REAL-TIME FALL DETECTION FOR ELDERLY WITH OBSTRUCTIONS CONSIDERATION USING KINECT

AMIR HAZIM BIN AMIR HUSIN

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

REAL-TIME FALL DETECTION FOR ELDERLY WITH OBSTRUCTIONS CONSIDERATION USING KINECT

AMIR HAZIM BIN AMIR HUSIN

This report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Electronics Engineering Technology with Honours

Faculty of Electronics and Computer Technology and Engineering
Universiti Teknikal Malaysia Melaka

2025

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek : Real-Time Fall Detection For Elderly With Obstructions Consideration Using Kinect

Sesi Pengajian : 2024/2025

Saya AMIR HAZIM BIN AMIR HUSIN mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

☐	**SULIT***	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD***	(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan.)
☑	**TIDAK TERHAD**	

Disahkan oleh:

(TANDATANGAN PENULIS) (COP DAN TANDATANGAN PENYELIA)

Alamat Tetap: .....
.....
.....
.....
.....

Tarikh : 22 Januari 2025 Tarikh : 22 Januari 2025

*CATATAN: Jika laporan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

DECLARATION

I declare that this project report entitled “Real-Time Fall Detection For Elderly With Obstructions Consideration Using Kinect” is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature :

Student Name : AMIR HAZIM BIN AMIR HUSIN

Date : 22 JANUARI 2025

APPROVAL

I hereby declare that I have checked this project report and in my opinion, this project report is adequate in terms of scope and quality for the award of the degree of Bachelor of Electronics Engineering Technology with Honours.

Signature : .....

Supervisor Name : IZADORA BINTI MUSTAFFA

Date : 22 JANUARI 2025

Signature : .....

Co-Supervisor : .....

Name (if any)

Date : .....

DEDICATION

To my beloved mother, Norizan binti Dan,

and

To my precious father, Amir Husin bin Abd Manaff,

and

To dearest grandmother, Misah binti Abdul Rahman

and

To my treasured siblings, Amir Nazrin bin Amir Husin and Ain Farisha binti Salleh

and

To my closest friend, Aiman Aliff bin Amir Ariffin

ABSTRACT

The increased number of elderly people is growing in tandem with improvements in healthcare. However, the risks that go along with it have also increased, including the number of falls. According to several studies, elderly persons fall at least once a year. For elderly persons aged 65 and older, falls are the leading cause of unintentional death. Many elderly people in Malaysia spend the day alone at home because their family members are either at work or school. Therefore, to properly identify the occurrence of falls, a fall detection system is required. The construction of such a system to identify when an individual falls or loses their balance is described in this thesis. To evaluate and spot patterns that point to a fall incident, the system uses Kinect sensors and algorithms to track a person's movements and postures. Skeletons and joints are identified and retrieved, including the heads, shoulders, hips, and left and right ankles. The Y-coordinate values and threshold values are obtained by implementing the fall algorithm. To ascertain fall status, the absolute values of the Y-coordinate and joints are compared to the threshold value. The system also differentiates between falls, sitting on a chair, sitting on the floor, and also fall incident behind an obstruction. When a possible fall incident is detected, the system activates an alarm. The expected results include high accuracy in detecting falls, minimizing false positives, and ensuring that the system operates effectively even when there is partial body visibility. The system aims to provide timely alerts to caretakers or emergency responders, significantly reducing response times and potentially saving lives. Additionally, the solution is designed to be minimally intrusive, ensuring it does not disrupt the elderly's daily activities.

ABSTRAK

Peningkatan bilangan warga emas semakin meningkat seiring dengan peningkatan dalam penjagaan kesihatan. Walau bagaimanapun, risiko yang menyertainya juga telah meningkat, termasuk bilangan kejatuhan. Menurut beberapa kajian, orang tua jatuh sekurang-kurangnya sekali setahun. Bagi warga emas berumur 65 tahun ke atas, jatuh adalah punca utama kematian yang tidak disengajakan. Ramai warga emas di Malaysia menghabiskan hari bersendirian di rumah kerana ahli keluarga mereka sama ada di tempat kerja atau sekolah. Oleh itu, untuk mengenal pasti kejadian jatuh dengan betul, sistem pengesanan jatuh diperlukan. Pembinaan sistem sedemikian untuk mengenal pasti apabila seseorang individu itu jatuh atau hilang keseimbangan diterangkan dalam tesis ini. Untuk menilai dan mengesan corak yang menunjukkan kejadian jatuh, sistem menggunakan penderia dan algoritma Kinect untuk menjejaki pergerakan dan postur seseorang. Rangka dan sendi dikenal pasti dan diambil, termasuk kepala, bahu, pinggul, dan buku lali kiri dan kanan. Nilai koordinat Y dan nilai ambang diperoleh dengan melaksanakan algoritma kejatuhan. Untuk memastikan status jatuh, nilai mutlak koordinat-Y dan penyambung dibandingkan dengan nilai ambang. Sistem ini juga membezakan antara kejadian jatuh, duduk diatas kerusi, duduk diatas lantai, dan juga kejadian jatuh dibelakang sesuatu pengahalang. Apabila kemungkin kejadian jatuh dikesan, sistem ini menghantar penggera mengaktifkan sistem penggera. Hasil yang dijangkakan termasuk ketepatan yang tinggi dalam mengesan jatuh, meminimumkan positif palsu, dan memastikan sistem beroperasi dengan berkesan walaupun penglihatan badan adalah separa. Sistem ini bertujuan untuk memberikan makluman tepat pada masanya kepada penjaga atau responden kecemasan, dengan ketara mengurangkan masa tindak balas dan berpotensi menyelamatkan nyawa. Selain itu, penyelesaian itu direka bentuk untuk mengganggu secara minimum, memastikan ia tidak mengganggu aktiviti harian warga tua.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor, Puan Izadora binti Mustaffa and co-supervisors, Dr Haslinah binti Mohd Nasir and Puan Dayanasari binti Abdul Hadi, for their precious guidance, words of wisdom and patient throughout this project.

I am also indebted to Universiti Teknikal Malaysia Melaka (UTeM) for the financial support which enables me to accomplish the project. Not forgetting my fellow colleague, Aiman Aliff bin Amir Ariffin for the willingness of sharing his thoughts and ideas regarding the project.

My highest appreciation goes to my parents, siblings, and family members for their love and prayer during the period of my study. An honourable mention also goes to Dr Haslinah binti Mohd Nasir for all the motivation and understanding.

Finally, I would like to thank all the staffs at the faculty, fellow colleagues and classmates, the faculty members, as well as other individuals who are not listed here for being co-operative and helpful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

This chapter aims to establish the framework and presents a brief concept of the project. It focuses on the overview of the project, describes the objectives, briefly the problem, the scope, and the results of the project.

1.2 Background

The rising number of senior citizens and the associated fall risks have increased the importance of fall detection systems in recent years. Falls are the leading cause of unintentional death among individuals aged 65 and older, with studies indicating that at least one fall occurs annually in this demographic. This has driven significant research and innovation in fall detection systems [1]. Over the years, various techniques, tools, and sensors have been explored, including gyroscopes, accelerometers, GPS modules, and Kinect sensors. Among these, vision-based systems like Kinect have stood out due to their non-intrusive nature, eliminating the need for elderly individuals to wear specialized equipment [2].

The Microsoft Kinect sensor has become a notable tool for fall detection due to its ability to accurately track human body movements and recognize skeletal joints. Unlike wearable-based solutions, Kinect-based systems offer the convenience of remote monitoring without

physical contact, making them more user-friendly and comfortable for elderly users. Additionally, the Kinect sensor's capacity to capture both depth and colour information enhances the precision of fall detection algorithms [3].

The system is designed for indoor use, particularly in public hospitals and private homes. Past fall detection systems relied heavily on wearable sensors, which were often inconvenient for users. In contrast, Kinect-based systems integrate vision-based subsystems, utilizing libraries for camera management and computer vision techniques to process both depth and colour data. This approach has demonstrated impressive reliability (97.3%) and efficiency (80.0%) in detecting falls, making it a promising solution for real-world applications [4].

Moreover, the integration of Kinect sensors with emergency notification systems can provide immediate assistance upon detecting a fall. The Kinect's ability to collect and analyse data in real time allows for a rapid response, potentially reducing the severity of injuries caused by falls. Over time, the data collected can offer valuable insights into mobility patterns, supporting proactive care and long-term monitoring [2].

The Kinect sensor's unique qualities address key challenges highlighted in fall detection research, such as dealing with sensor imperfections and environmental interference. Its precise skeletal tracking minimizes false alarms and ensures consistent data, making it a reliable option for real-time fall detection. Furthermore, the Kinect's robust data collection capabilities position it as a valuable tool for developing standardized datasets and advanced algorithms, helping to advance the field of fall detection research [5].

1.3 Problem Statement

The detection rate's performance in real-world scenarios is one of the fall detection system's primary concerns. It has been demonstrated that elements including flooring surfaces, barriers, and lighting conditions may have an impact on fall detection systems' accuracy and promising outcomes. Usability and user acceptance are another difficulty. For users to adopt fall detection systems, they must be simple to use and not impede on normal activities. Since delays in detecting falls might result in serious injuries, real-time operations are also crucial for fall detection systems.

The imperfection of the data collected, which can be brought on by a few things such as sensor noise, calibration mistakes, and environmental interference, is another significant obstacle in fall detection. Another issue is the unreliability or diversity of sensor systems, which can result in false alarms and inconsistent data. Additionally, there are certain difficulties that are common to other frameworks with data fusion needs, like choosing the right sensors, creating efficient data fusion algorithms, and handling missing or insufficient data. Furthermore, there isn't a standardized dataset for assessing fall detection systems, which makes contrasting various strategies challenging. Investigating novel sensor technologies and creating increasingly complex data fusion algorithms are important.

1.4 Project Objective

The project aims to achieve the following objectives: -

1. To recreate different fall scenarios with object obstructions for the development of the Kinect algorithm.

2. To analyse the functionality of the developed system by determining the optimal location for the Kinect sensor in terms of distance, height, and area coverage.
3. To develop a fall detections system based on the skeletal tracking data collected by the Kinect sensor with alarm notification.

1.5 Scope of Project and Limitations

This project focuses on developing a Kinect-based fall detection system specifically for elderly individuals in indoor residential settings. Utilizing the Microsoft Kinect v1 sensor, the system leverages its depth-sensing and skeletal tracking capabilities to monitor body movements and detect falls within a single-room environment. Designed for single-sensor implementation, the system ensures non-intrusive monitoring without the need for wearable devices, prioritizing user comfort. The system integrates predefined fall detection algorithms to provide real-time alerts to caregivers or emergency responders, enhancing safety for elderly individuals living alone.

To maintain efficiency, the system is optimized to detect up to two elderly individuals falling simultaneously, allowing for effective tracking while minimizing computational complexity. However, since it relies on a single Kinect v1 sensor, it is not designed to monitor multiple rooms or overcome significant obstructions, requiring careful placement to minimize blind spots. The system's effectiveness is best suited for controlled indoor environments, where external factors such as furniture arrangement and lighting conditions can be managed to optimize detection accuracy.

Despite these constraints, the project establishes a functional prototype that serves as a foundation for future improvements. Enhancing detection accuracy, expanding coverage

capabilities, and refining fall classification methods remain key areas for further research and development. By focusing on a specific indoor residential scenario, this project contributes to the advancement of non-intrusive fall detection solutions for elderly individuals.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The project has involved a great deal of research and investigation. Books, articles, journals, websites, and other relevant sources provided data and studies for the project. The data was an invaluable resource for verifying if the project could be finished in the allotted time. The research and data collection centered on important and pertinent project-related subjects.

Section 2.2 until 2.10.3 discuss everything gained after examined a number of thesis and publications from journals found on the Google Scholar and Mendeley website. A few keywords such as "fall detection", "Kinect", "algorithms", "machine learning", and "object obstruction", were necessary to locate the relevant content. The fall detection system that is integrated with the application and sensors to identify and notify people or caretakers when a person has a fall was the main topic of the literature study.

2.2 Impact of Fall Towards Elderly

Falls can have serious effects on health and well-being, they are a major concern, especially for the elderly and those with specific medical conditions. Serious injuries include fractures, brain trauma, and even death can result from falls. In addition to the acute physical damage, falls can lead to diminished quality of life, a loss of independence, and an increase in fear of falling. There will be a significant financial impact as well because emergency

services, hospital stays, and long-term rehabilitation will result in higher healthcare expenses [4].

The number of falls among older persons is on the rise due to worldwide population aging. The World Health Organization (WHO) reports that falls rank as the second most common cause of unintended or accidental injury deaths globally. Falls frequently cause a decline in health and functional capacity in older persons. In addition to causing physical harm, falls can have psychological implications like dread of falling again, which can result in despair, social isolation, and less physical activity. This decrease in activity feeds into a vicious cycle by raising the chance of additional falls [6].

2.3 Overview of Technologies Used in Fall Detection

2.3.1 Wearable Devices

Accelerometers, gyroscopes, and other body-worn motion sensors are common examples of these devices. They use abrupt changes in orientation or movement to identify falls. Smartwatches, sensors attached to belts, and specific fall detection pendants are a few examples. Because wearables are so portable and can track a user's movements over time, they can instantly warn of a fall. Some users may find them uncomfortable or bothersome, and wearing them continuously is necessary for them to be successful [1].

2.3.2 Ambient Sensors

These systems make use of ambient sensors scattered throughout the house, such as pressure mats, infrared sensors, and ultrasonic sensors. They keep an eye out for environmental changes, like someone lying on the ground, to identify falls. It doesn't require the user to wear any devices and are non-intrusive [1]. Ideal for ongoing surveillance over bigger

regions. Installation work can be costly and intricate. The arrangement of the surroundings and the locations of the sensors may restrict their efficacy.

2.3.3 Vision-Based Systems

These devices analyze visual data to detect falls using cameras and computer vision algorithms. One well-known example is the Microsoft Kinect sensor, which uses bone tracking and depth sensing to follow movements and detect falls. can offer in-depth movement analysis and extensive contextual information. Ideal for discreet surveillance in interior spaces. privacy issues brought on by ongoing video surveillance. Obstacles and lighting can both have an impact on performance [2].

