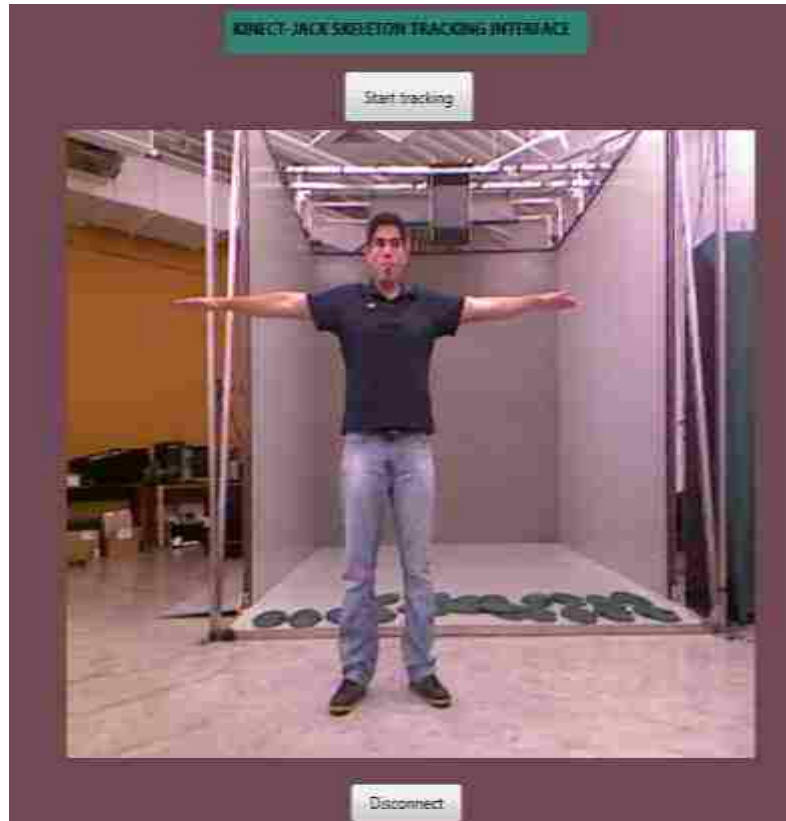
Once the Jack Mocap Server (JMS) is started, marker triads are created in Jack using this information. The Mocap interface has the capability of subject-specific auto-scaling of the default human model and constraining the marker triads to the respective body joints. Table 4.3 shows the joint data required for Jack’s Mocap Protocol and the joint data available with Kinect’s ST system.

Table 4.3. Kinect and Jack skeleton joint comparison

Kinect	Jack
Head	Head
Shoulder Center	Neck
<i>Not Available</i>	<i>Clavicle (R/L)</i>
Shoulder (R/L)	Upper Arm (R/L)
Elbow (R/L)	Lower Arm (R/L)
<u>Wrist (R/L)</u>	<u>Not Required</u>
Hand (R/L)	Hand (R/L)
Spine	Spine
Hip Center	Root
Hip (R/L)	Upper Leg (R/L)
Knee(R/L)	Lower Leg (R/L)
Ankle (R/L)	Foot (R/L)
Toe (R/L)	Toe (R/L)

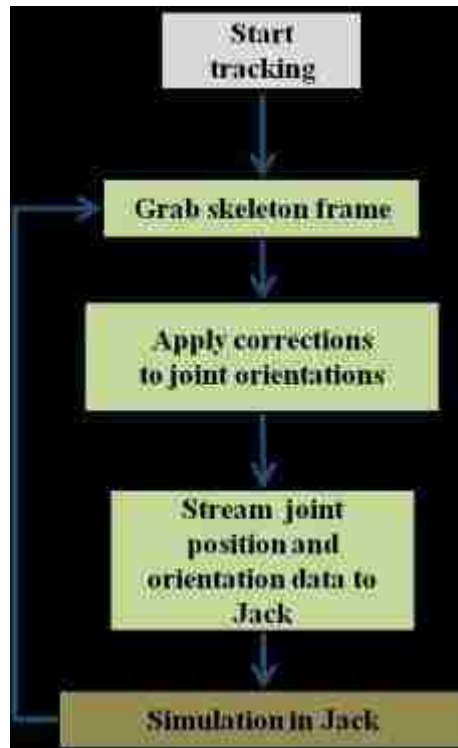
It was found that the majority of the joints required by the MoCap Protocol were provided by Kinect SDK. However, the right and the left clavicle joints (italic) were not available while the default orientations of the joints such as Spine, Hip Center, Ankle

(R/L) and Toe (R/L) (boxed) were not matching with the default orientations of the triads as required by the MoCap Protocol. An application was thus developed with the help of Kinect for Windows SDK (C#), whose User Interface (UI) is shown in the Figure 4.10 (a). Figure 4.10 (b) shows a flowchart explaining the functions involved in the application which are as follows: (1) Capture joint position and orientation data from the Kinect skeleton tracking stream, (2) Convert the bone orientation matrix into Euler angles (3) Apply corrections to default bone orientations from Kinect to map them to the Jack default orientations, and (4) Create a TCP connection and transfer the data to Jack.



(a)

Figure 4.10. Kinect-Jack skeleton tracking interface: (a) GUI and (b) data streaming process



(b)

Figure 4.10. Kinect-Jack skeleton tracking interface: (a) GUI and (b) data streaming process (cont)

The Natural User Interface (NUI) of Kinect for Windows SDK provides two types of data capture architectures viz., Polling based and Event based, for retrieving the skeletal data from the skeletal stream [41]. In Polling, the application manually requests a new frame from the skeletal stream and specifies a timeout value (in milliseconds). The method attempts to retrieve a new frame of data from the sensor before the timeout expires. If the timeout expires, the method returns a null frame. For the Event based model, every time an event of frame capture (30 fps) is fired, a function is called to get the next frame from the skeletal stream. No timeout value is required for this architecture. Though Polling is complex, performance-wise it is a more effective process as compared to the Event based model and is therefore used for our application. Once the skeleton

joint position and orientation data is obtained, the bone orientation matrix is converted into Euler angles and corrections are applied as discussed. This data is then encoded into a *registration message*. The application which acts as a client then sends this registration message through the TCP socket to JMS. Representative figures (triads) for each registered body joint are created in the Jack scene along with the ORIGIN figure (triad). Once the marker triads are registered, the next set of data is encoded and then sent with the help of a *data message*.

4.2.2. Mapping Jack and Kinect Coordinates Systems. The application developed above was used to track the movements of a person performing fastening operation on a mockup of the belly section of an aircraft fuselage. The first step was to map the coordinate system of the virtual world in Jack with the real-world coordinate system. To achieve this, a CAD model of a fuselage mockup was imported to Jack and scaled to the actual dimensions.

Three sites P_0 , P_1 , P_2 were selected on the base of the fuselage as shown in Figure 4.11 and their horizontal distances from the points O_1 and O_2 were manually measured. Points O_1 and O_2 were obtained by projecting end-points of the Kinect sensor box on the ground plane, whose Z coordinate is 0 cm as shown in Figure 4.12. These distances were used to calculate the 3D positions of the selected points. For example the 3D position of the points P_0 was found out using the distances D_1 and D_2 . The position of the fuselage model was then adjusted with the help of the calculated 3D points. Also, to ensure that the coordinate systems were mapped, the subject being tracked posed in certain postures and touched key points on the fuselage which was then compared in real time to their simulations in Jack as shown in the Figure 4.13.

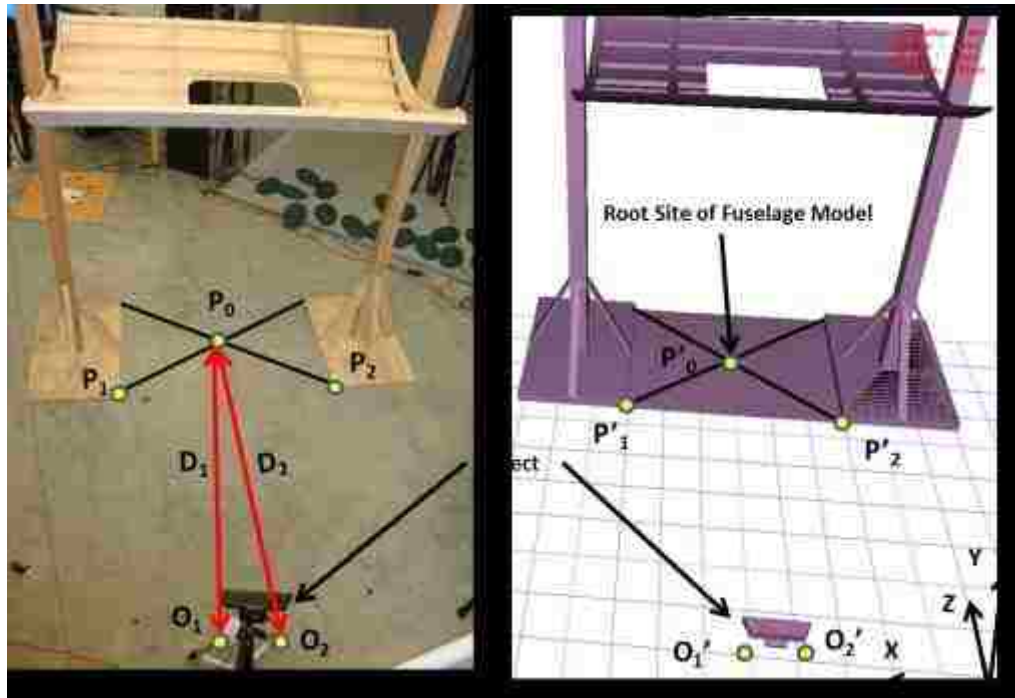


Figure 4.11. Mapping of real and virtual worlds, Left: real world scenario. Right: virtual world in Jack

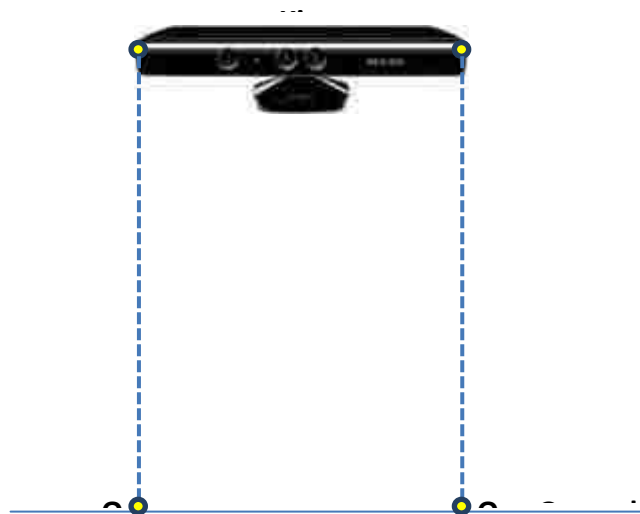


Figure 4.12. Definition of reference points

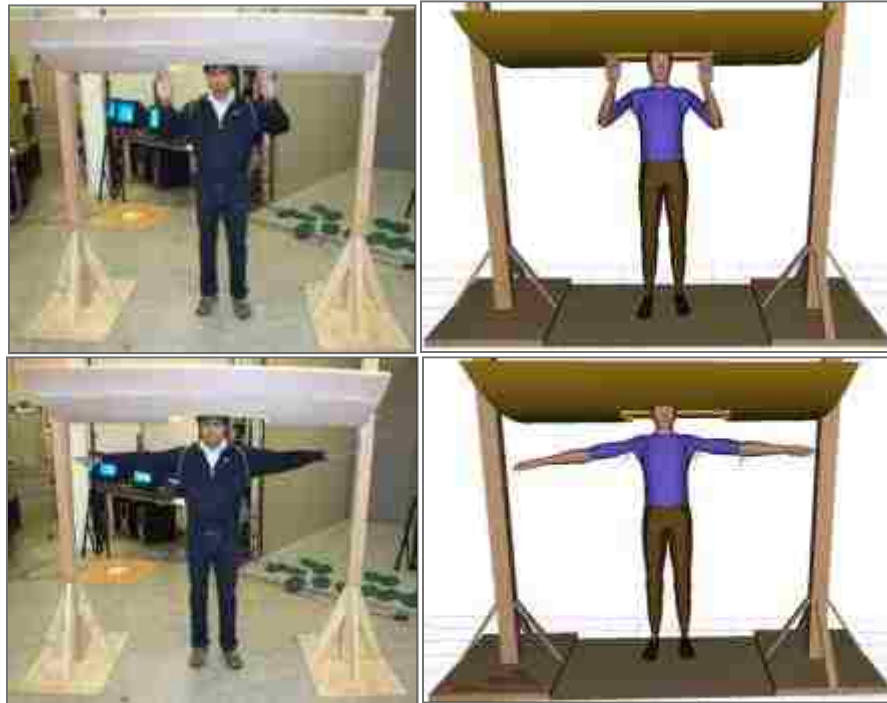


Figure 4.13. Evaluation of mapping, Left: real world poses, Right: virtual world poses in Jack

4.2.3. Simulation Using Single Kinect. Once the real and virtual worlds are mapped the person was asked to perform fastening operation on the mockup which was captured using a single Kinect as shown in Figure 4.14, the left column of the figure shows the snapshots of a sequence of person's movements tracked by the Kinect sensor while performing fastening operation on the mockup, and the right column shows the snapshots of the real-time simulation in the Jack.

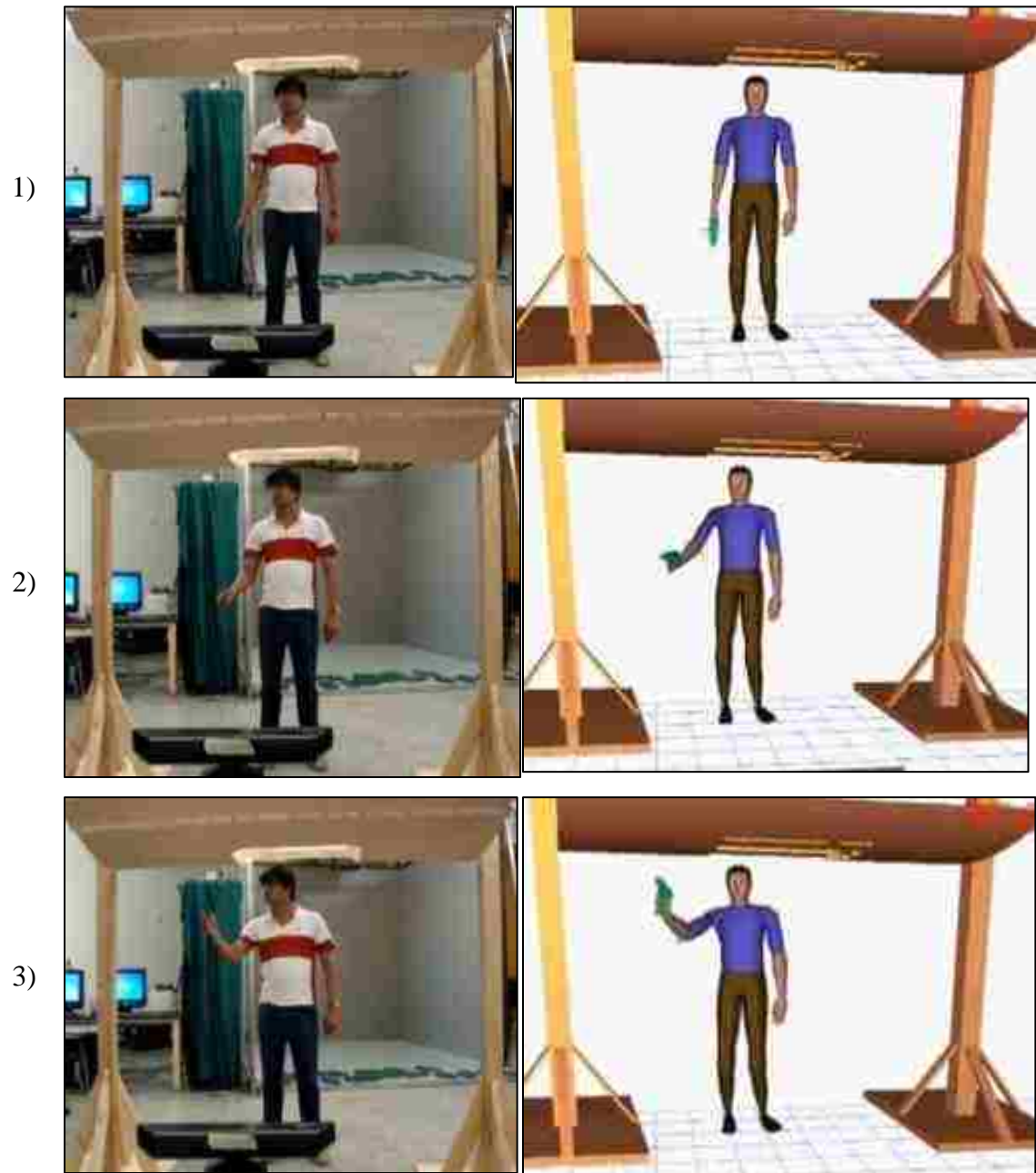


Figure 4.14. Simulation of fastening operation using a Kinect sensor, column 1: sequence of snapshots showing a person performing fastening operation, and column2: the corresponding simulation snapshots in Jack

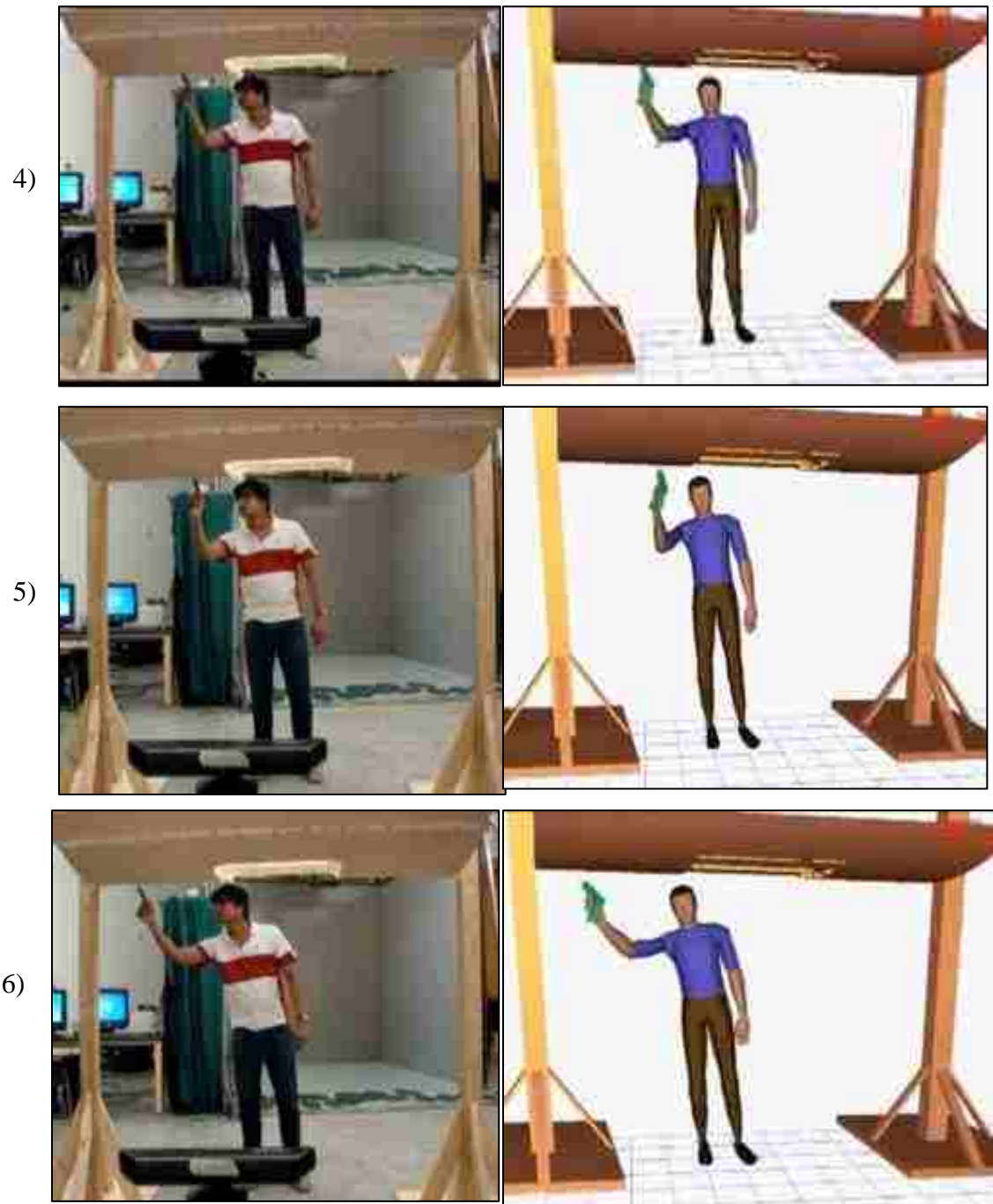


Figure 4.14. Simulation of fastening operation using a Kinect sensor, column 1: sequence of snapshots showing a person performing fastening operation, and column2: the corresponding simulation snapshots in Jack (cont)

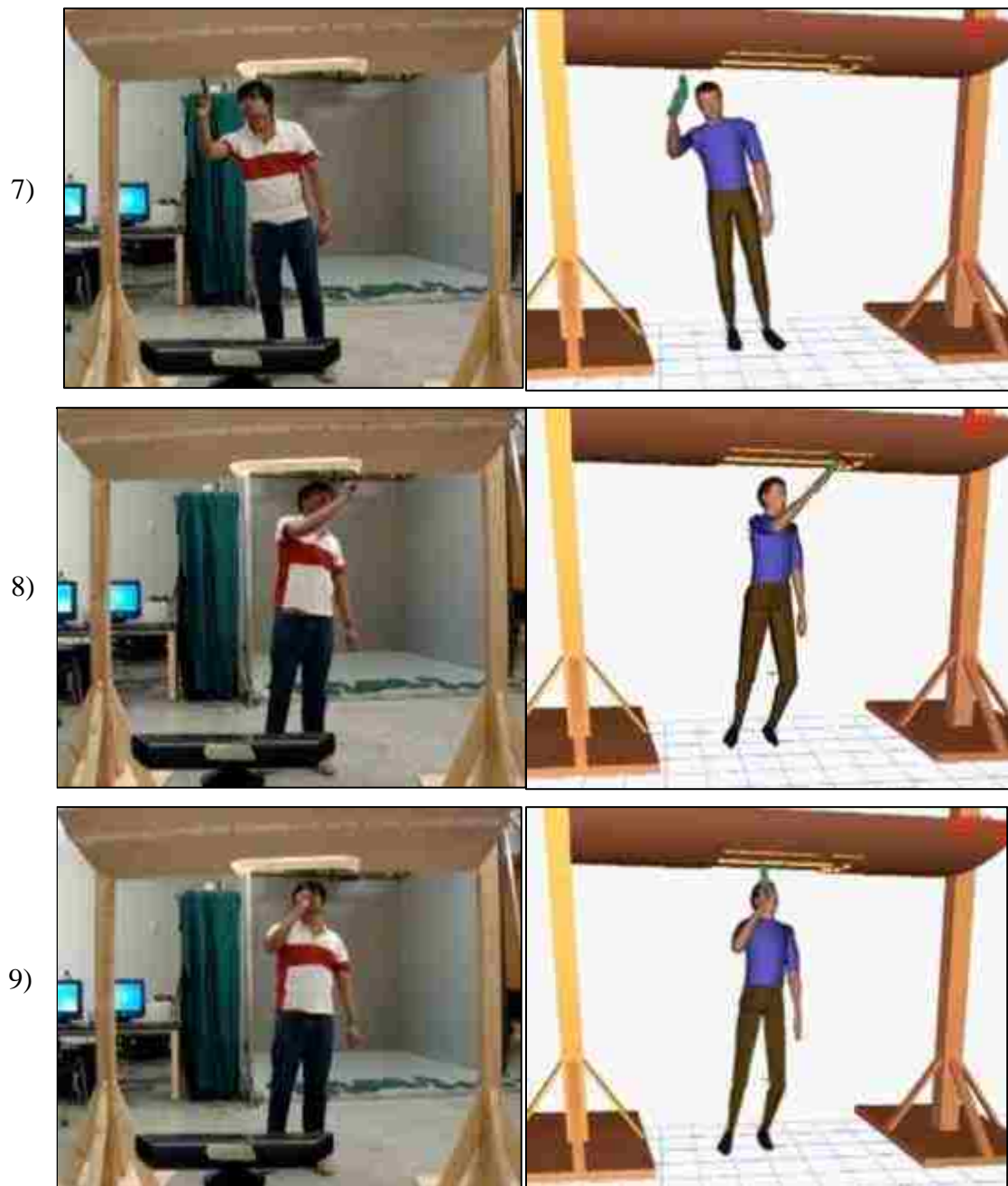


Figure 4.14. Simulation of fastening operation using a Kinect sensor, column 1: sequence of snapshots showing a person performing fastening operation, and column2: the corresponding simulation snapshots in Jack (cont)