2.4 Algorithms and Techniques for Fall Detection

2.4.1 Skeleton Features-Based Skeleton

Analyzing skeletal characteristics is one of the main ways that the Kinect sensor is used for fall detection. Because the Kinect sensor can track and monitor an individual's skeletal structure, it offers a comprehensive dataset that can be utilized to identify falls based on variations in joint movements and body posture. This technique uses positioning data from the head, shoulders, hips, and knees among other joints in the body to identify anomalous patterns that point to a fall. A fall detection system that makes use of Kinect sensor-extracted skeletal characteristics. Figure 2.1 is a visual representation of the skeletal point of human body that can be detect by Kinect sensor. To remove outliers, the system uses a one-class classifier, which reduces unnecessary or non-fall-related movements and increases fall detection accuracy. This approach's primary benefit is in its capacity to precisely detect falls through an examination of the spatial relationships and movement patterns of various bodily joints [7].

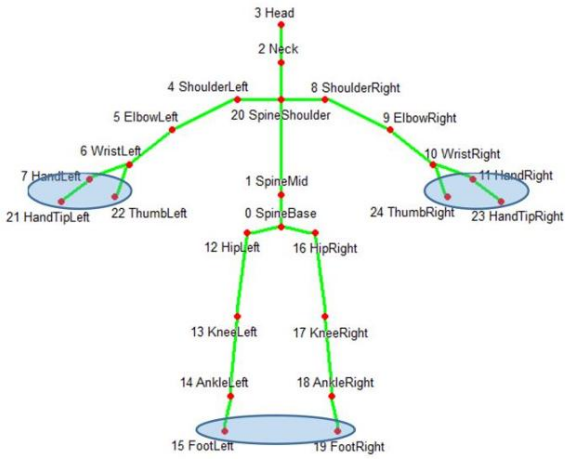


Figure 2.1 Skeleton provided by Microsoft Kinect v2 and points excluded from analysis (grey ovals) [7].

2.4.2 Machine Learning and Deep Approaches

Fall detection systems have been using machine learning and deep learning techniques more and more to improve their accuracy and dependability. These methods train models that can automatically identify patterns linked to falls using massive datasets. A fall detection system that uses information from the Kinect sensor to assess movement patterns and gait using deep convolutional neural networks (CNNs) as shown in the flowchart in figure 2.2. The system's ability to distinguish between typical activities and fall events with high accuracy is derived from its ability to learn intricate details from the skeletal data [8].

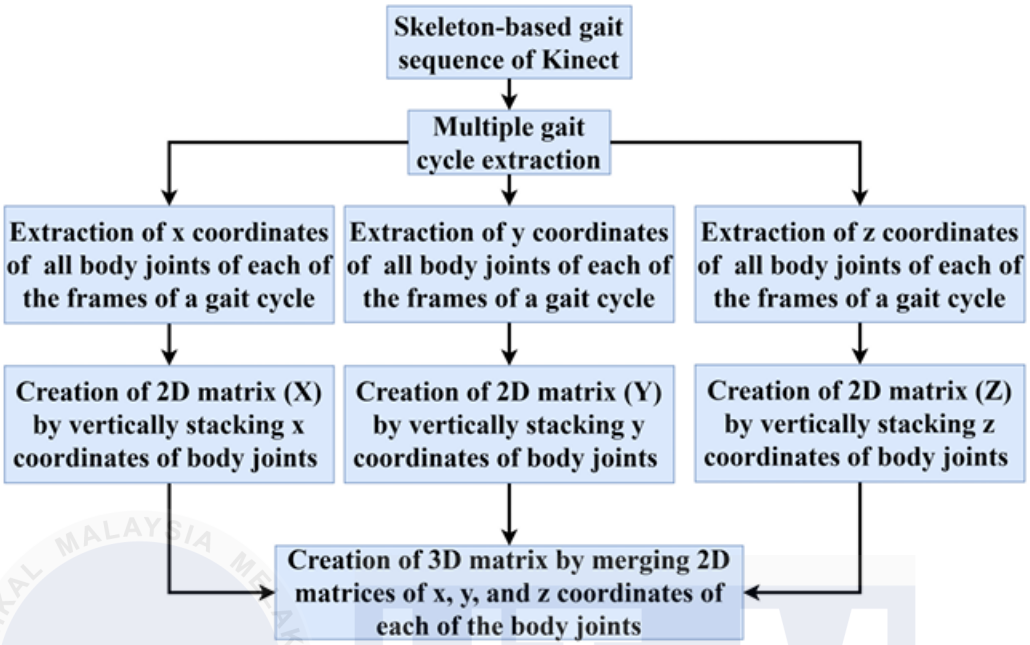


Figure 2.2 Flowchart of a 3D matrix generation from the body joints over the frames of a gait cycle [8].

Tsai and Hsu created a fall detection system that analyzes 3D skeletal data from the Kinect sensor using deep learning algorithms. Figure 2.3 shows the method that was implement by them which use a dataset of skeletal motions to train a deep neural network to identify the features of falls. The system achieves reliable and precise fall detection by utilizing the comprehensive 3D joint positions that the Kinect provides [9].

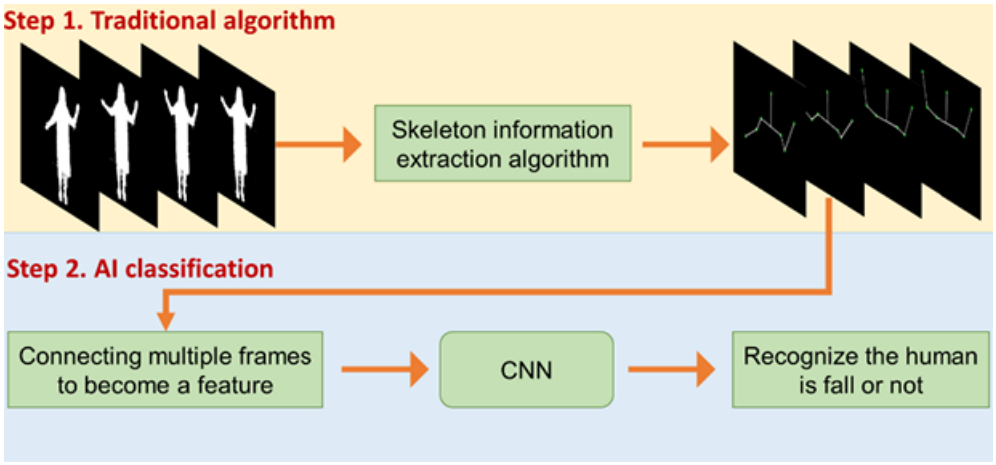


Figure 2.3 Deep Learning System Overview [9].

2.5 Comparative Analysis of Fall Detection Approaches

2.5.1 Survey of Fall Detection Approaches Using Kinect Technology

2.5.1.1 Overview of Fall Detection Methods

Numerous fall detection techniques have been investigated, utilizing diverse technologies and methods. These techniques can be generally divided into three categories: deep learning models, machine learning algorithms, and threshold-based techniques. Each of these categories has unique properties and uses [4].

2.5.1.2 Threshold-Based Methods

Fall detection techniques that use thresholds depend on pre-established levels to recognize falls in relation to variables like inclination angle, acceleration, and velocity. These techniques are appropriate for real-time applications since they are simple and computationally cheap. But because they have trouble adjusting to individual variations and the unpredictability of fall events, their accuracy can be restricted [4]. Figure 2.4 shows the advantages and disadvantages of sensors used in existing fall detection system.

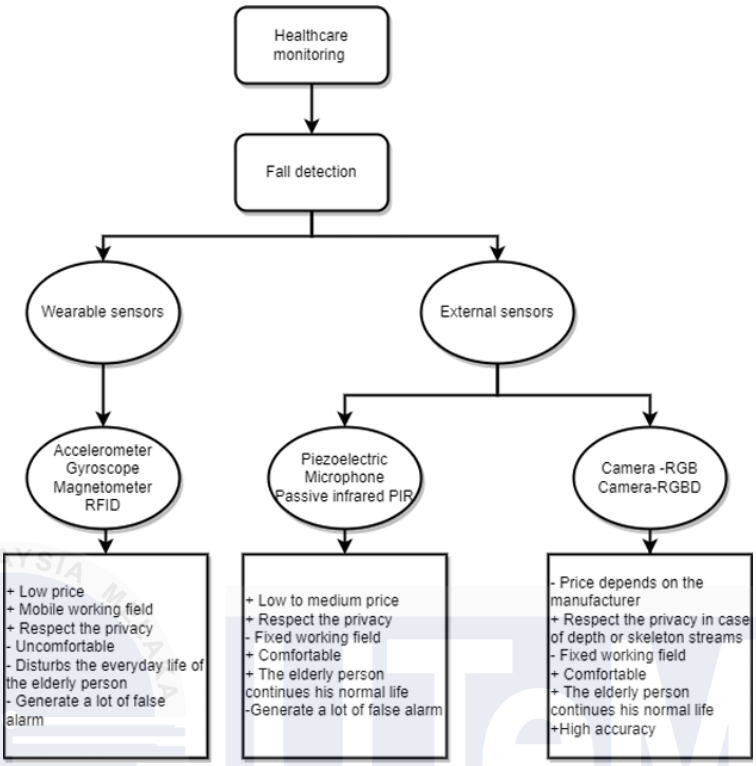


Figure 2.4 Types of sensors deployed in fall detection, with (+) indicating advantages and (-) indicating disadvantages[4].

2.5.1.3 Machine Learning Algorithm

In order to distinguish between fall and non-fall occurrences, classifiers trained on labeled datasets according to figure 2.5 are used in machine learning techniques to fall detection. K-nearest neighbors (k-NN), decision trees, and support vector machines (SVM) are examples of common methods. Compared to threshold-based approaches, these strategies are more accurate and flexible because they are able to identify intricate patterns in the data. However, they can be computationally demanding and require a large amount of labeled data for training, which could limit real-time performance [4].

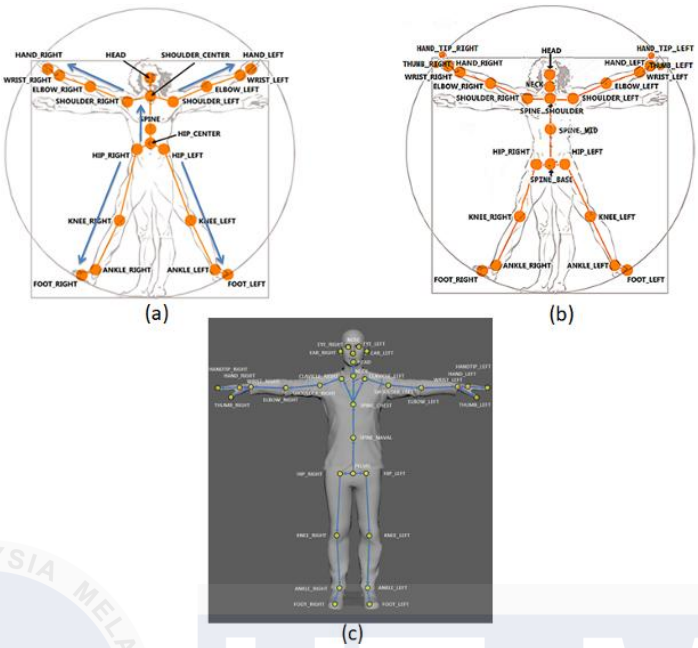


Figure 2.5 Hierarchy of joints provided by (a) V1, (b) V2, and (c) Azure Kinect [4]

2.5.1.4 Comparative Analysis

The various fall detection methods can be compared to see trends toward increasingly advanced deep learning and machine learning methods. These cutting-edge techniques, which improve fall detection systems by utilizing the comprehensive depth and skeleton data supplied by the Kinect sensor just as shown in the general flowchart in figure 2.6, which offer increased accuracy and resilience over conventional techniques [4].

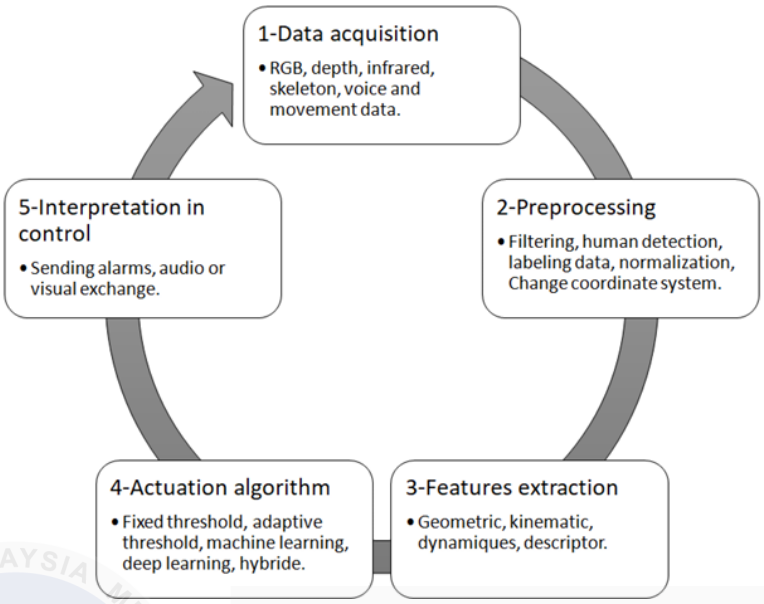


Figure 2.6 General flowchart of Kinect-based fall detection approaches [4].

2.5.2 Computer Vision-Based Methods

2.5.2.1 Overview of Computer Vision-Based Fall Detection

Computer vision-based fall detection methods utilize visual data to identify falls. The Kinect camera, with its ability to capture both RGB images and depth maps, plays a significant role in these methods. Techniques in this category can be divided into image-based analysis, depth map analysis, and hybrid approaches [2].