10)



Figure 4.14. Simulation of fastening operation using a Kinect sensor, column 1: sequence of snapshots showing a person performing fastening operation, and column2: the corresponding simulation snapshots in Jack (cont)

In practice, there is signal noise present in the Skeletal Tracking (ST) system, and noise removal is a necessary step before using ST data. This can be managed by use of a noise reduction filter. Mehran [30] provided an overview of different filtering techniques and best practices for using the skeleton data for Kinect-enabled applications. It stated that the captured joint positions were accurate within one centimeter. The noise associated with joint position data provided by the ST system is dependent upon numerous factors such as room lighting, a subject's body size, the subject's distance from the Kinect sensor, the subject's pose, and location of the sensor. To address the noise issue, the paper suggested using a number of low pass filters such as a jitter removal filter to remove signal spikes, smoothing filters and a forecasting filter to reduce latency. The Kinect for Windows SDK provides an application called Avateering [39] which makes use of a combination of these filters to handle the noise associated with the raw data from the skeletal tracking system. This function was used for our application to obtain smooth simulations. It first applies the jitter removal filter [30] which dampens the spikes in the

input data. The filter's output is the same as the input if the difference between the current input data and the previous filter output is less than a threshold. Otherwise a double exponential filter [30] is used to dampen the changes in the output. This filter takes care of smoothing the data and predicting the future filter outputs to reduce the latency.

An experiment was carried out to test the filter. A subject was asked to stand in front of a Kinect sensor in a T-pose and the raw skeleton data and the filter output were recorded for 350 frames each. A typical output graph for this filter is shown in Figure 4.15. It could be observed that the static noise present in the X-coordinate of the left-hand is handled with the help of this filter. Thus this filter can be effectively used for removing the noise in the skeletal tracking data.

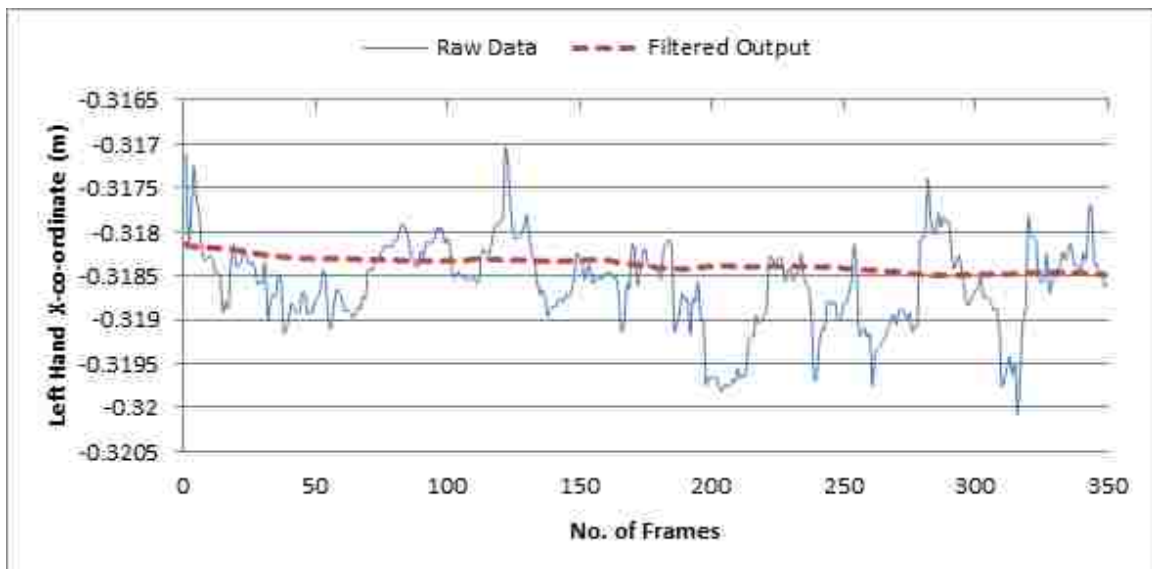
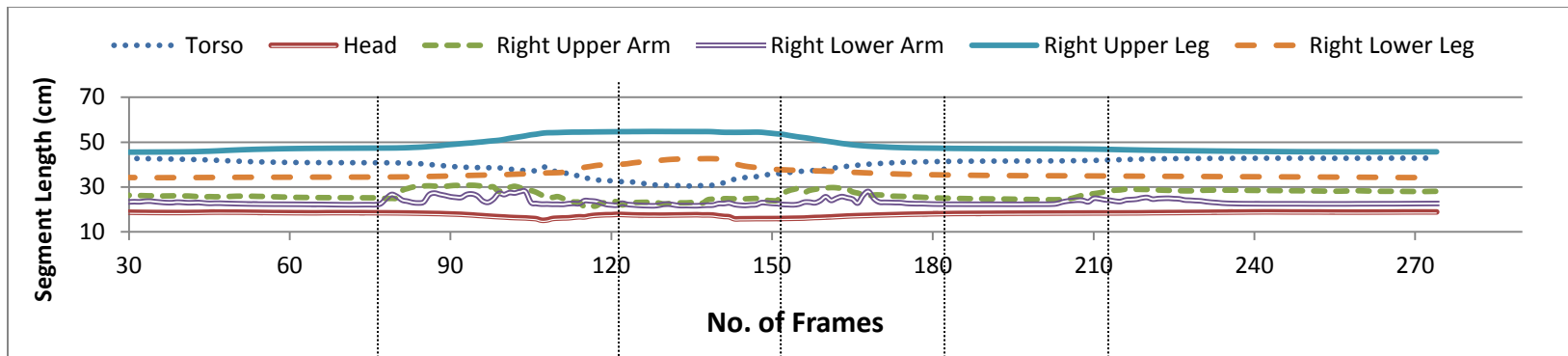


Figure 4.15. Filtering of skeletal tracking data

Apart from the static noise, it was observed from the simulation that the digital human model did not accurately reflect the actual postures. After carefully observing the motions of the triads in Jack during the simulation it was speculated that lengths of the segments might be changing for different postures resulting into incorrect fitting of the Jack model to the tracked data. Therefore, an experiment was carried out to evaluate the stability of the segment lengths for random movements of an operator facing the Kinect sensor.

4.2.3.1. Evaluation of Kinect skeleton tracking data for dynamic motions. For this experiment three random sets of motions were performed by a person in front of a Kinect sensor. Segment lengths were calculated using the 3D coordinates returned by the Kinect ST system. Torso, Head, Right Upper Arm, Left Upper Arm, Right Lower Arm, Left Lower Arm, Right Upper Leg, and Left Upper Leg were used as test segments. Figure 4.16 (a) shows the plots of some of these segment lengths vs. the number of frames for the first set of motions. It can be observed from this graph that the segment lengths varied for the entire duration of the motions. Figure 4.16 (b), top row shows the snapshots of the sequence of movements, where the subject moved both the arms up and down in the coronal plane in a standing posture, and Figure 4.16 (b), bottom row shows the corresponding snapshots from the simulation in Jack. Note that the actual length (not shown in the figure) of each of the subject's body-segments, was obtained by taking a mean of each of the segment lengths calculated using Kinect data, for the first 30 frames of the motion for which the subject was requested to stand in a static posture. These were used for scaling the segment lengths of the digital human model in Jack.



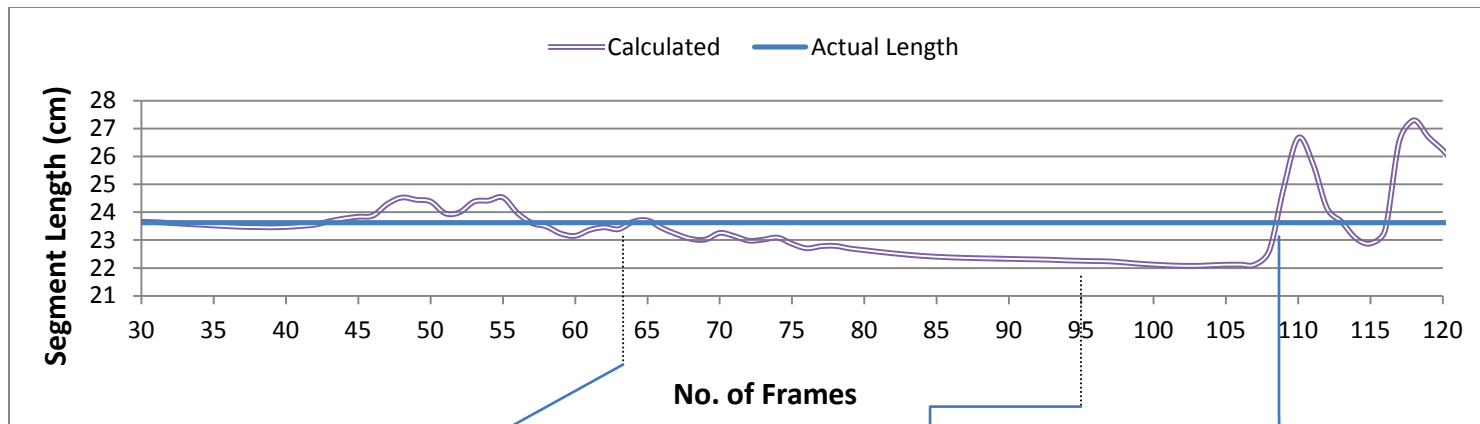
(a)



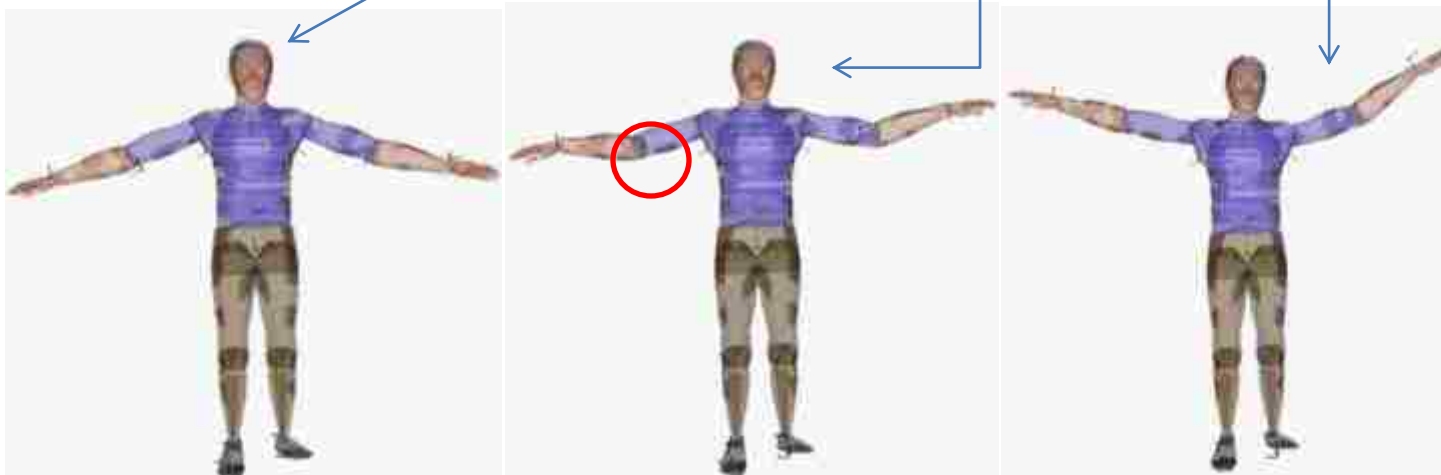
(b)

Figure 4.16. Dynamic motion analysis for first set of motions: (a) Graph showing variations in segment lengths over the duration of motion one (b) Row One: snapshots showing sequence of the actual motion and Row Two: corresponding snapshots of simulation in Jack

From the simulation it was observed that at certain instances, the human model does not behave similar to the actual actions performed by the subject. The cause of this misbehavior is the change in segment lengths with changing postures while performing dynamic motions. This may be due to the noise in the ST data; different postures may result in varying segment lengths [39]. The digital human model in Jack cannot be fitted properly to such changes in the segment lengths. This will be explained with the help of a sample from the simulation obtained using data captured for the first set of motions. Figure 4.17 (a) is a graph showing a plot of actual and the calculated segment lengths of the right lower-arm over a small duration of the simulation. It was observed that there was a decrease in the length of right lower arm which may be due to the noise in the 3D coordinates of the right-elbow joint, right-hand joint or both. As discussed earlier, the constraints in Jack will try to minimize the distance between the goal sites (marker triad) and the end-effector (joint) sites of the right-elbow joint and the right-hand joint. To satisfy the constraint, a decrease in the elbow joint angle was observed which can be seen in Figure 4.17 (b). It was observed that for the calculated right lower arm segment lengths, which were approximately equal to actual lengths, the elbow angle was observed to be similar to the actual values of the elbow angle. However, when the segment length decreased with respect to the actual length, a decrease in elbow angle was observed as compared to the actual angle.



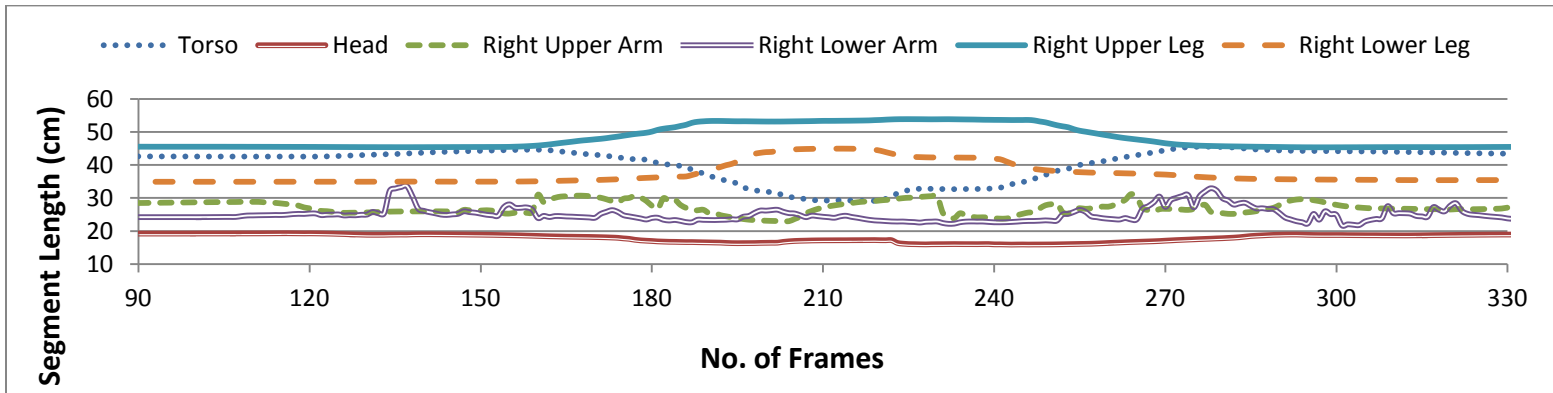
(a)



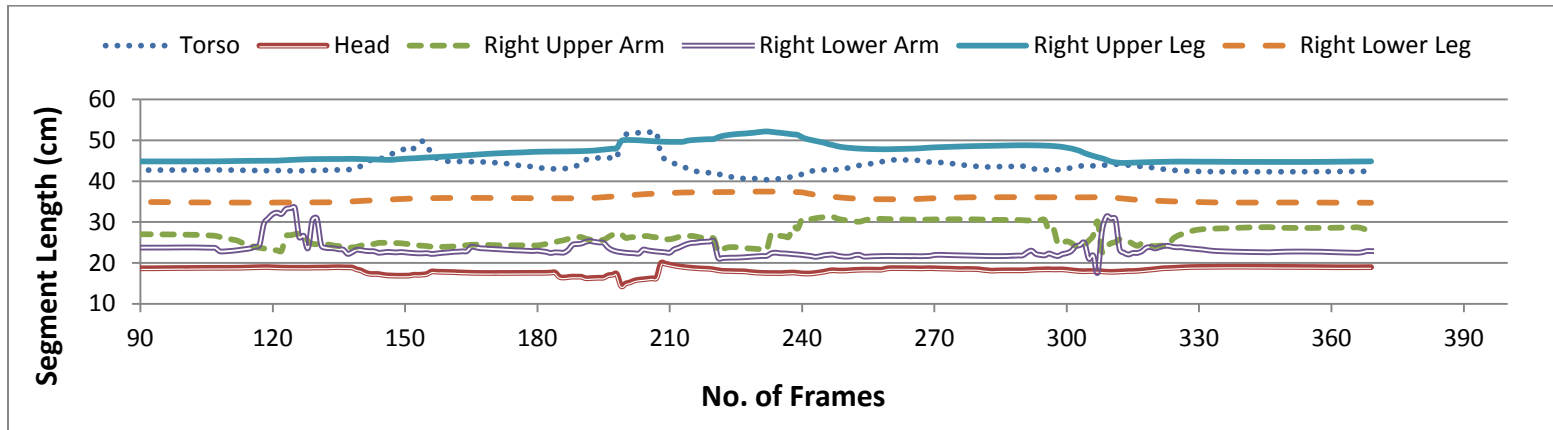
(b)

Figure 4.17. Dynamic motion analysis for a sample from first set of motions: (a) graph showing variations in right lower arm segment length over the duration of motion one (b) snapshots of simulation in Jack

Similar results were observed for the other two motions. The second motion was similar to the first motion except that the subject moved both arms in the sagittal plane (the plane parallel to the y-z plane of the Kinect coordinate system and intersecting the person facing the Kinect sensor). In the third motion the subject imitated an overhead fastening operation. It is evident from graphs shown in the Figures 4.18 (a) and (b) that the segment lengths varied over the duration of time of the motion which affected the simulations in Jack. Note that since the motions performed were symmetrical for the left and right sides, only the right side segments are shown in the graphs. The left side segments showed a similar trend as the right side segments.



(a)



(b)

Figure 4.18. Dynamic motion analysis for second and third set of motions: (a) graph showing variations in segment lengths over the duration of second set of motions (b) graph showing variations in segment lengths over the duration of third set of motions

4.2.3.2. Accuracy analysis of elbow angles for static postures. Another experiment was carried out to compare the accuracy of the right and left elbow angles calculated using the 3D positions of the shoulder, elbow and wrist joints available from Kinect. A mannequin was placed in the field of view of and facing (0°) the Kinect in five different postures as shown in Figures 4.19.

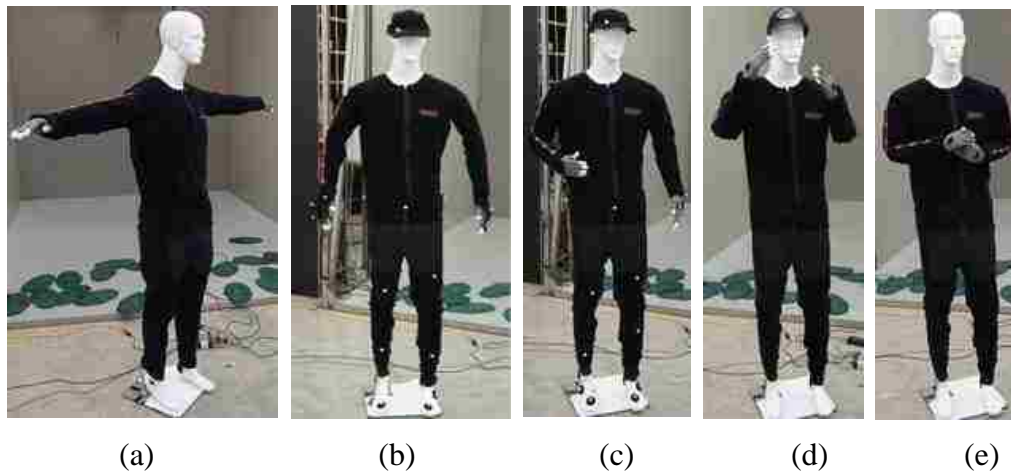


Figure 4.19. Mannequin postures for angle accuracy analysis: (a) T-Pose, (b) Standing Relaxed, (c) Right Forearm Up, (d) Working Posture A and (e) Working Posture B



Figure 4.20. Angle measurement set-up

Lego strips were placed on the right and left upper and lower arms as shown in the Figure 4.20, which also shows a 360° angle measurement plate. These strips were placed such that the elbow angles could be measured in a common plane of the upper and the lower arms of the mannequin. The elbow angles were calculated by using simple vector algebra where one vector was directed from elbow joint to the shoulder joint while the other was directed from the elbow joint to the wrist joint.

Three hundred frames of data were recorded for each posture. Table 4.4 and Table 4.5 show the absolute mean errors and the standard deviations of the right elbow and left elbow joint angles, respectively, for the five postures.

Table 4.4. Measured vs. actual, right elbow joint angle at different postures











Right Elbow Angle (Degrees)				
Mean of measured values (300 frames)	Actual	Error	Standard Deviation	Posture
166.13	180.00	13.87	0.23	
142.49	150.00	7.51	1.20	
110.11	105.00	5.11	2.23	
91.11	83.00	8.11	0.43	
92.04	92.00	0.04	1.09	

Table 4.5. Measured vs. actual, left elbow joint angle at different postures

Left Elbow Angle (Degrees)				
Mean of measured values (300 frames)	Actual	Error	Standard Deviation	Posture
171.77	180.00	8.23	0.22	
145.84	155.00	9.16	1.19	
144.95	150.00	5.05	2.23	
99.38	88.00	11.38	2.56	
80.17	88.00	7.83	0.21	

The mean error for the right elbow joint varied between 0° and 14° while for left elbow joint mean error varied between 5° and 12° . Similar results were observed in [38] where a comparison of shoulder and hip joint angles was carried between Kinect and a marker-based motion capture system which was more accurate but more expensive.

It can be concluded from the above experiment that the skeleton tracking system provides approximate predictions of 3D positions of the body joints. However, it will be shown in the next sections that using this data, reasonably good simulations can be

quickly obtained on a real-time basis which can prove to be a useful tool for ergonomic analysis.

4.3. SIMULATION RESULTS FOR MULTIPLE KINECTS

The switching algorithm was implemented to track the movements of an operator performing a fastening operation on the mockup of a fuselage in the laboratory environment with the use of two-Kinect system and a three Kinect system.

4.3.1. Simulation Results for Two-Kinect System. As shown in Figure 4.21, a mockup of the belly section of a fuselage was set up in the laboratory. Two Kinect devices were placed at 60° from each other with respect to the operator performing a fastening operation on the fuselage mockup.



Figure 4.21. Set-up for two-Kinect system

Figure 4.22 (a) shows the snapshot of an operator working on the mockup while Figure 4.22 (b) shows the snapshot of the corresponding posture of the digital human model in Jack. The simulation was obtained by using the raw data from the two Kinect devices which were calibrated as explained in Section 3.4.

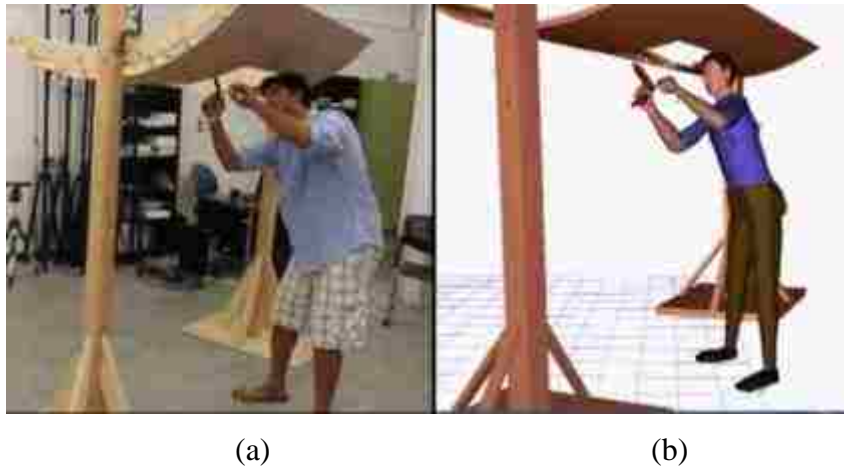


Figure 4.22. Motion simulation using a tracking system with two Kinects

The jittery characteristic of ST data was also observed in the simulation with two Kinects which indicates the need of filtering to stabilize the data.