2.5.2.2 Image-Based Analysis

RGB images from the Kinect camera are used by image-based analytic techniques to identify falls. Methods including silhouette analysis, motion history images (MHI), and backdrop subtraction are frequently used. These techniques take advantage of the rich visual data that RGB photographs provide, allowing for a thorough examination of the scene and motions. They are, however, susceptible to variations in background clutter and lighting, which may compromise accuracy [2].

2.5.2.3 Depth Map Analysis

Depth map analysis is the process of detecting falls using depth maps produced by the infrared sensor on the Kinect. This method examines abrupt changes in elevation, motion patterns in the depth data, and depth gradients. Depth map analysis helps differentiate between falls and other activities since it is less impacted by lighting and offers precise distance measurements. However, accuracy may be limited by the depth sensor's low resolution and range, especially in bigger or more complicated surroundings [2].

2.5.2.4 Hybrid Techniques

RGB and depth data are combined in hybrid approaches to improve fall detection precision. These ways can get around the drawbacks of using only one form of data as explained in figure 2.7 by utilizing both. Better robustness and accuracy are provided by hybrid approaches, which can handle a larger variety of scenarios and environmental variables. To properly handle and integrate the data, they also add to the computational complexity and need for more complex algorithms [2].

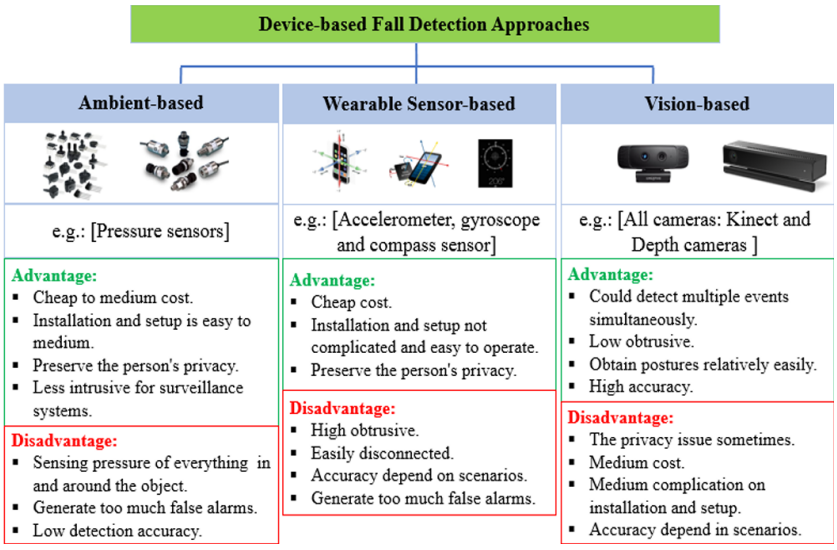


Figure 2.7 The current fall detection methods classification into three device-based approaches: ambient based, wearable sensor-based, and vision-based approaches [2].

2.5.2.5 Combined Kinect Data

The advantages and disadvantages of each strategy show how promising computer vision-based fall detection techniques can be. Even under difficult circumstances, a strong tool for precisely spotting falls is provided by combining depth sensing with conventional RGB imaging as illustrates in figure 2.8. Subsequent investigations could concentrate on enhancing these methods to augment their efficiency and relevance [2].

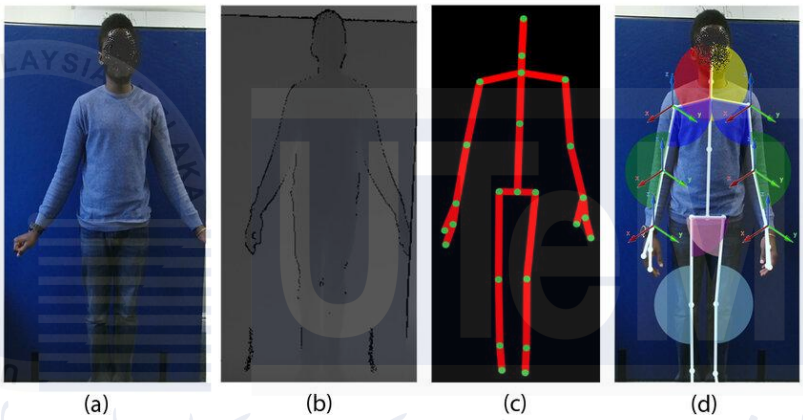


Figure 2.8 Illustration of depth image transformation: (a) colour stream; (b) depth stream; (c) skeleton (joint are shown in green dots); and (d) tracked skeleton and joints (similar joints are presented with the same colour) [2].

2.6 Object Obstruction

Fall detection systems face considerable hurdles when it comes to detecting falls in which just a portion of the body is visible. These issues include maintaining detection accuracy in congested situations, guaranteeing dependable performance under shifting lighting conditions, and effectively identifying fall events despite occlusions produced by objects or other environmental factors. Fall detection systems' efficacy may be jeopardized by misclassification or missing detections resulting from partial body visibility [10].

2.6.1 Techniques for Overcoming Object Obstruction

2.6.1.1 Pedestrian Occlusion Level Classification

A strategy for dealing with object blockage is to categorize the degree of occlusion a pedestrian encounters. Methods in this field concentrate on classifying the degree to which an object impedes an individual. From figure 2.9, it is evident that classification can aid in modifying the detection algorithms to account for occlusion and enhance overall fall detection accuracy even in situations where the body is only partially visible [11].

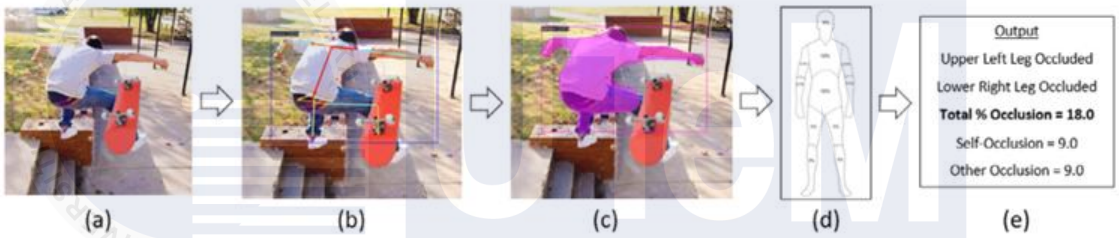


Figure 2.9 Occlusion level classification overview. (a) Read input image (b) Apply keypoint detection to each pedestrian instance and assess keypoint visibility to identify occluded keypoints (c) Correlate visible keypoints with pedestrian mask to confirm visibility [11].

2.6.1.2 Obstruction Detection 4D BIM Construction Planning

Methods for identifying and controlling obstacles in a 4D Building Information Modeling (BIM) environment have been developed for construction planning. By applying comparable concepts to recognize and evaluate obstacles inside the detection region, these techniques can be modified for fall detection, allowing for a more precise interpretation of body components that are only partially visible [10].

2.6.1.3 Real-Time Object Detection for Obstruction Scenarios

Algorithms for real-time object identification are made to recognize and follow objects in a variety of situations, including those with obstacles as provide in figure 2.10. By

using sophisticated neural networks capable of identifying and processing objects that are partially visible, these techniques enhance the system's capacity to detect falls even in situations where the view is partially obscured [12].

Figure 2.10 Image after data enhancement. (1) Magnifcation and brighten, (2) magnifcation and mirror, (3) magnifcation and darken, (4) image reduction and translation, (5) image reduction and translation, and (6) mirror, reduction, and translation [12].

2.6.2 Case Study: Partial Body Detection Using Kinect

2.6.2.1 Fall Detection Base on Point Cloud

According to figure 2.11, a fall detection system using point cloud data was developed by Peng et al. (2019). Because the point cloud data may record depth information, this method works especially well in situations where visibility is limited. It allows the system to identify falls by looking at the spatial arrangement of visible body components [13].

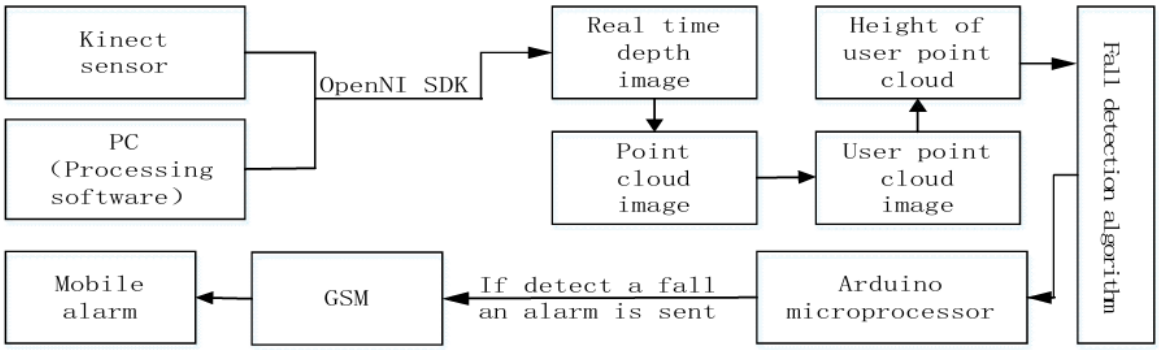


Figure 2.11 Point Cloud System Structure Diagram [13].

2.6.2.2 Detecting Human Falls in Poor Lighting and Object Obstruction Conditions

Zi et al. (2023) suggested techniques that make use of sophisticated object detection and tracking algorithms to identify falls in low light and blocked environments. By accounting for occlusions and low visibility situations, these techniques are intended to improve fall detection reliability and guarantee precise detection even in difficult circumstances such as in figure 2.12 [14].

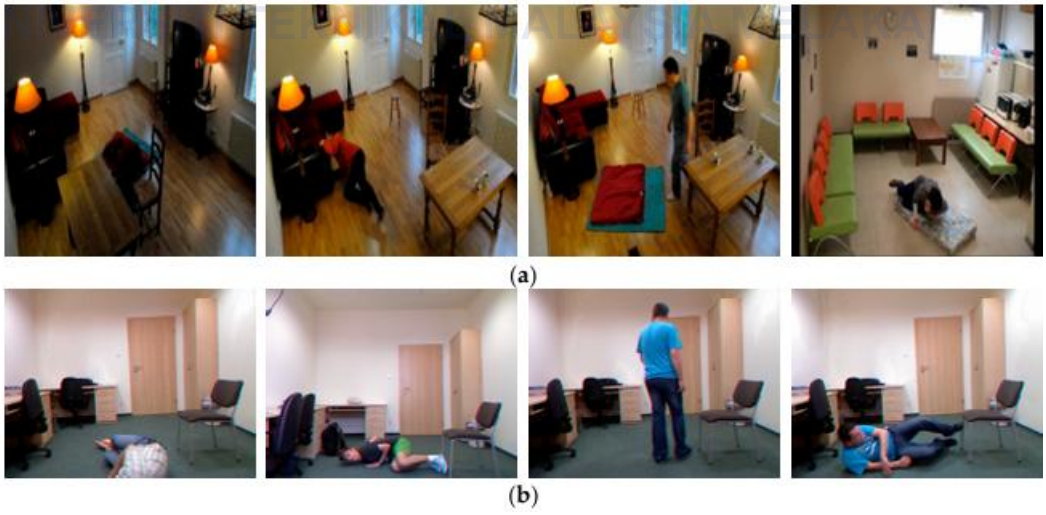


Figure 2.12 Example video frames from the publicly available datasets (a). Le2i and (b) URFD datasets [14].

2.7 Integration of Fall Detection with Alerting System

2.7.1 Smartphone-Based Online System with Alert Notification

Fall detection system integration with smartphone-based alert notifications entails creating applications that, upon detection of a fall, can instantly notify emergency services or caretakers. Figure 2.13 proves that these solutions provide instant notification and contextual information about the fall occurrence by leveraging the computing power and connection of smartphones. Typically, the design consists of fall detection algorithms, an intuitive alarm configuration interface, and a robust communication protocol to guarantee timely alert delivery. These systems improve people's safety and freedom by making sure that assistance is quickly summoned in the case of a fall [15].

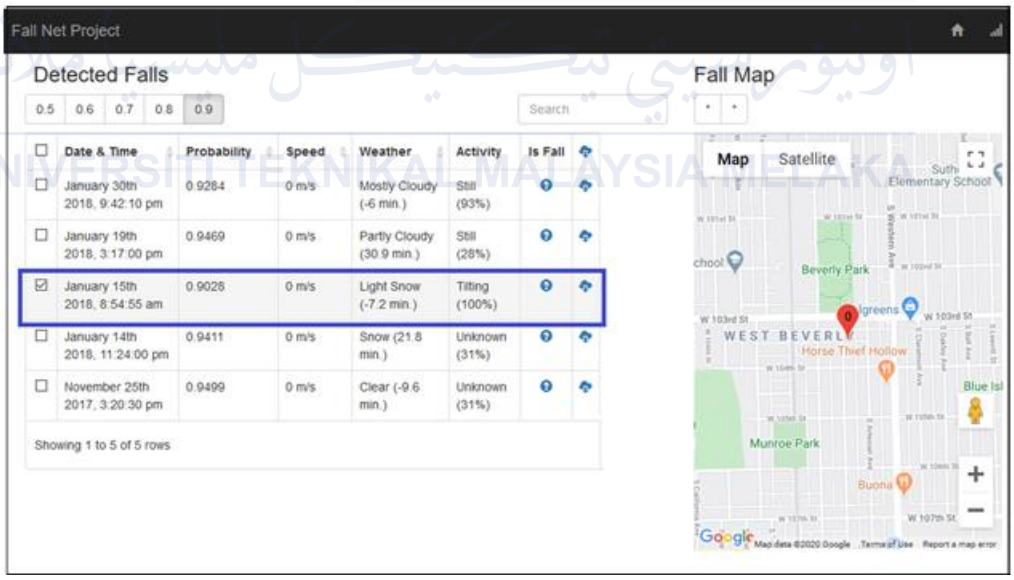


Figure 2.13 An example of the web portal data summary for a single study participant [15].

2.7.2 Fall Detection and Emergency Notification System

In order to build a reliable alerting system, a complete fall detection and emergency notification system integrates sensors, detecting algorithms, and communication modules.