4.3.2. Comparison of Simulation from Single and Two-Kinect Systems. In one experiment, a sequence of postures of an operator while performing a fastening operation on a fuselage mockup was captured, first with the use of a single Kinect, and then the same movements were tracked using a two-Kinect system. The purpose of this experiment was to show that the simulation obtained using data from the two-Kinect system has an obvious advantage over the simulations using data obtained from a single Kinect. In Figure 4.23, column one shows the operator performing the fastening

operation on the fuselage mockup which was captured using a single Kinect sensor, and column two shows the corresponding simulation in Jack. Column three shows the operator performing the fastening operation similar to those shown in column one on the fuselage mockup. These movements were captured using the two-Kinect system. Column four shows the corresponding simulation in Jack. It can be observed from Figure 4.23, column two that the simulation obtained using data from a single Kinect system started deteriorating from 10 to 17 frames which correspond to the operator angles $> 35^\circ$. This was mostly due to the self-occlusion of the right-hand side segments of the operator. Hence, keeping the value of α_{\max} as 34° , the two Kinects were placed at 60° (β) with respect to each other. The simulations obtained using data from the two-Kinect system was better than the simulation obtained using the data from the one-Kinect system. Note that the red circles in column two (single kinect sensor) of Figure 4.23 show the misbehavior of the segment of the digital human model in Jack, while the blue circles in column four (two-Kinect system) of Figure 4.23 shows that the misbehavior of the segments of the digital human model is addressed. Thus the issue of self-occlusion can be addressed by using a two-Kinect system over a single Kinect sensor.

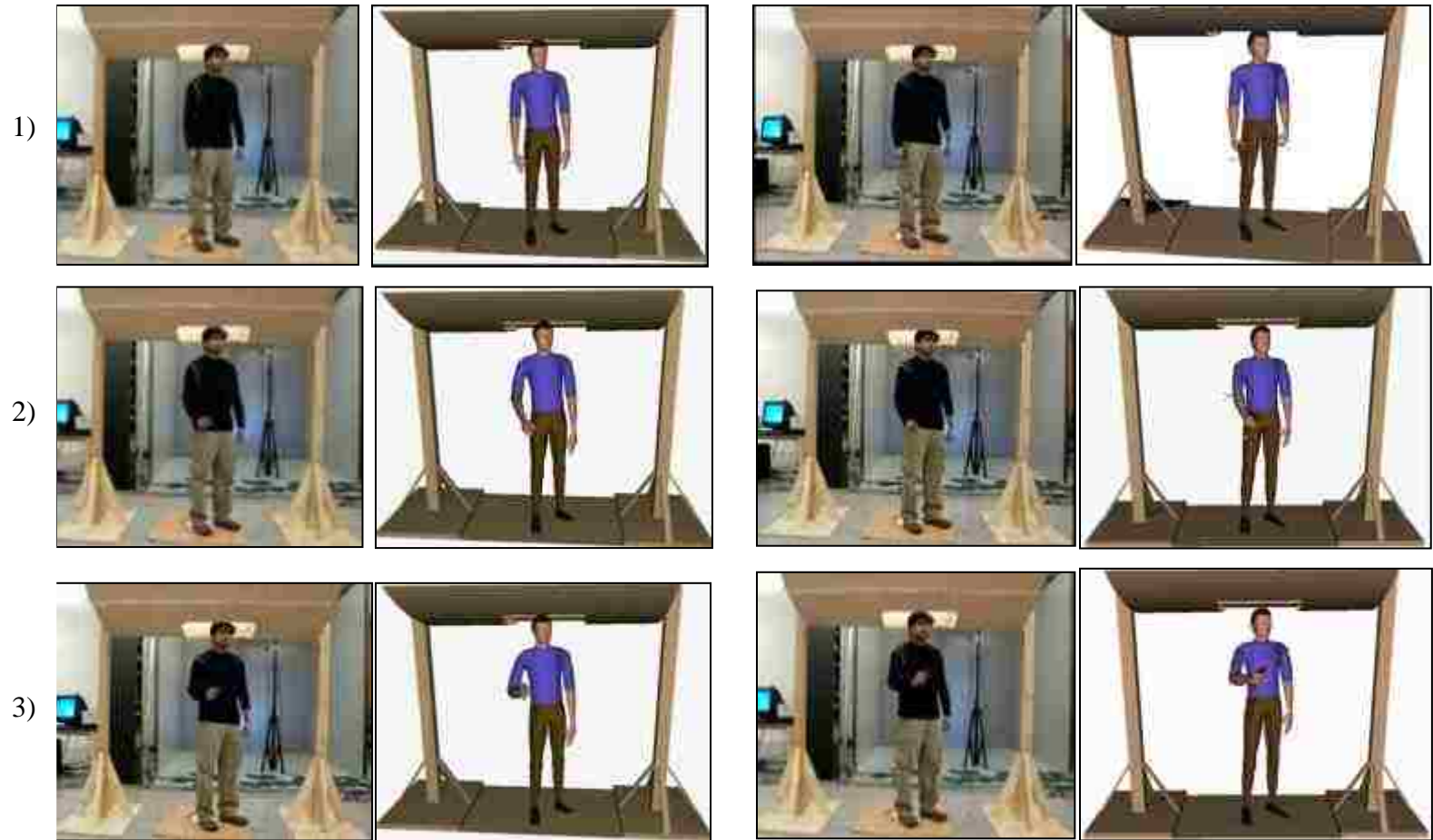


Figure 4.23. Comparison of simulation from one-Kinect and two-Kinect systems, column 1: fastening operation on fuselage mockup captured using a single Kinect, column 2: corresponding simulation with data obtained from one-Kinect system, column 3: fastening operation on fuselage mockup captured using a two- Kinect system, and column 4: simulation with data obtained from two-Kinect system for similar movements

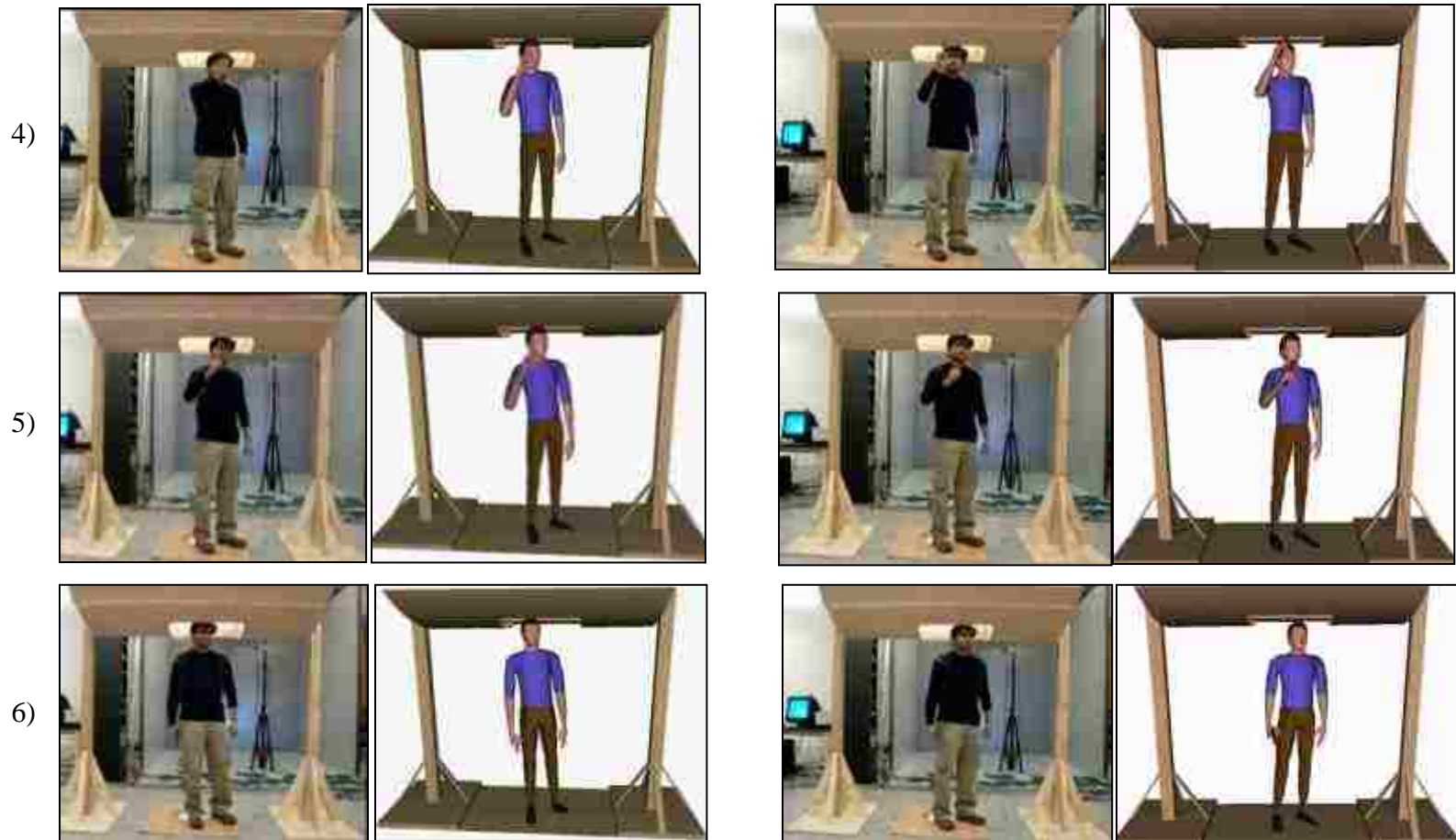


Figure 4.23. Comparison of simulation from one-Kinect and two-Kinect systems, column 1: fastening operation on fuselage mockup captured using a single Kinect, column 2: corresponding simulation with data obtained from one-Kinect system, column 3: fastening operation on fuselage mockup captured using a two- Kinect system, and column 4: simulation with data obtained from two-Kinect system for similar movements (cont)

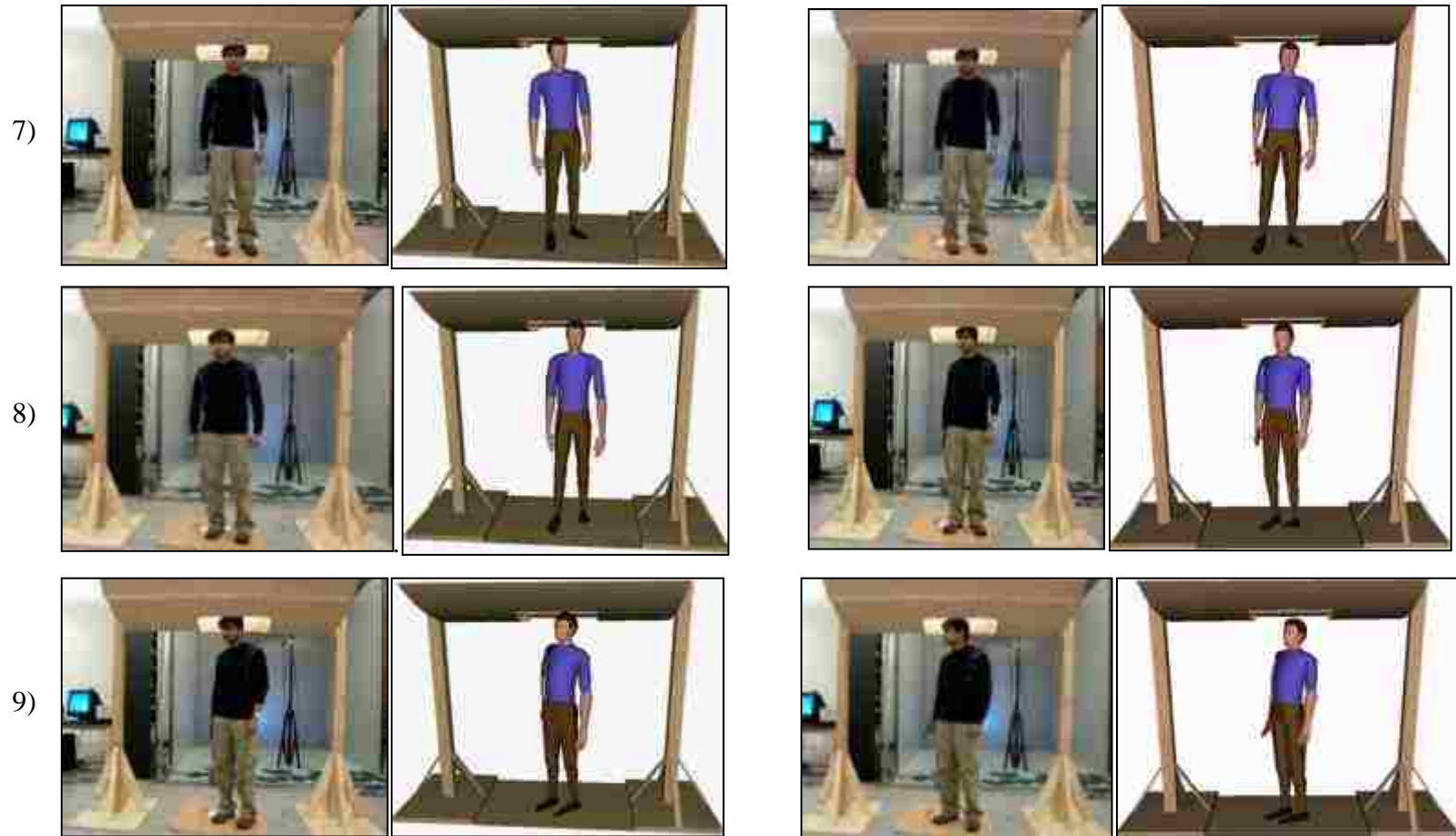


Figure 4.23. Comparison of simulation from one-Kinect and two-Kinect systems, column 1: fastening operation on fuselage mockup captured using a single Kinect, column 2: corresponding simulation with data obtained from one-Kinect system, column 3: fastening operation on fuselage mockup captured using a two- Kinect system, and column 4: simulation with data obtained from two-Kinect system for similar movements (cont)

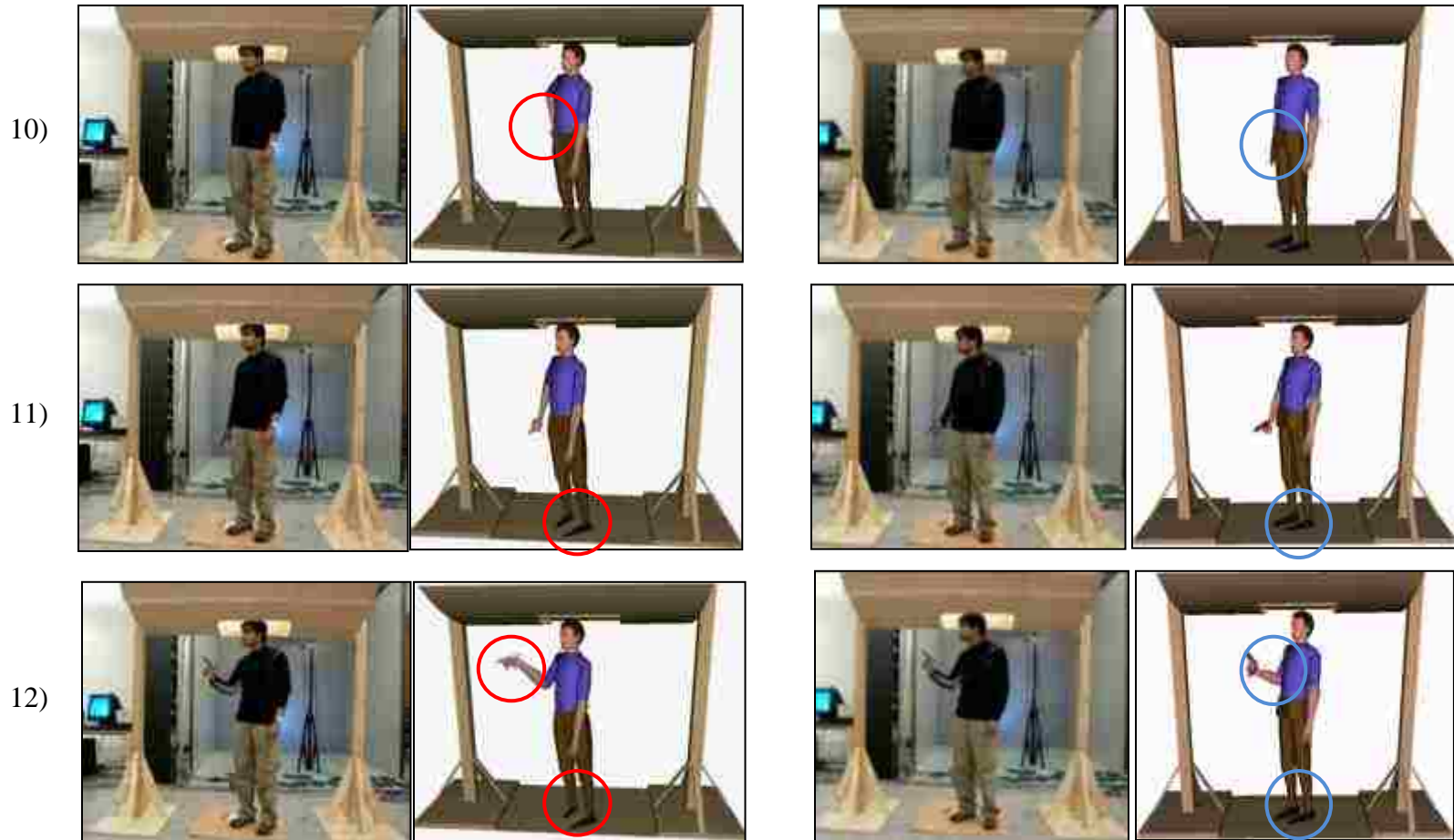


Figure 4.23. Comparison of simulation from one-Kinect and two-Kinect systems, column 1: fastening operation on fuselage mockup captured using a single Kinect, column 2: corresponding simulation with data obtained from one-Kinect system, column 3: fastening operation on fuselage mockup captured using a two- Kinect system, and column 4: simulation with data obtained from two-Kinect system for similar movements (cont)

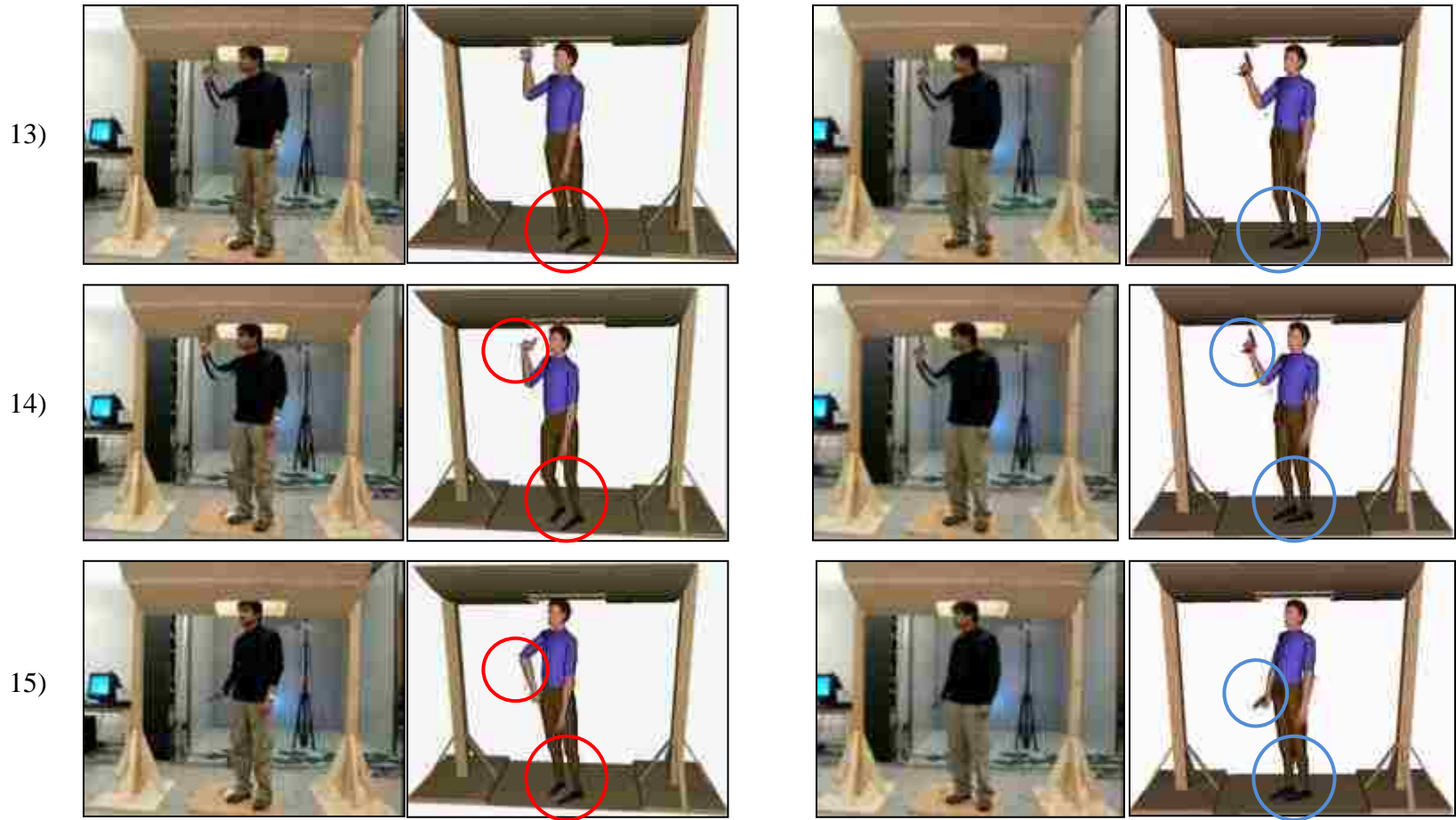


Figure 4.23. Comparison of simulation from one-Kinect and two-Kinect systems, column 1: fastening operation on fuselage mockup captured using a single Kinect, column 2: corresponding simulation with data obtained from one-Kinect system, column 3: fastening operation on fuselage mockup captured using a two- Kinect system, and column 4: simulation with data obtained from two-Kinect system for similar movements (cont)