This system not only recognizes falls but also notifies authorized contacts or services in case of emergency. The implementation consists of setting up a communication system to transmit alerts by SMS, email, or automated calls; configuring sensors to accurately capture fall events; and designing algorithms to process the sensor data and detect falls. These kinds of technologies play a critical role in both delivering aid promptly and possibly lessening the degree of injuries sustained from falls [6].

2.8 Case Studies and Applications

2.8.1 Kinect4FOG for Monitoring Parkinson's Patients

Figure 2.14 provides a visual representation of one noteworthy use of Kinect technology in healthcare is the Kinect4FOG system, which was created especially to track and enhance movement in Parkinson's disease patients. This system makes use of Kinect sensors to analyze patients' movements and gaits, giving patients and healthcare professionals comprehensive analysis and feedback. The system enables prompt interventions and individualized treatment regimens by identifying movement abnormalities that may point to a higher risk of falls. Kinect4FOG helps manage Parkinson's disease progression and improves patients' quality of life by lowering fall risk and improving mobility in patients under constant observation [16].

Figure 2.14 System's companion smartphone application in action [16].

2.8.2 Video Surveillance for Fall Detection

Another important use for video surveillance systems is fall detection, especially for watching elderly people at home or in care facilities. With the use of sophisticated image processing and machine learning techniques, these systems use cameras to continuously monitor and analyze the surroundings in order to identify falls. The method of detecting falls entails identifying abrupt movements or changes in posture that point to a fall and then notifying emergency personnel or caretakers of the situation. This approach improves the safety of senior citizens by guaranteeing that falls are identified quickly and help is given right away, which reduces the risk of serious injuries [17].

2.9 Kinect Specific Applications and Innovations

2.9.1 Kinect in Gait Analysis and Mobility Monitoring

Since Kinect technology can follow movement patterns and gather comprehensive skeletal data, it has found widespread application in gait analysis and mobility monitoring which can be seen in figure 2.15. The Kinect4FOG system analyzes a patient's gait and provides rehabilitation feedback in order to monitor and enhance mobility in individuals with Parkinson's disease [18]. The system can identify mobility problems and assist in customizing interventions to improve the patient's quality of life by utilizing Kinect sensors. Similar to this, KinectGaitNet recognizes gait patterns using deep convolutional neural networks as provided in figure 2.16, offering precise and effective gait analysis that may be applied to a range of healthcare applications [8].

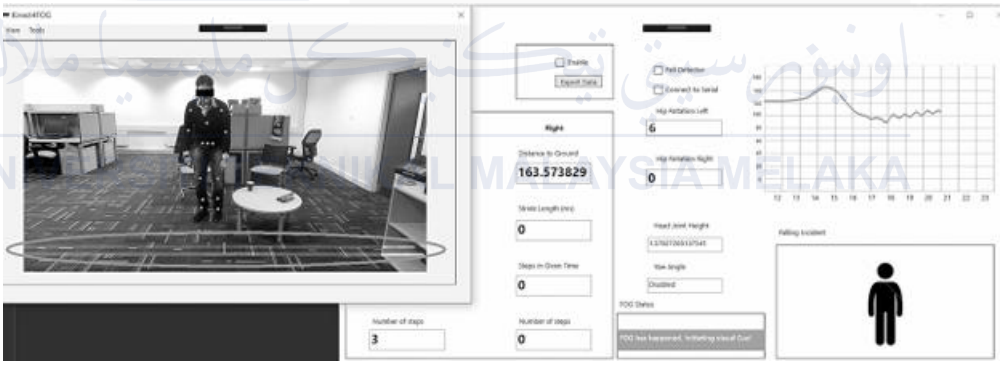


Figure 2.15 Graphical User Interface for the developed software [16].

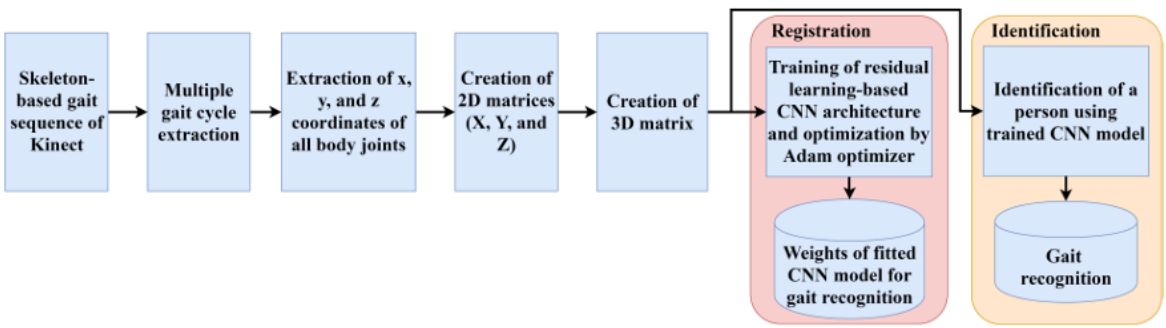


Figure 2.16 Overall system flowchart of the proposed framework [8].

2.9.2 Posture Recognition and Human Activity Analysis

Identifying posture and analyzing human behavior are essential for creating fall detection systems that work. The figure 2.17 presents the process which resulting the accuracy of recognizing various postures and activities is improved by applying fuzzy and rough logic to posture identification, which is crucial for fall detection [5]. Additionally, in order to monitor different activities and identify anomalous postures that can point to a fall, Kinect v2's human posture recognition technology collects and analyzes skeletal data. In practical situations, these methods increase the dependability of fall detection systems [19].

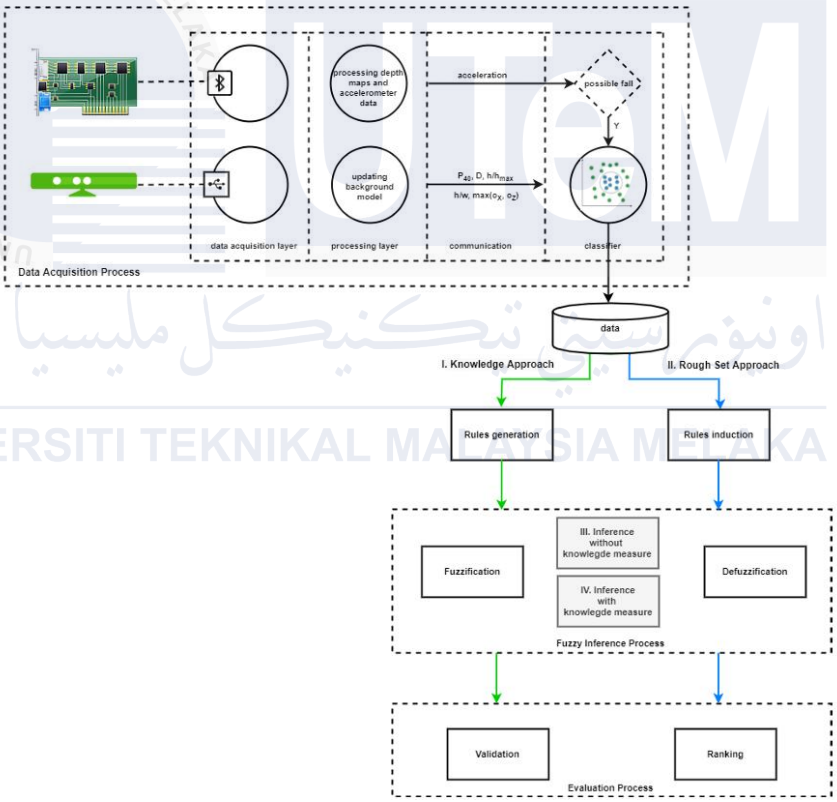


Figure 2.17 Scheme of the fuzzy inference process [19].

2.9.3 Multi-Sensor Integration and Advanced Processing

Fall detection systems can be greatly improved by utilizing sophisticated processing techniques and integrating many sensors. For example, a system can improve fall detection accuracy by combining data from many sensors through dual-channel feature integration. To

provide accurate fall detection, this technique analyzes a variety of variables, including motion and skeletal data as proven in figure 2.18 [20]. It can be seen in figure 2.19, to improve the robustness of fall detection, time-level decision fusion classification is another strategy that integrates decisions taken at various time levels [21].

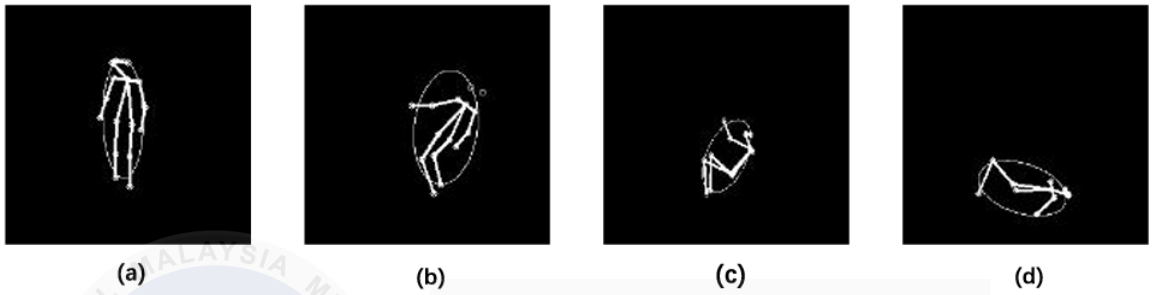


Figure 2.18 Human body external ellipse [20].

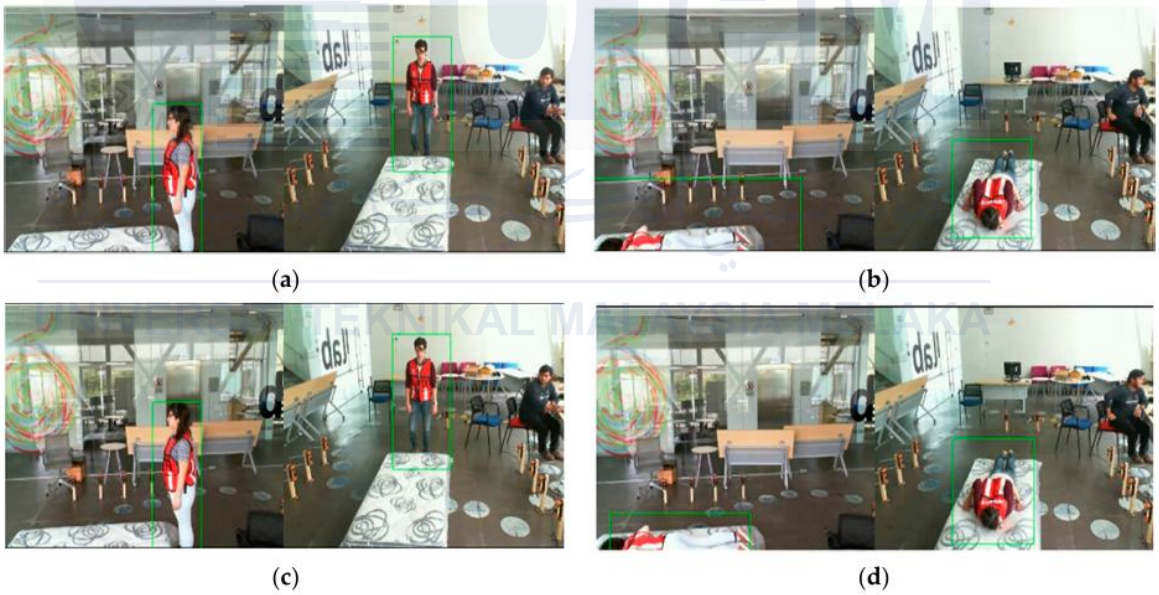


Figure 2.19 Results of person detection using YOLOv3 and YOLOv7. (a) Detection results of YOLOv3 for normal activity. (b) Detection results of YOLOv3 for fall activity. (c)Detection results of YOLOv7 for normal activity. (d) Detection results of YOLOv7 for fall activity [21].

2.9.4 Real-World Deployment and User Feedback

In order to enhance system performance, real-world fall detection system deployments involving Kinect technology require resolving pragmatic issues and obtaining

user input. An older person's mobility may now be tracked and analyzed with a Kinect-based platform, which provides important information for fall detection [3]. Furthermore, to ensure reliable performance and user satisfaction, a versatile fall detection framework that makes use of object recognition and motion analysis has been designed to handle a variety of real-world circumstances [22].

2.10 Future Directions and Challenges

2.10.1 Emerging Trend in Fall Detection

Fall detection technology is always changing, and a number of new developments are expected to improve its dependability and efficacy. The combination of machine learning (ML) and artificial intelligence (AI) approaches is one prominent development that enables more in-depth study of environmental elements and movement patterns [23]. With these technologies, systems may learn from enormous volumes of data, increasing their precision in fall detection and lowering false alarms. Furthermore, wearable technology and the Internet of Things (IoT) are being used more and more to build networked systems that offer thorough monitoring of the elderly [24]. These systems can collect information from several sources, providing a comprehensive picture of the health and activity levels of the user [25]. According to figure 2.20, it is an example of a fall detection system that applied the IoT.

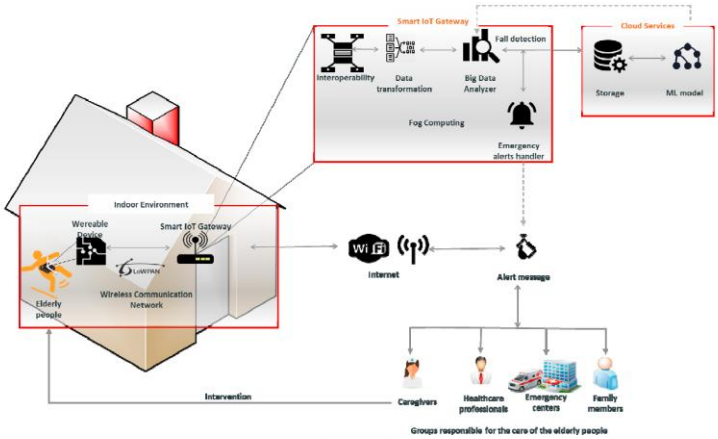


Figure 2.20 The FD IoT System Overview [24].

2.10.2 Challenges and Limitations

There are still a number of obstacles and restrictions in the field of fall detection, despite tremendous progress. Making sure detecting systems are accurate and dependable in a variety of settings and circumstances is one of the main problems. Occlusions, background noise, and lighting are only a few examples of the variables that can seriously impair the operation of vision-based systems like Kinect-based ones [26]. The possibility of false positives and false negatives, which might either fail to signal when a fall has occurred or induce unwarranted alarm, is another restriction. Another difficulty is privacy issues, particularly with systems that entail ongoing video surveillance. If users believe their privacy is being violated, they could be reluctant to use such services [27].

2.10.3 Potential Improvements and Innovations

Several potential advancements and innovations are being investigated as solutions to these problems. Improving the resilience of algorithms to manage various settings as shown in figure 2.21 and circumstances is an essential area of emphasis. Creating increasingly complex machine learning models that can adjust to various situations and enhancing the caliber of sensor data fusion are two examples of this. Enhancing the accuracy of fall detection systems can also be accomplished through innovations in sensor technology, such as the use of additional sensors like LIDAR or higher resolution cameras. Iterative testing and user feedback can also be used to improve these systems and make them more end-user-friendly. Fall detection systems must also boost user acceptability and confidence, which calls for privacy-preserving measures like on-device processing and data anonymization [28].

Figure 2.21 The detection results of fall in different directions [28].

2.11 Sensor Comparison from Previous Work Related to the Project

Table 2.1 Comparison between sensors used in fall detection system

No	Year	Sensor	Purpose and Specifications	Advantages	Disadvantages
1	2023 [29]	Kinect sensor	• Montion sensor. • Tracks motion and gestures. • Comprises of camera, infrared sensors	• Tracks skeletal data for motion tracking and accurate fall detection. • Non-intrusive • Real-time monitoring with immediate alerts	• Limited range (~4.5 meters). • Needs precise placement or multiple devices. • High power consumption. • Raises privacy concerns.
2	2023[30]	Camera for Image and Video Capture	• Human body movement tracking • Non-intrusive monitoring • High resolution and frame rate	• Captures high-resolution visuals. • Records detailed environmental context. • Affordable compared to advanced sensors.	• No depth sensing, less accurate for fall detection. • High computational demand for processing visuals. • Significant privacy concerns.

3	2021[31]	Accelerometer	Measure the acceleration and determine the changes in velocity and movement.	• Compact and energy efficient. • Detects sudden motion changes effectively. • Affordable	• Requires users to wear the device. • No environmental or spatial data. • Limited to motion-based detection.
4	2024 [32]	Gyroscope	Measure the angular velocity and provides information of rotational movement.	• Measures rotational movements accurately. • Small, efficient, and wearable-friendly. • Works well with accelerometers.	• Requires wearables, potentially inconvenient. • Only provides rotational data, lacks spatial context. • Limited accuracy without additional sensors.

The Kinect was chosen for its non-intrusive design, eliminating the need for wearables. Its 3D depth sensing and skeletal tracking offer spatial and motion data, making it more accurate than standard cameras or motion sensors. While it has limitations like range and power use, its ability to monitor and analyze falls in real-time makes it ideal for this project.

2.12 Journal Comparison from Previous Work Related to the Project

Table 2.2 Journal comparison

No.	Year	Title	Software	Hardware	Findings
1	2018	An image-based fall detection system for the elderly [25]	not specified	Camera	• Developed image-based fall detection system achieved high accuracy in detecting falls among the elderly. • System showed a notable reduction in false alarms compared to existing systems. • Findings suggest that image-based approach has potential to improve reliability and effectiveness of fall detection systems for the elderly.
2	2018	Computer Vision Based Fall Detection Methods Using the Kinect Camera : A Survey [2]	Computer vision algorithms	Kinect sensor	• Kinect-based fall detection methods show promise in accurately and reliably detecting falls among the elderly. • These methods use the Kinect camera's depth sensing capabilities to monitor changes in body position and posture. • Early detection of falls is facilitated by these techniques. • Challenges include occlusions (objects blocking the view), varying lighting conditions, the need for robust algorithms • Further research and development are needed to address these challenges.

No.	Year	Title	Software	Hardware	Findings
3	2018	Design and Development of the Fall Detection System based on Point Cloud [13]	Kinect SDK	Kinect Sensor	• No equipment needs to be worn during system operation, ensuring comfort. • Kinect uses point cloud images and color spectrum for human detection, protecting privacy. • The infrared camera is unaffected by external illumination. • The system provides continuous real-time detection of the human body for 24 hours. • Detection efficiency is improved.
4	2018	Fall Detection System for Elderly People Using IoT and Big Data [24]	Contiki OS	LSM6DS0; 3D-axis accelerometer	• The system was evaluated for recognizing three types of falls: forward, backward, and lateral falls while walking caused by a slip. • Recognition accuracy: 91.67% • Precision: 93.75% • Gain: 91.67% • The high success rate in fall detection is indicated by these metrics.

No.	Year	Title	Software	Hardware	Findings
5	2019	Cell-Based Transport Path Obstruction Detection Approach for 4D BIM Construction Planning [10]	Computer algorithm	none	• The cell-based transport path obstruction detection approach effectively identifies potential obstructions in construction projects. • Integration into 4D BIM allows for accurate visualization and simulation of material and equipment movement. • Enables better-informed decisions and improved construction planning. • Potential to enhance efficiency, safety, and overall success of construction projects.
6	2019	Kinect4FOG: monitoring and improving mobility in people with Parkinson's using a novel system incorporating the Microsoft Kinect v2 [16]	Software for analyzing gait patterns, machine learning algorithms.	Kinect sensor.	• Kinect4FOG effectively monitors and improves mobility in people with Parkinson's disease. • Provides a non-invasive and cost-effective method. • Allows for early intervention and personalized treatment strategies. • Highlights potential to improve mobility and quality of life for Parkinson's patients.

No.	Year	Title	Software	Hardware	Findings
7	2019	A skeleton features-based fall detection using Microsoft Kinect v2 with one class-classifier outlier removal [7]	Fall detection algorithm	Kinect sensor.	• Skeleton features-based fall detection system using Kinect v2 achieved high accuracy and reduced false alarms. • One-class classifier used for outlier removal. • System focuses on unique skeletal characteristics for improved reliability. • Skeleton features-based approach coupled with one-class classifier enhances performance of fall detection systems.
8	2019	Human Posture Recognition and Fall Detection Using Kinect V2 Camera [5]	Kinect SDK, NITE SDK	Kinect Sensor	• Proposed integrated neural network fall detector application operates in real-time. • Based solely on depth maps, ensuring privacy and functioning in poor light conditions.
9	2019	Implementation of Fall Detection System based on 3D Skeleton for Deep Learning Technique [9]	Kinect SDK	Kinect Sensor	• Implemented on NVIDIA Jetson TX2 platform. • Tested in real demonstration environment. • Achieves 15 frames per second for real-time implementation.

No.	Year	Title	Software	Hardware	Findings
10	2019	Kinect-Based Platform for Movement Monitoring and Fall-Detection of Elderly People [3]	Kinect SDK	Kinect Sensor	• Developed a Kinect-based platform for monitoring movement and detecting falls in elderly people. • Platform includes a fall detection algorithm and a web application for remote monitoring. • Tested the system with 30 volunteers and achieved a 96.3% success rate in fall detection. • Found Kinect to be effective in monitoring movement and detecting falls.
11	2020	Fall Detection Based on Dual-Channel Feature Integration [20]	Computer algorithm	Accelerometer and gyroscope sensors.	• Dual-channel feature integration approach for fall detection achieved higher accuracy and reliability. • Combining features from acceleration and angular velocity channels improved sensitivity in detecting falls and reduced false alarms. • Dual-channel feature integration is an effective strategy for enhancing fall detection system performance.

No.	Year	Title	Software	Hardware	Findings
12	2020	A Fall Detection and Emergency Notification System for Elderly [6]	Kinect SDK	Kinect Sensor	• System calculates and analyzes velocities of body joints and angles of body vectors to distinguish falls from daily activities. • Differentiates between three types of falls: Prone Position, Crawl Position, and Kneel Position. • Fall notification based on Q-Learning algorithm, considering contact person's probability of answering and level of busyness.
13	2020	An Elderly Fall Detection System Using Depth Images [23]	Kinect SDK	Kinect Sensor	• Microsoft Kinect's depth image resolution decreases as distance increases. • Decreased resolution makes background subtraction and depth image segmentation challenging.
14	2020	Old man fall detection based on surveillance video object tracking [26]	Object tracking and fall detection algorithm	none	• Fall detection system based on surveillance video object tracking effectively detects falls in elderly individuals. • Analyzes surveillance video footage and tracks movements to detect falls in real-time. • Shows promise in improving timely response to falls and reducing injury risk for elderly individuals.

No.	Year	Title	Software	Hardware	Findings
15	2020	Fall detection system for people using video surveillance [17]	Computer algorithm	none	• Video surveillance-based fall detection system effectively detects falls and triggers alerts in real-time. • Analyzes footage for specific movement patterns and behaviors associated with falls. • Accurately detects fall events, improving safety and well-being of individuals at risk. • Provides peace of mind for caregivers and family members.
16	2021	A Smartphone-based Online System for Fall Detection with Alert Notifications and Contextual Information of Real-Life Falls [15]	android 6.0.1; Purple Robot, preinstalled sensor data collection app	Phone; accelerometer and gyroscope	• Smartphone-based system requires minimum 2G signal for sending alert notifications. • Preferably uses 4G-LTE for exporting sensor data. • Falls in locations without cellular reception won't be centrally detected for real-time notification.

No.	Year	Title	Software	Hardware	Findings
17	2022	An objective method for pedestrian occlusion level classification [11]	Computer algorithm	none	• Developed method for pedestrian occlusion level classification. • Effective in objectively categorizing extent of pedestrian occlusion in scenes. • Analyzes visual features related to visibility of body parts and degree of occlusion. • Accurately classifies occlusion levels. • Objective method can improve performance of pedestrian detection and tracking systems, especially in challenging environments.
18	2022	Real-Time Object Detection for the Running Train Based on the Improved YOLO V4 Neural Network [12]	Improved YOLO V4 neural network.	Cameras for capturing video data.	• Real-time object detection system based on improved YOLO V4 neural network effective for detecting objects near running train. • Analyzes video data from train-mounted cameras to identify obstacles or hazards in real-time. • Deep learning techniques enhance safety and efficiency of train operations by providing early detection of potential hazards.

No.	Year	Title	Software	Hardware	Findings
19	2022	Smart Assistive System for Visually Impaired People Obstruction Avoidance Through Object Detection and Classification [27]	Object detection and classification algorithm	Camera or sensor for capturing image or video data.	• Smart assistive system for visually impaired effectively avoids obstructions through object detection and classification. • Analyzes image or video data from camera or sensor to detect and classify obstacles in real-time. • System improves mobility and safety of visually impaired individuals. • Provides greater independence and confidence in navigating surroundings.
20	2022	Application of Fuzzy and Rough Logic to Posture Recognition in Fall Detection System [19]	Fuzzy and rough logic algorithms	Sensors or cameras for capturing posture data.	• Application of fuzzy and rough logic to posture recognition in fall detection system improves ability to distinguish normal activities from falls. • Analyzing posture data using these techniques increases accuracy in detecting falls and reduces false alarms. • Incorporating fuzzy and rough logic enhances performance and reliability of fall detection systems. • Makes systems more suitable for real-world applications.

No.	Year	Title	Software	Hardware	Findings
21	2022	KinectGaitNet: Kinect-Based Gait Recognition Using Deep Convolutional NeuralNetwork [8]	Deep CNN for gait recognition, Kinect (SDK)	Kinect sensor for capturing gait data.	• KinectGaitNet achieved high accuracy in gait recognition using Kinect sensor data. • Deep CNNs used to accurately identify individuals based on unique gait patterns. • KinectGaitNet offers promising approach to gait recognition. • Potential applications in security, surveillance, and healthcare for reliable biometric identification.
22	2023	Image-based fall detection in bus compartment scene [28]	Image processing algorithms	Cameras for capturing images.	• Image-based fall detection system for bus compartments effectively detects falls in real-time. • Analyzes images from cameras installed in bus compartments. • Accurately detects fall events. • Enhances safety of bus passengers by providing timely alerts to bus drivers or authorities.

No.	Year	Title	Software	Hardware	Findings
23	2023	A Flexible Fall Detection Framework Based on Object Detection and Motion Analysis [22]	Object detection algorithm, motion analysis algorithm, fall detection algorithm	Cameras or sensors for capturing video data.	• Flexible fall detection framework based on object detection and motion analysis effective in detecting falls in various environments. • Combines techniques for improved accuracy and reliability in detecting fall events. • Suggests integrated approach enhances performance of fall detection systems for real-world applications.
24	2023	Detecting Human Falls in Poor Lighting: Object Detection and Tracking Approach for Indoor Safety [14]	Object detection algorithm, motion analysis algorithm, fall detection algorithm	Cameras or sensors for capturing video data.	• Object detection and tracking approach effectively detects human falls in poor lighting conditions. • Utilizes techniques to accurately detect falls in challenging lighting conditions. • Offers reliable method for enhancing indoor safety by improving fall detection in poorly lit areas.

No.	Year	Title	Software	Hardware	Findings
25	2023	Fall Detection Approaches for Monitoring Elderly HealthCare Using Kinect Technology: A Survey [4]	fall detection algorithms and techniques, Kinect SDK	Kinect sensor for capturing depth and skeleton data.	• Kinect technology widely adopted for fall detection in elderly healthcare. • Approaches include machine learning algorithms, rule-based systems, and combination methods. • Kinect-based systems show promising results in accuracy and efficiency. • Highlights potential of Kinect technology in monitoring elderly healthcare.
26	2024	Fall Recognition Based on Time-Level Decision Fusion Classification [21]	fusion algorithm	Accelerometers or gyroscopes for capturing motion data.	• Time-level decision fusion classification approach improves accuracy of fall recognition. • Combines multiple classifiers at different time levels for enhanced performance. • System shows improved reliability in detecting falls across various scenarios. • Decision fusion techniques make fall recognition systems more suitable for real-world applications.