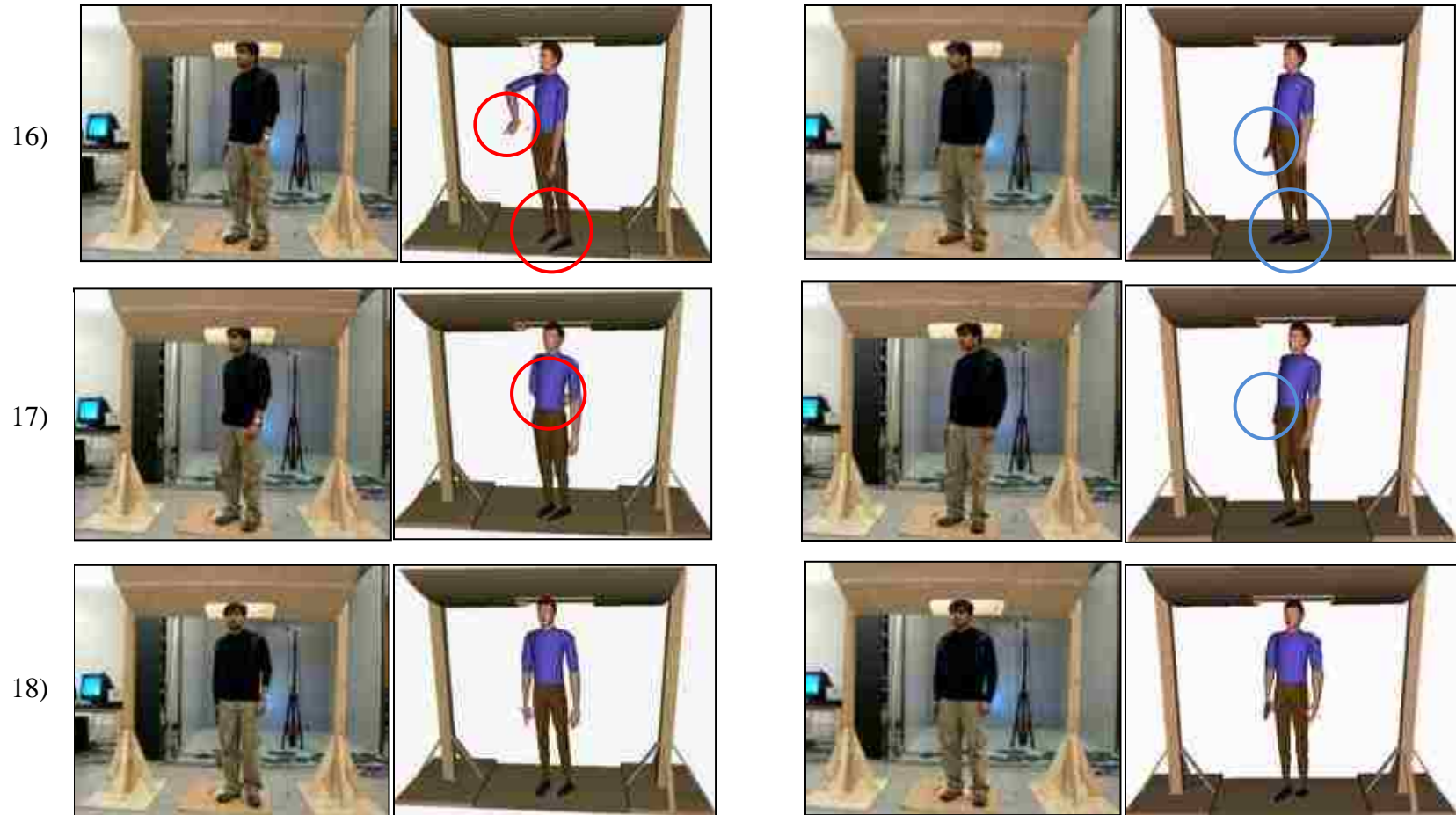
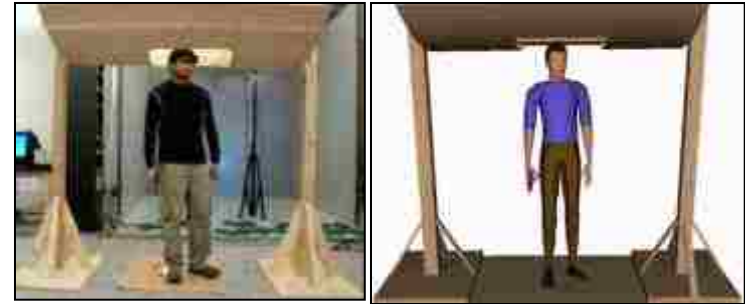
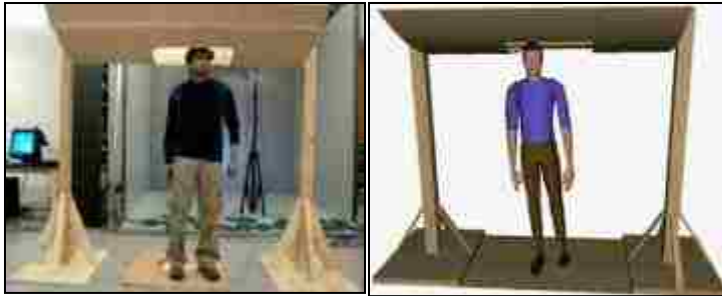


Figure 4.23. Comparison of simulation from one-Kinect and two-Kinect systems, column 1: fastening operation on fuselage mockup captured using a single Kinect, column 2: corresponding simulation with data obtained from one-Kinect system, column 3: fastening operation on fuselage mockup captured using a two- Kinect system, and column 4: simulation with data obtained from two-Kinect system for similar movements (cont)

19)



20)



Figure 4.23. Comparison of simulation from one-Kinect and two-Kinect systems, column 1: fastening operation on fuselage mockup captured using a single Kinect, column 2: corresponding simulation with data obtained from one-Kinect system, column 3: fastening operation on fuselage mockup captured using a two- Kinect system, and column 4: simulation with data obtained from two-Kinect system for similar movements (cont)

4.3.3. Simulation Using Three-Kinect System. A three-Kinect system was also set up in the laboratory (Please refer to Figure 3.21) to capture the movements of an operator performing a fastening operation on the fuselage mockup. The angles β_1 and β_2 were set to 30° while the angles α_1 and α_2 were set to 19° . Figure 4.24 shows a number of snapshots in the sequence of the fastening operation, as tracked by Kinect sensors K_1 , K_2 and K_3 . Figure 4.25 shows the corresponding simulation which was obtained by combining data obtained from all three Kinect devices. Here, the coordinate system of K_2 was considered as the world coordinate system of the three-Kinect system. K_1 and K_3 were calibrated with respect to K_2 . In Figure 4.24, the red (filled) circle indicates the active state of a sensor while the blank circles denote the inactive state. Note that, active and inactive states correspond to the open and closed positions of the shutter, respectively.

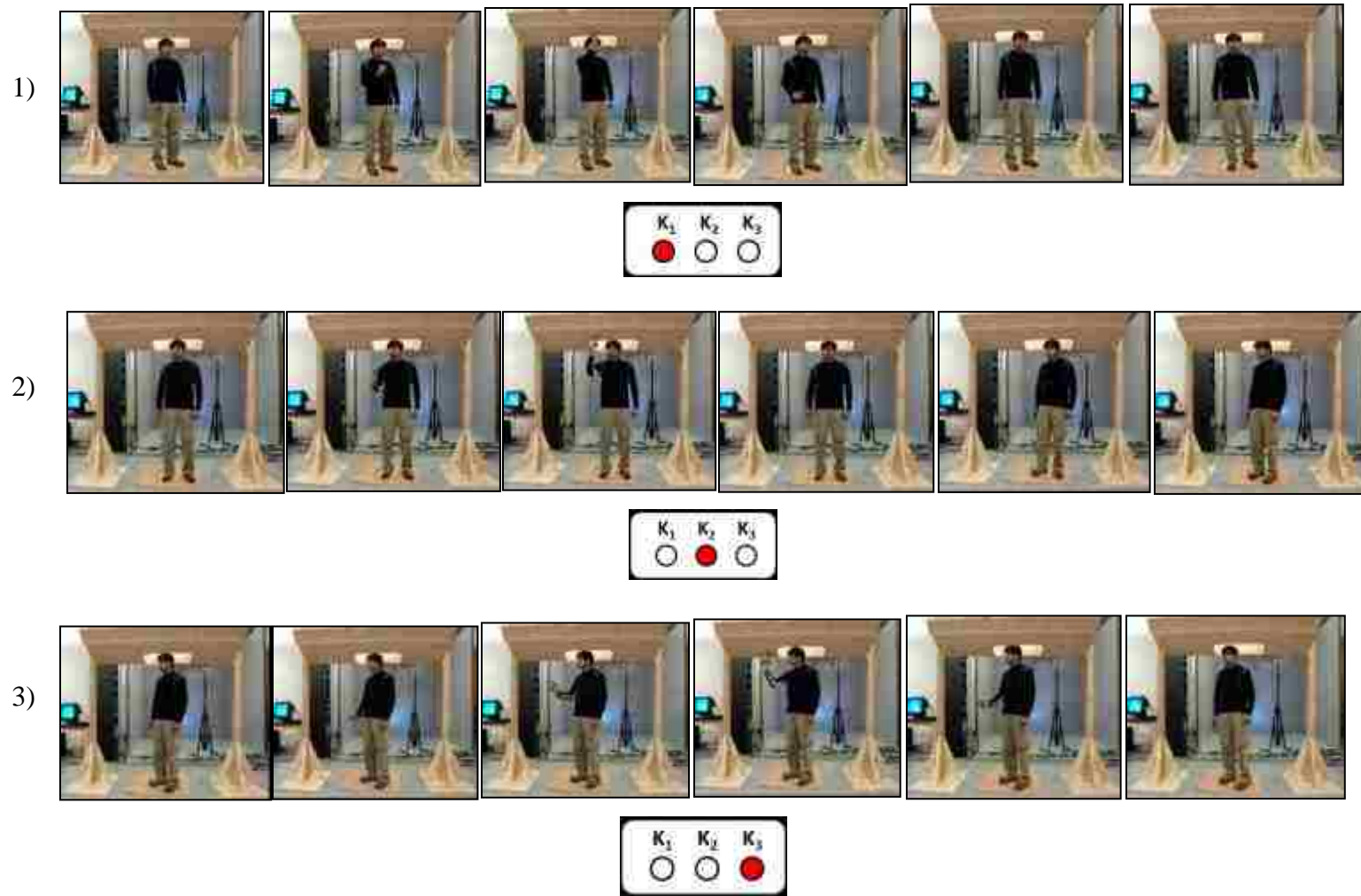


Figure 4.24. Sequence of fastening operation captured using a three-Kinect system, strips 1, 2 and 3: sequence captured by Kinects K_1 , K_2 and K_3 , respectively

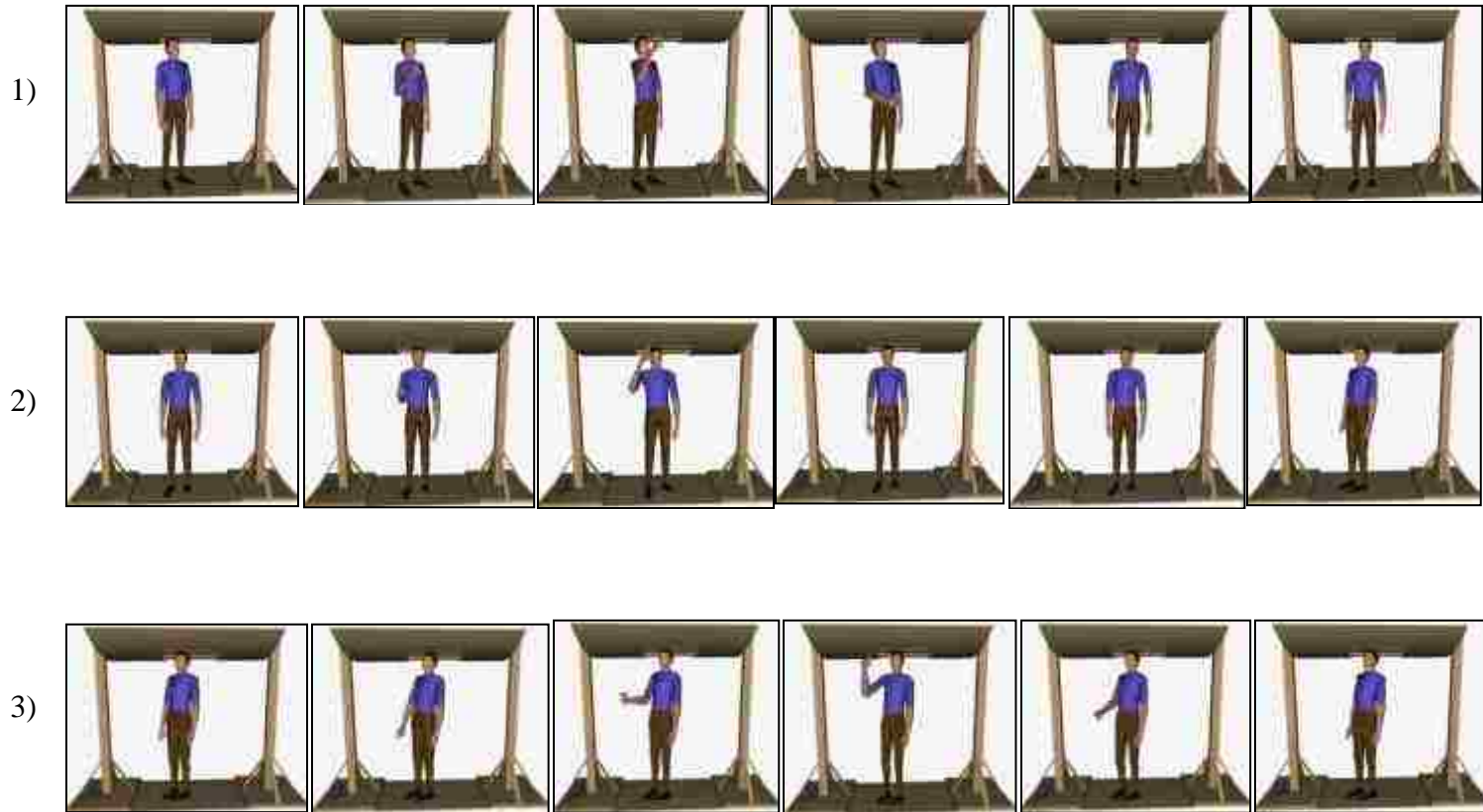


Figure 4.25. Simulation of fastening operation in Jack using data captured from three-Kinect system, corresponding to the sequence in Figure 4.24

4.4. ERGONOMIC ANALYSIS

With the operator's movements captured by the described motion capture systems, ergonomic analysis can then be performed in Jack to analyze the operator's posture during the fastening operation. Jack's Task Analysis Toolkit (TAT) is a set of human factor analysis tools that can be applied to the simulated human. Lower Back Analysis, Static Strength Prediction, NIOSH (National Institute for Occupational Safety and Health) Lifting Analysis, Fatigue analysis and RULA (Rapid Upper Limb Assessment) are some of the ergonomic analysis tools available from TAT. These tools can be run in the background of the simulation and hence can provide real-time updates of the analysis with the simulation.