The table compares various research studies on fall detection systems, highlighting their software, hardware, and key findings. Many studies utilize Kinect sensors (either v1 or v2) to track body movements, leveraging depth sensing and skeletal tracking for fall detection. Some research explores alternative technologies, such as accelerometers, gyroscopes, and image-based methods. The findings indicate that Kinect-based systems generally achieve high accuracy, provide real-time monitoring, and offer non-intrusive solutions for elderly care.

Recent studies (2022-2024) explore advanced detection techniques, including deep learning, object tracking, and decision fusion algorithms, demonstrating improved performance in challenging scenarios. The research highlights the growing potential of Kinect technology and hybrid approaches for enhancing fall detection in elderly healthcare and indoor safety applications.

2.13 Summary

This chapter explored various fall detection methods, categorized into three primary approaches: smartphone-based systems, wearable-sensor systems, and vision-based systems. Each category utilizes different technologies and techniques to detect falls effectively. Smartphone-based systems primarily leverage built-in sensors such as accelerometers and gyroscopes to monitor motion and detect sudden changes indicative of a fall. Wearable-sensor systems, on the other hand, rely on external devices like accelerometers, gyroscopes, and other specialized sensors attached to the user's body to collect motion and positional data. Vision-based systems utilize cameras or depth sensors, such as the Kinect sensor, to capture RGB and depth information for visual analysis of human movement.

To develop robust and reliable fall detection systems, researchers have employed advanced computational techniques, including thresholding algorithms, machine learning, and deep learning models. Thresholding algorithms are commonly used for simple and efficient fall detection based on predefined criteria, such as abrupt changes in acceleration or position. Machine learning models enhance the system's ability to distinguish falls from non-fall activities by training on large datasets of motion patterns. Deep learning techniques further improve accuracy by extracting complex features from sensor data or images, enabling the system to recognize falls even under challenging conditions, such as partial occlusions or cluttered environments.

This chapter highlights the strengths and limitations of each method, providing a comprehensive overview of the current advancements and challenges in fall detection research.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In this chapter, the implementation of a real-time fall detection system for the elderly, with consideration for obstructions using Kinect, was described. Section 3.2 detailed the project execution flow and planning; section 3.2.1 explained the flowchart outlining the system design, while section 3.2.2 covered project planning aspects, including the timeline and the duration of each task. Section 3.3 discussed the development of algorithms for fall detection, including handling all real-world obstacles such as barriers and lighting conditions. It also covered the block diagram and the project flowchart. Section 3.4 listed the software and hardware used, including Kinect SDK, Visual Studio, and the Kinect sensor itself. While section 3.5 explains the setup of the project during the testing and collecting data stage of the project and section 3.6 explain the Sustainable Development Goals (SDG) that correlates to the project. Finally, section 3.7 summarized the chapter.

3.2 Project Design

3.2.1 Project Execution Flow

The goal of the project is to implement a Kinect-based fall detection system to assist in monitoring elderly individuals. In the first phase, research was conducted to understand different fall detection techniques, algorithms, and technologies, including the use of Kinect sensors. Existing fall detection systems were analysed to identify suitable hardware and

software, as well as to establish criteria for testing fall scenarios. Data was also collected to assess the limitations and advantages of Kinect-based systems.

In the second phase, a fall detection algorithm was developed, focusing on identifying falls under normal conditions. This algorithm was rigorously tested, and upon achieving satisfactory results, advanced functions were incorporated. These functions included detecting falls with object obstructions, detecting falls in various lighting conditions, differentiating non-fall postures (e.g., sitting on a chair or floor), handling multiple people in the frame, identifying multiple fall postures, and integrating a help command for immediate alerting in emergencies.

The completed system was tested extensively to ensure it could accurately detect falls under different conditions and scenarios. Data from these tests were collected and analysed to validate the results. The final system achieved the goal of developing an efficient and reliable Kinect-based fall detection system with advanced features. The process combined research, algorithm development, software testing, and analysis to provide a comprehensive fall detection solution.

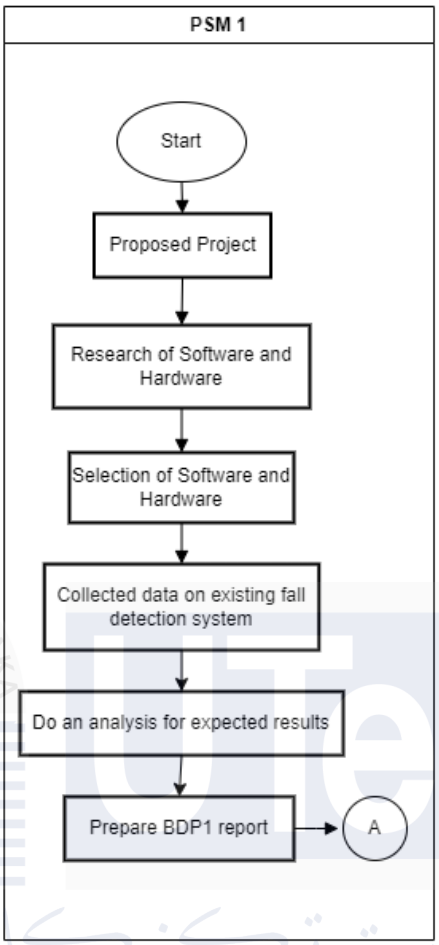


Figure 3.1 PSM 1 Project Execution Flow

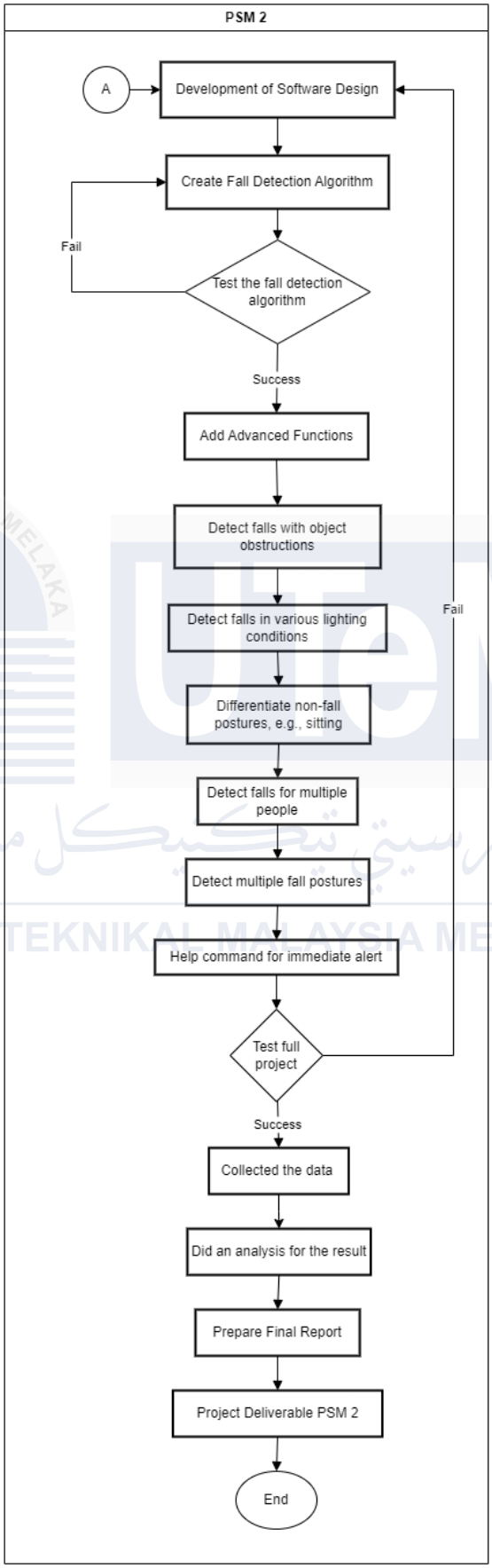


Figure 3.2 PSM 2 Project Execution Flow

3.2.2 Project Planning

Table 3.1 Outline Planning

Activity	Duration (weeks)	Start week	End week
PSM 1			
PSM1 project planning	2	1	2
Research on software and hardware	4	3	6
Research on Previous related projects	6	6	11
Prepare PSM 1 report	5	9	14
PSM 2			
Development of fall detection algorithm	6	15	20
Troubleshoot system program	2	18	19
Improvement of system program	7	19	25
Testing project	2	25	26
Collecting data for analysis	3	24	26
Presentation preparation	2	26	27
Prepare PSM 2 report	1	28	28

The project outline planning provides a clear timeline and structure for the development of the Kinect-based fall detection system, divided into two phases: PSM1 and PSM2. In PSM1,

the project began with two weeks of planning, where the scope, objectives, and methodology were established to ensure the project had a well-defined direction. This was followed by four weeks of research on suitable software and hardware tools, ensuring that the necessary resources were identified for implementing the fall detection system.

Subsequently, six weeks were dedicated to studying previous related projects. This research provided valuable insights into existing methodologies and helped refine the approach for the project. The findings, along with progress during PSM1, were compiled into a report over five weeks, from Week 9 to Week 14, concluding the first phase.

In PSM2, the focus shifted to implementation and testing. The development of the fall detection algorithm began in Week 15 and lasted six weeks, forming the foundation for the system's functionality. Once the algorithm was developed, troubleshooting was carried out in Weeks 18 and 19 to identify and resolve issues in the program. This was followed by seven weeks of improving the system, where advanced features such as detecting falls in obstructed views, handling multiple users, and differentiating non-fall postures were integrated and refined.

From Weeks 25 to 26, comprehensive testing was conducted to validate the system's functionality and reliability. During this period, data was also collected for analysis to assess the accuracy and performance of the fall detection system. In Weeks 26 and 27, the team prepared for the project presentation by creating slides, rehearsing, and organizing results for effective delivery. Finally, in Week 28, the second phase concluded with the preparation of the PSM2 report, documenting all aspects of the project, including methods, results, and conclusions.

This structured timeline ensured a systematic approach to the project, with each phase building upon the previous one. By adhering to this plan, the project was executed efficiently, culminating in a comprehensive and reliable fall detection system.

3.3 System's Algorithm

Developing a Kinect-based fall detection system with the ability to detect falls even under partial body visibility and in the presence of multiple individuals was the aim of this research. This project was executed systematically, progressing through several phases from the conceptualization of the idea to the implementation and testing of the final system. Each step was carefully planned and executed to ensure reliable operation and comprehensive fall detection coverage.

To fully develop the system, the project involved analysing various scenarios, including falls with partial visibility due to object obstruction, as well as movements involving multiple individuals in the frame. Based on these requirements, a fall detection algorithm was designed and implemented using Visual Studio. Subsequently, the algorithm's performance was tested and evaluated to ensure accuracy in detecting falls while minimizing false alarms.

The implementation phase involved the use of the Kinect sensor to capture body joint data, which served as input to the algorithm. The algorithm employed advanced techniques, including the analysis of joint positions and motion patterns, to identify falls effectively. Once the algorithm was developed, the system was debugged and improved iteratively to resolve any errors and optimize its performance.

The finalized fall detection system was then simulated and tested with various scenarios to ensure its behaviour matched the expected outcomes without errors. This testing phase verified that the system could accurately detect falls under different conditions, such as partial body visibility or the presence of multiple people. The results confirmed that the system met the design requirements and performed reliably.

This thorough and systematic approach ensured the successful development of a Kinect-based fall detection system capable of addressing the challenges posed by real-world scenarios.

3.3.1 Fall Detection Algorithm

The provided pseudocode outlines the design and functionality of a Kinect-based fall detection system. This system is intended to monitor body movements in real-time and trigger alerts in the event of a detected fall. The structure of the code is modular, making it easier to implement, test, and maintain. Below is an explanation of its key components and flow.

The *positionThreshold* variable is key to determining whether a potential fall has occurred. It is designed to track the relative movement of the subject (usually an elderly person) within the Kinect's field of view. Essentially, this threshold defines the acceptable range of motion for a person standing or moving normally. The system compares the position of the detected body joints to this threshold to decide whether a fall has taken place.

When the system detects the user's skeletal data via the Kinect sensor, it evaluates the position of key joints, such as the head, torso, and legs. The *positionThreshold* is used as a reference to compare how far these joints have moved from their typical standing or walking

positions. A fall is generally detected when the system identifies that these joints, especially the torso, have dropped significantly in relation to the threshold. For example, if the torso is suddenly much lower than expected and is outside the predefined range, the system may classify this as a fall.

The fall detection mechanism operates by continuously monitoring the skeletal data and comparing it to the set position thresholds. The program is designed to detect abnormal joint positions that would indicate a fall. For instance, if the subject's torso moves below a certain height threshold, it might suggest that the person has collapsed to the ground, triggering a fall event.

The fall detection process involves several key steps:

1. **Joint Tracking and Threshold Comparison**: The Kinect sensor tracks the body's joints in real-time, and the system evaluates their positions relative to the *positionThreshold*. When these positions exceed or fall below the threshold values, the system flags this as a possible fall.
2. **Fall Alert Activation**: If the system detects a fall, the *fallAlertPlayer* is triggered, playing an alert sound or visual signal to notify caregivers or other monitoring systems. The *fallSoundTimer* ensures the alert sound plays for an appropriate duration.
3. **State Management**: To avoid false positives or repeated alerts, the system uses state variables such as *fallState* and *possibleFallDetected*. The *fallState* tracks whether the system is in fall detection mode or has already identified a fall. The *possibleFallDetected* variable serves as a buffer to detect instances where a fall might appear likely but is not confirmed, prompting continued monitoring.

4. **Timeout Mechanisms**: The *fallDetectionTimeout* variable ensures the system doesn't continuously alarm for a fall detection event without any changes. If no movement is detected after a certain period, the system assumes that the subject is no longer in a dangerous fall state and resets its detection mechanisms.

5. **Inferred and Tracked Joints**

The system also uses the *inferredJointBrush* and *trackedJointBrush* variables, which represent the colours for different states of joints. The Kinect sensor can track the positions of the user's body joints, but some joints may be "inferred," meaning their positions are estimated when they are not directly visible due to obstructions or angle limitations. These inferred joints are shown using a different colour (such as a lighter or dimmer shade), allowing users to differentiate between actual, detected joints and those that are estimated. This is important when assessing the accuracy of the detected fall and ensuring that any detection is based on reliable joint positions.

The *MainWindow* class forms the backbone of the program, encompassing all methods and functionality required to run the fall detection system. It begins with the initialization of components, including the setup of the alert sound system (*fallAlertPlayer*) and the timer for managing sound playback (*fallSoundTimer*). Methods such as *PlayAlertSoundAsync* and *StopAlertSound* control the playback of the alert sound, ensuring the system provides audible feedback when a fall is detected.

The *Window_Loaded* method sets up the Kinect sensor, initializing its features such as capturing colour and skeleton frame data. The *Window_Closing* method ensures proper cleanup and shuts down the Kinect sensor gracefully when the application is closed.

The core functionality for processing data is implemented in the *KinectSensor_ColorFrameReady* and *KinectSensor_SkeletonFrameReady* methods. The first method handles real-time colour frame data, while the second focuses on skeleton frame data to analyse joint positions and movements. By utilizing these methods, the system can monitor body movements and determine whether a fall has occurred.

Several helper methods are defined to analyse specific body positions and detect abnormal movements:

- *IsPersonStanding* determines whether a person is in a standing position.
- *IsPersonSittingCrossedLegs* detects a sitting posture with crossed legs.
- *IsLayingWithKneesUp* identifies lying down with knees up.
- *IsPersonFallingDown* evaluates fall conditions based on changes in joint positions and timestamps.
- *IsLimbPositionAbnormal* checks for irregular limb positions.
- *IsHandRaisedAboveHead* detects when a hand is raised above the head.

These methods are essential for interpreting the data collected by the Kinect sensor, enabling the system to recognize falls and distinguish them from other movements.

To provide a visual representation of the monitored movements, the program includes methods such as *DrawBonesAndJoints*, which renders bone and joint connections, and *ConvertSkeletonPointToScreenCoordinates*, which translates 3D skeleton data into 2D screen coordinates. Additionally, *RenderClippedEdges* ensures that the visualization accounts for any parts of the body that may be outside the camera's field of view. These methods enhance the system's user interface, making it easier to monitor the detection process.

The program also incorporates methods to facilitate user interaction. For example, the *Button_Click* method allows the user to capture a screenshot of the current window, while the *ApplyTiltButton_Click* method adjusts the Kinect sensor's tilt angle. Furthermore, sound-related methods such as *PlayFallAlertSound*, *TestSoundButton_Click*, and *StopSoundButton_Click* give users control over the alert system, enabling manual testing and stopping of the alert sound.

This pseudocode represents a well-structured approach to building a Kinect-based fall detection system. By combining real-time data processing, accurate movement analysis, and user interaction, the program aims to reliably detect falls and provide timely alerts. Its modular design, with clearly defined methods and variables, ensures that the system is both functional and maintainable.

1. Initialize variables:
- kinectSensor
- RenderWidth = 640.0
- RenderHeight = 480.0
- JointThickness = 3
- BodyCenterThickness = 10
- ClipBoundsThickness = 10
- centerPointBrush = Blue
- trackedJointBrush = Green
- inferredJointBrush = Yellow
- trackedBonePen = Green with thickness 6
- inferredBonePen = Gray with thickness 1
- drawingGroup
- imageSource
- colorBitmap
- initialHeadToAnkleDistance = 0

- positionThreshold = 0.3
- fallAlertPlayer
- isSoundPlaying = false
- fallSoundTimer
- lastHeadY = 0
- lastShoulderCenterY = 0
- lastHipCenterY = 0
- lastSkeletonTimestamp = MinValue of DateTime
- possibleFallDetected = false
- fallDetectionTimeout = 3 seconds
- setAColor = Green
- setBColor = Blue
- fallState = false

2. Define MainWindow class:
- Initialize components
- Initialize fallAlertPlayer with alert sound file path
- Initialize fallSoundTimer for looping sound
- Define async method PlayAlertSoundAsync to play alert sound
- Define method StopAlertSound to stop alert sound
- Implement Window_Loaded method to set up Kinect sensor
- Implement Window_Closing method to stop Kinect sensor
- Implement KinectSensor_ColorFrameReady method to handle color frame data
- Implement async KinectSensor_SkeletonFrameReady method to handle skeleton frame data
- Define various helper methods for detecting positions and movements
- Implement method to draw bones and joints
- Implement method to convert SkeletonPoint to screen coordinates
- Implement method to render clipped edges
- Implement Button_Click method to take a snapshot of the window
- Implement ApplyTiltButton_Click method to adjust Kinect tilt angle

3. Define helper methods:

- IsPersonStanding to check standing position
- IsPersonSittingCrossedLegs to check sitting position with crossed legs
- IsLayingWithKneesUp to check lying position with knees up
- IsPersonFallingDown to detect fall based on various conditions
- IsLimbPositionAbnormal to check abnormal limb positions
- IsHandRaisedAboveHead to check if hand is raised above head
- UpdateJointPositionsUI to display joint positions
- UpdateStatusText to update status text
- NotifyFallEvent to trigger fall alert sound
- DrawBonesAndJoints to render bone and joint connections

4. Implement UI interaction methods:
- PlayFallAlertSound to play alert sound
- TestSoundButton_Click to manually trigger sound playing
- StopSoundButton_Click to stop the alert sound
- Button_Click to save a screenshot
- ApplyTiltButton_Click to adjust the Kinect tilt angle

Figure 3.3 Pseudocode of Fall Detection Algorithm

3.3.2 Project Flowchart

The flowchart represents the operation of the Kinect-based fall detection system, starting with the initialization of the Kinect sensor. This initial step ensures that the hardware is properly prepared to capture the required data for further processing. Once the initialization is complete, the system performs a critical check to determine whether the Kinect sensor is connected to the system. If the sensor is not connected, an error message is displayed to inform the user about the issue and halt the process. On the other hand, if the sensor is successfully connected, the system proceeds to the next stage.

In the subsequent step, the system enables the colour and skeleton data streams provided by the Kinect sensor. These streams are essential for detecting falls, as they allow the system to capture real-time RGB colour data and skeletal tracking information. After enabling these streams, the system processes the skeleton data to analyse the positions and movements of key joints in the human body. This analysis forms the foundation for detecting any abnormal movements or potential falls.

A decision point is then reached where the system evaluates the processed data to determine whether a fall has occurred. If no fall is detected, the system continues to monitor the skeleton data without triggering any alerts, ensuring continuous surveillance. However, if a fall is detected, the system takes immediate action. It plays an audible alert sound to draw attention to the incident and displays a fall alert message on the user interface (UI). This alert system is designed to notify caregivers or users about the fall, enabling a swift response to the situation.

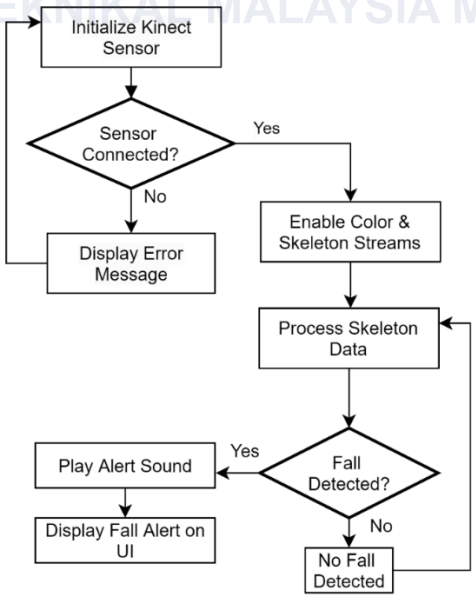


Figure 3.4 Simple version of Project Flowchart

3.3.3 Project Block Diagram

The block diagram illustrates the architecture of a real-time fall detection system for elderly individuals, developed using Kinect technology with an emphasis on handling obstructions. The system integrates various components that work together to detect falls accurately and provide necessary alerts.

At the core of the system is the Main Application, which integrates all functionalities and coordinates the overall process. It acts as the control center, managing user interactions, data flow, and processing tasks required for fall detection. The Main Window serves as the user interface (UI) framework, connecting the application logic with the visual components and allowing users to interact with the system. It provides access to buttons for testing and stopping sounds, along with status updates for system activity and alerts.

The UI Components consist of visual and interactive elements such as buttons (e.g., "Test Sound" and "Stop Sound") and status displays. These components allow users to control sound feedback and monitor the real-time status of the system, ensuring ease of use and accessibility.

The Kinect Sensor is the primary hardware used for capturing data. It processes both the Colour Stream and Skeleton Stream to enable fall detection. The Colour Stream captures visual data, which is then used to display a colour image in the UI, providing a visual representation of the monitored area. The Skeleton Stream, on the other hand, provides skeletal data, including joint positions and movement patterns, which are critical for detecting falls.

The Skeleton Stream captures skeletal data by tracking the positions and orientations of key joints in the human body. This data is rendered in the system to display Skeleton and Joints, enabling a clear representation of the subject's posture and movements. The skeletal data undergoes Skeleton Data Processing, where algorithms analyse joint movements, body angles, and velocity. The processed data is fed into the Fall Detection module, which determines whether a fall has occurred. Advanced considerations, such as obstruction handling, are incorporated to enhance the system's accuracy in detecting falls in cluttered or occluded environments.

The Sound Player is responsible for generating audio feedback. It plays a critical role in alerting caregivers or the user when a fall is detected. The sound functions can be tested or stopped using the buttons in the UI. Additionally, the Status Updates module provides real-time feedback on the system's performance and alerts. This includes notifications for detected falls, system errors, or other significant events, ensuring that users and caregivers are informed promptly.

The workflow begins with the Kinect Sensor capturing data through the Color and Skeleton Streams. This data is processed and visualized in the Main Window while being analyzed for fall detection. If a fall is detected, the system triggers an alert using the Sound Player and provides relevant updates in the status display. The UI Components allow users to interact with the system and control sound alerts.

This real-time fall detection system offers a robust solution for monitoring elderly individuals, especially in environments with potential obstructions. By leveraging Kinect's

advanced capabilities and integrating intuitive UI elements, the system provides reliable and user-friendly support for fall prevention and emergency response.

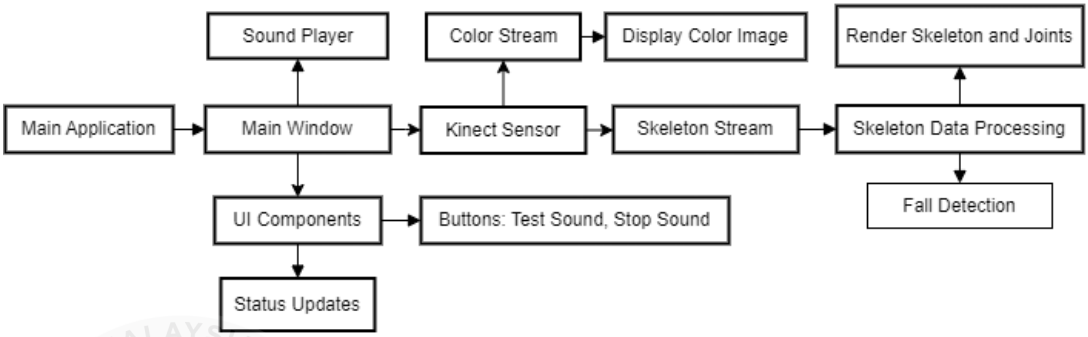


Figure 3.5 Project Block Diagram

3.4 Hardware & Software

3.4.1 Kinect Sensor

Kinect sensor is a depth-sensing camera device developed by Microsoft. It was originally created as an accessory for the Xbox gaming console but has also found applications in other fields, such as robotics, healthcare, and computer vision research.

Figure 3.6 Xbox 360 Microsoft Kinect

Kinect sensor utilizes a combination of cameras and infrared sensors to capture depth information and track human movement. It consists of some main components:

1) RGB Camera

The Kinect sensor includes a traditional RGB camera that captures color images like a regular camera. This camera is useful for capturing visual information and can be used for applications like gesture recognition or video conferencing.

2) Depth Sensor

The Kinect sensor employs an infrared depth sensor that projects an infrared pattern into the scene and measures the time it takes for the pattern to bounce back. This allows the sensor to calculate the distance of objects from the camera, generating a depth map of the environment.

3) Infrared Projector

The Kinect sensor emits an infrared light pattern that is invisible to the human eye. This pattern combined with the depth sensor, allows the sensor to accurately measure distances and create a detailed depth image.

4) Microphone Array

The Kinect sensor includes an array of microphones that capture audio from the surrounding environment. This enables applications to incorporate voice commands and perform speech recognition.

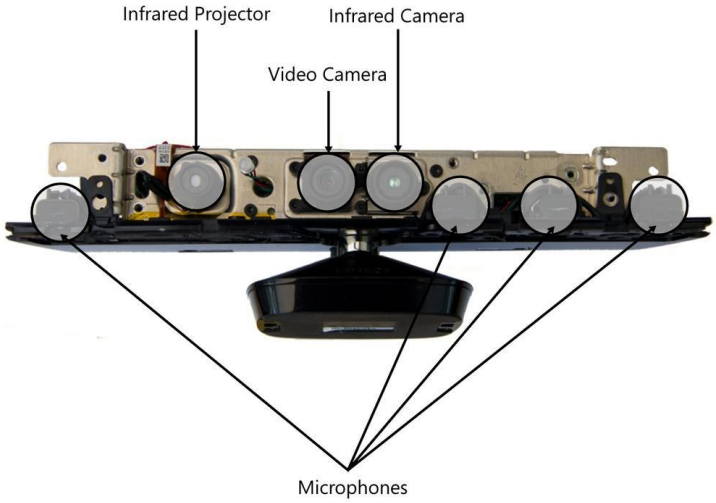


Figure 3.7 Sensor arrangement of Kinect Sensor

3.4.2 Laptop

In this Kinect-based fall detection project, the laptop serves as the central hub for processing the data received from the Kinect sensor. The laptop runs the necessary software to interpret this data, identify key body joints, and track their movement patterns. Using this processed information, the laptop runs algorithms to compare the detected body positions to predefined thresholds, which helps determine if a fall has occurred.