RULA is a survey method that has been developed for use in ergonomic investigations of workplaces [42]. It is especially useful for scenarios in which work-related upper limb disorders are reported. The RULA analysis tool uses different operator postures and load conditions as input and provides an assessment of the upper body of the operator along with the risk level and suggestive action for that posture. RULA uses a scoring system based on posture, muscle use, and force to assign an action level to the evaluated task as shown in Table 4.6. For the fastening operation in the standing posture, the weight of the tool used was assumed to be less than 2 kg, and the force acting on the hand was assumed to be intermittent. After setting these parameters in RULA, the analysis was obtained. RULA calculates a grand score as a combined score of two different body groups: Group A (upper arm, lower arm and wrist) and Group B (neck and trunk) [42].

Table 4.6. Action levels in RULA

Level	Grand Score	Action
1	1 and 2	Acceptable posture if not maintained or repeated for long
2	3 and 4	Further investigation needed, may require changes
3	5 and 6	Investigation, changes required soon
4	7	Investigation, changes required immediately

The data obtained from the Kinect motion capture systems was used to obtain different postures for which RULA analysis has to be carried out. Figure 4.26 (a) shows an operator posture while performing fastening operation in front of a single Kinect system, while Figure 4.26 (b) shows the simulated posture in Jack along with the RULA *Analysis Summary* tab which can be explained in detail with the help of Figure 4.27. Thus, it could be concluded that for the posture shown in Figure 4.27, the grand RULA score was 4, which indicates the action level 2 suggesting that further investigation is needed for this posture and changes may be required.



Figure 4.26. RULA: (a) posture in real world and (b) simulation in Jack showing *Analysis Summary* tab of RULA tool

Rapid Upper Limb Assessment (RULA)

Task Entry | Reports | Analysis Summary

Job Title: Job Number:
 Location: Analyst:
 Comments: Date:

Body Group A Posture Rating

Upper arm: 4
 Lower arm: 2
 Wrist: 3
 Wrist Twist: 2
 Total: 5

Body Group B Posture Rating

Neck: 1
 Trunk: 2
 Total: 2

Muscle Use: Normal, no extreme use
 Force/Load: < 2 kg intermittent load
 Arms: Not supported

Muscle Use: Normal, no extreme use
 Force/Load: < 2 kg intermittent load

Legs and Feet Rating

Standing, weight even. Room for weight changes:

Grand Score: 4

Action: Further investigation needed. Changes may be required.

Update Analysis

Usage Dismiss

Figure 4.27. RULA: analysis summary

5. CONCLUSION

The study in this thesis demonstrates the capability of using low-cost, marker-less Kinect sensors as an effective tool to develop applications involving human motion simulation and ergonomic analysis.

A novel method was introduced to develop a multiple-Kinect system for tracking human movements by resolving the main issue of interference between multiple infrared structured light patterns. This ensures that the frame rate of the data collection is not reduced, which is also a major contribution of this research.

An interface was developed to transfer the skeletal tracking data obtained from a Kinect system on real-time basis to the commercial digital human modeling software Jack to obtain simulations. These simulations can be used to perform ergonomic analysis of different postures while performing various assembly tasks. A digital human Auto-Scaling application was also developed using Kinect for Windows SDK and Jack as an assist tool for developing simulations involving a limited number of human body parts.

Single Kinect, two-Kinect and three-Kinect systems were used to track the movements of an operator performing fastening operation on a fuselage mockup. Simulations were obtained in Jack and compared to demonstrate the advantage of a multiple-Kinect system over a single Kinect system in this scenario. Experiments were carried out to evaluate the data from the ST system. It was observed that the data requires filtering to obtain smoother simulations. Though the Kinect skeletal tracking system might not be as accurate as the commercial marker-based techniques, with further improvements in the accuracy of the ST system, it might prove to be an attractive and

low-cost alternative for developing applications in various fields. However, with the current technology, the data is sufficient to obtain quick simulations on a real-time basis which makes Kinect an effective tool, having an advantage over the time-consuming key frame method to obtain such simulations.

BIBLIOGRAPHY

- [1] <http://ilocis.org/documents/chpt90e.htm>, “ILO Encyclopedia of Occupational health and Safety,” *Aerospace Manufacture and Maintenance*, 90, 2012.
- [2] Mündermann, L., Corazza, S., Chaudhari, A.M., Andriacchi, T. P., Sundaresan, Chellappa, R., “Measuring human movement for biomechanical applications using marker-less motion capture,” *Proceedings of SPIE* , 6056, pp. 246-255, 2006.
- [3] <http://www.naturalpoint.com/optitrack>, Natural Point, 2012.
- [4] <http://www.vicon.com>, Vicon, 2012.
- [5] <http://www.motionanalysis.com>, Motion Analysis Corp, 2012.
- [6] <http://www.phasespace.com>, PhaseSpace, 2012.
- [7] Reed, M. P., Faraway, J., Chaffin, D. B. and Martin, B. J., “The HUMOSIM Ergonomics Framework: A New Approach to Digital Human Simulation for Ergonomic Analysis,” *Digital Human Modeling for Design and Engineering Conference Lyon, France*, 2006.
- [8] Duffy, V. G. & Li, Z., “Digital Human Modeling Packages,” *Handbook of digital human modeling*, 54, pp.1-20, 2010.
- [9] Jie, G., Guiling, C., Lin, H. & Dong, Z., “Application Research on Motion Capture System Data Reuse in Virtual Reality Environment,” *Second International Conference on Intelligent Human-Machine Systems and Cybernetics*, 2010.
- [10] Zhu, W., Daphalapurkar, C. P., Puthenveetil, S. C., Leu, M. C., Liu, X. F., Chang A. M., Gilpin-Mcminn, J. K., Wu, P. H. & Snodgrass, S., “Motion capture of fastening operation using Wiimotes for ergonomic analysis,” *Proceedings of the ASME 2012 International Symposium on Flexible Automation ISFA*, 2012.
- [11] Dua, J. & Duffy, V. G., “A methodology for assessing industrial workstations using optical motion capture integrated with digital human models,” *Occupational Ergonomics*, 7, pp. 11–25, 2007.
- [12] <http://www.xbox.com/en-US/KINECT>. Microsoft Kinect Xbox, 2012.
- [13] Lindstrom, K., “Current applications of Microsoft[®] Kinect[™] for non-commercial development projects”, 2012.
- [14] Ray, S. J. and Teizer, J., “Real-Time Posture Analysis of Construction Workers for Ergonomics Training,” *Construction Research Congress*, 2012.

- [15] Martin, C. C., Burkert, D. C., Choi, K. R., Wiczorek, N. B., McGregor, P. M., Herrmann, R. A. & Beling, P. A., "A Real-time Ergonomic Monitoring System using the Microsoft Kinect," *Proceedings of the 2012 IEEE Systems and Information Engineering Design Symposium, University of Virginia, Charlottesville, VA, USA*, April 27, 2012.
- [16] Ray, S. J. and Teizer, J., "Real-Time Posture Analysis of Construction Workers for Ergonomics Training," *Construction Research Congress*, 2012.
- [17] Wang, L., Hu, W. & Tan, T., "Recent developments in human motion analysis," *Pattern Recognition*, 36, pp. 585-601, 2003.
- [18] Thang, N. D., Kim, T-S., Lee, Y-K. and Lee, S., "Fast 3-D Human Motion Capturing from Stereo Data Using Gaussian Clusters," *International Conference on Control, Automation and Systems*, 2010.
- [19] Ganapathi, V., Plagemann, C., Koller, D. & Thrun, S., "Real Time Motion Capture Using a Single Time-Of-Flight Camera," *IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 755 – 762, 2010.
- [20] Fofi, D., Sliwa, T. & Voisin, Y., "A comparative survey on invisible structured light," *Proceedings of the SPIE*, 5303, pp. 90-98, 2004.
- [21] http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO_LIVE., Asus Xtion Pro Live, July 2012.
- [22] <http://www.primesense.com>. PrimeSense Ltd., July 2012.
- [23] Freedman, B., Shpunt, A., Machline M. & Arieli. Y., "Depth mapping using projected patterns," Patent Application, 05 2010. US2010/0118123 A1.
- [24] Zalevsky, Z., Shpunt, A., Maizels, A. & Garcia, J., "Method and system for object reconstruction," Patent Application. 03 2007, WO 2007/043036 A1.
- [25] Zalevsky, Z. & Shpunt, A., "Three dimensional sensing using speckled patterns," Patent Application. 09 2007. WO 2007/105205 A2.
- [26] <http://www.microsoft.com/en-us/kinectforwindows>. Microsoft Kinect for Windows Software Development Kit, July 2012.
- [27] <http://openni.org>. OpenNI, July 2012.
- [28] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A & Blake, A. "A, Real-Time Human Pose Recognition in Parts from Single Depth Images," *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2011.

- [29] <http://msdn.microsoft.com/en-us/library/hh855347>, Kinect for Windows SDK MSDN Library, 2012.
- [30] Mehran, A., <http://msdn.microsoft.com/en-us/library/jj131429>, “Skeletal Joint Smoothing White Paper,” 2012.
- [31] Berger, K., Ruhl, K. Schroeder, Y., Bruemmer, C., Scholz, A. & Magnor, M., “Markerless Motion Capture using multiple Color-Depth Sensors,” *Vision, Modeling, and Visualization*, pp. 317-324, 2011.
- [32] Maimone, A. & Fuchs, H., “Reducing Interference Between Multiple Structured Light Depth Sensors Using Motion,” *Virtual Reality Workshops*, pp. 51-54, 2012.
- [33] <http://www.ni.com/mydaq>, NI myDAQ, 2012.
- [34] http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example5.html, MATLAB Camera Calibration Toolbox, 2012.
- [35] Gaonkar, R., Madhavan, V. & Zhao, W., “Virtual Environment for Assembly Operations with Improved Grasp Interaction,” *Proceedings of the 10th IJIE Conference*, pp. 245 – 254, 2005.
- [36] Badler, N., Phillips, C.B. & Webber, B. L., “Simulating Humans: Computer Graphics, Animation and Control,” Department of Computer and Information Science, University of Pennsylvania, 1999.
- [37] Castaneda, V., Navab, N., “Time-of-Flight and Kinect Imaging,” *Kinect Programming for Computer Vision*, Technical University of Munich, 2011.
- [38] Fernandez-Baena, A., Susin, A., Lligadas, X., “Biomechanical Validation of Upper-body and Lower-body Joint Movements of Kinect Motion Capture Data,” *Ramon Llull University*, Barcelona, Spain, 2012.
- [39] <http://msdn.microsoft.com/en-us/library/jj131041.aspx>, “Kinect for Windows SDK,” *Avateering C# sample*, 2012.
- [40] Jack User Manual Version 7.1, 2012.
- [41] Webb, J., Ashley, J., “Beginning Kinect Programming with Microsoft Kinect SDK,” 2012.
- [42] McAtamney, L. and Corlett, E. N., “RULA: a survey method for the investigation of world-related upper limb disorders,” *Applied Ergonomics*, 24(2), pp. 91-99, 1993.

VITA

Chinmay Prakash Daphalapurkar was born in 1986 in India. He received his primary and secondary education in Pune, India. After completing his junior college, Chinmay received his Bachelor's degree in Mechanical Engineering from Pune University and graduated first class with distinction in August of 2008. After moving to US he pursued his Masters degree in Mechanical Engineering at Missouri University of Science and Technology, Rolla, and received his degree in December 2012.