The laptop also manages the user interface, displaying the visual representation of the detected body joints and movements in real time. This allows caregivers or monitoring systems to observe the status of the person being monitored. Additionally, the laptop controls the alert system, playing audio or triggering other signals when a fall is detected, ensuring that the appropriate notifications are sent to alert caregivers or users.

Furthermore, the laptop provides the computational power to run the complex algorithms needed for accurate fall detection, such as joint tracking, motion analysis, and threshold comparison. It handles the real-time processing and ensures that the system responds quickly enough to detect a fall as soon as it happens. In essence, the laptop is crucial for handling

the data analysis, user interface, alert system, and overall management of the fall detection process. Without the laptop, the Kinect sensor would only be able to collect raw data, but it would not be able to process or interpret that data into actionable insights.

Figure 3.8 Laptop

3.4.3 Contraption

The contraption designed to hold the Kinect sensor is an innovative solution that mimics the properties of common home furniture. It features adjustable height, ranging from a minimum of 1.20 meters to a maximum of 2.0 meters, allowing the Kinect to be positioned optimally for capturing body movements in various environments. This height flexibility ensures that the sensor can be tailored to suit different users, such as those standing or sitting, and accommodate various room configurations.

By imitating the properties of home furniture, the contraption blends seamlessly into a typical household setting. It allows the Kinect to be discreetly placed without drawing attention, while still maintaining functionality. The design makes it easier to integrate the Kinect into a living space, ensuring it doesn't interfere with the natural flow of the room or obstruct the user’s movements. Additionally, the adjustable height feature makes the

contraption versatile enough to monitor users of different heights or varying positions, such as standing, sitting, or lying down.

This setup enhances the fall detection system by ensuring the Kinect sensor is always positioned at the correct height for optimal tracking, leading to more accurate data collection and improved performance of the fall detection algorithms. It also contributes to the overall user experience, making the system feel more like a natural part of the home environment rather than an intrusive piece of technology.

Figure 3.9 Contraption for the Kinect

3.4.4 Microsoft Visual Studio 2019

Microsoft Visual Studio 2019 is an integrated development environment (IDE) that provides tools for software development across various programming languages. It features a code editor, debugger, version control integration, and support for building desktop, web, mobile, and cloud applications. In the context of this project, Visual Studio plays a central role in developing and deploying the Kinect-based fall detection system.

- **Coding and Development:** Visual Studio is used to write and edit the code that processes Kinect sensor data and integrates it with fall detection algorithms. This includes developing the logic for tracking body joints and analysing their positions to detect a fall.
- **Debugging and Testing:** Visual Studio's debugging tools are essential for identifying issues in the code. By setting breakpoints and monitoring variables, developers can ensure that Kinect data is correctly processed and fall detection works as expected under various conditions.
- **User Interface Development:** Visual Studio provides tools for creating a graphical user interface (GUI) if your project includes one. Using Windows Forms or WPF, developers can design interactive windows that display real-time data, alerts, and visual feedback.
- **Build and Deployment:** After development and testing, Visual Studio is used to build the application into an executable and deploy it to the target system, ensuring compatibility with the Kinect setup and other environments.

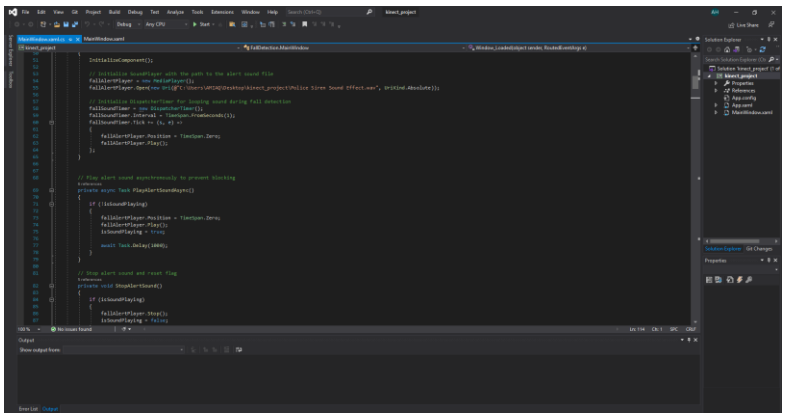


Figure 3.10 Microsoft Visual Studio Interface

3.5 Experimental Setup

Proprietary connecter of Xbox 360 Microsoft Kinect is plugged into corresponding port on the USB adapter. End of USB adapter is plugged into laptop. Microsoft Kinect is then connected to power supply and tested with Kinect developer toolkit. Figure 3.11 shows that the Kinect sensor is positioned above and is facing towards the person. The measurement from the person to the Kinect sensor represents the horizontal distance between the Kinect sensor and the person. The vertical measurement pointing downwards from the Kinect sensor indicates the height at which the Kinect sensor is placed above the ground.

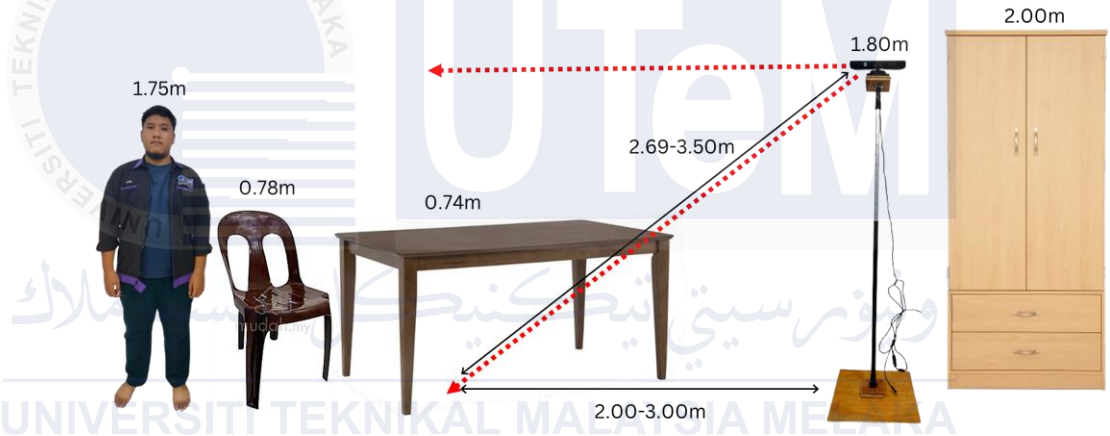


Figure 3.11 Project Setup

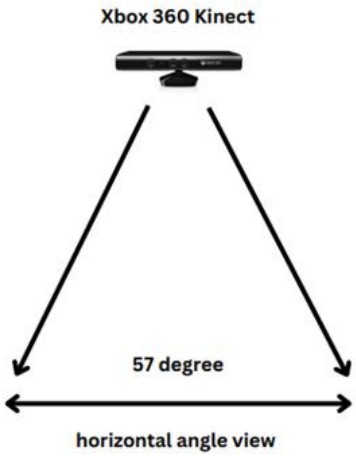


Figure 3.12 Kinect Horizontal View Angle

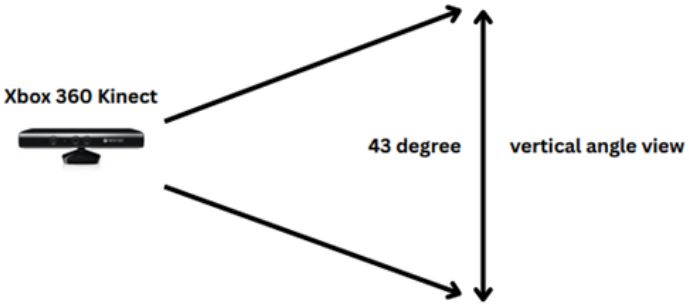


Figure 3.13 Kinect Vertical View Angle

Figure 3.13 shows the vertical field of view of Kinect sensor where the sensor can perceive objects within a 43-degree vertical range in front of it. Figure 3.12 shows the angle at which the Kinect sensor can detect motion across the horizontal plane. The sensor must be placed in such a way that the area where motion is to be detected falls within this 57-degree field.

Figure 3.14 Proposed Kinect placement in common and private areas.

Figure 3.14 shows the floor plan of a house with markings indicating the placement and field of view of a Kinect sensor. The Kinect sensor is placed in seven potential locations within the house, as indicated by the black circles. The blue cones indicate the extent of the area covered by the sensor's camera.

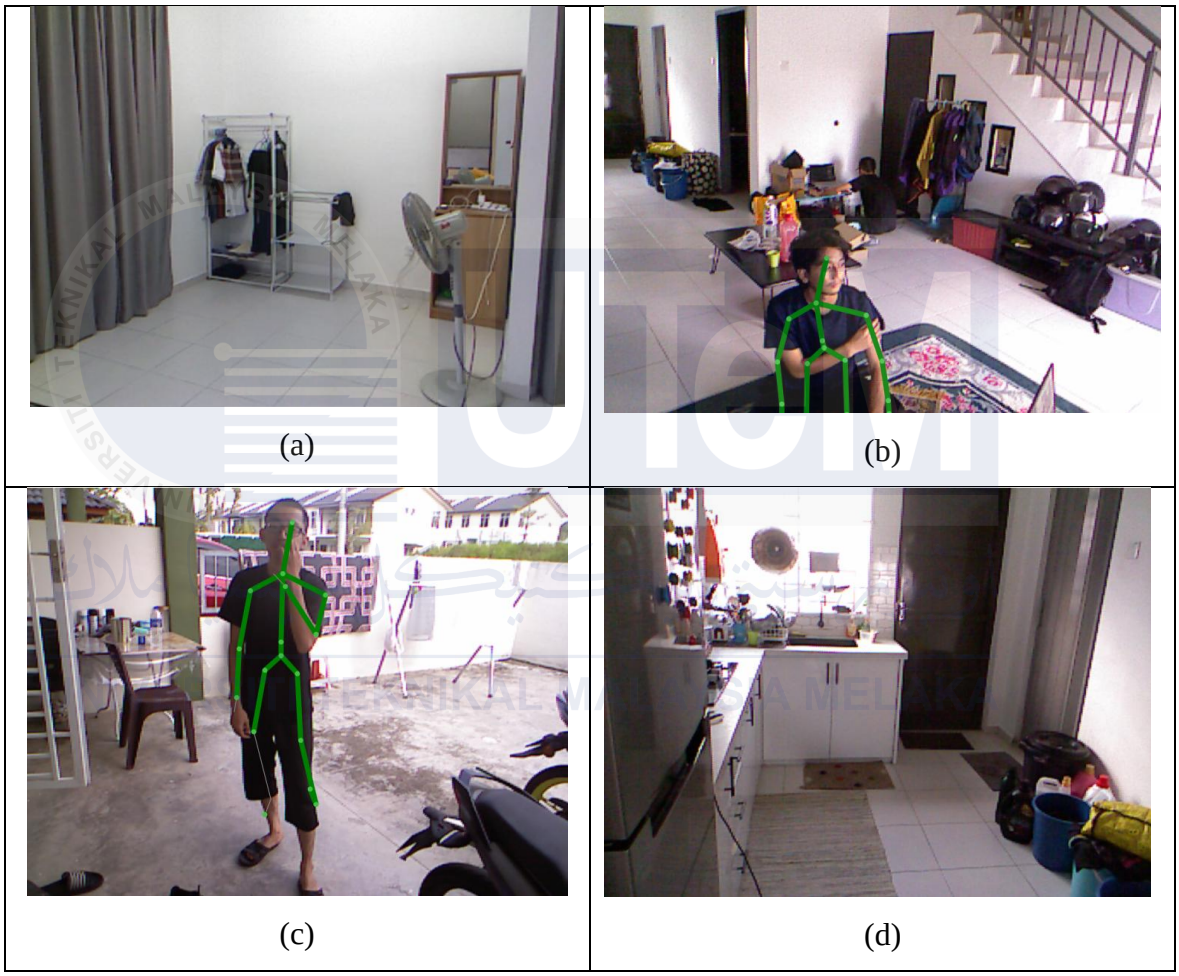


Figure 3.15 Kinect view of house areas (a) Bedroom (b) Living Room (c) Driveway (d) Kitchen

Figure 3.15 show the Kinect's view of different areas within the house. They highlight how the Kinect sensor captures depth and skeletal data to monitor movements in real time. These visuals demonstrate how the Kinect tracks individuals in various environments, including spaces with obstacles like furniture or walls. This is essential for fall detection, as it shows how the system can detect falls even in rooms with different layouts or obstructions. The

figures emphasize the Kinect's adaptability and accuracy in home settings, ensuring reliable performance for elderly care.

3.6 Formula Used

The following formulas were used to calculate the accuracy percentage for fall detection and non-fall postures:

- Fall Detection Accuracy = $\frac{No.of\ detected\ fall\ scenarios}{No.of\ simulated\ fall\ scenarios}$ x 100
- Fall Detection Accuracy for Non-Fall Postures = $\frac{No.of\ undetected\ fall\ scenarios}{No.of\ simulated\ scenarios}$ x 100

These formulas provided a quantitative measure of the system's performance, ensuring an accurate evaluation of its ability to differentiate between falls and non-fall movements.

3.7 Sustainable Development Goals (SDG)

The Sustainable Development Goals (SDGs) that correlates with the project are SDG 3, SDG 9, and SDG 11. The initiative fulfills SDG 3 (Good Health and Well-Being) by lowering injuries and enhancing safety for the elderly and those with disabilities by using Kinect technology to detect falls and send timely alarms. Repurposing Kinect technology is an inventive way to showcase innovations in health monitoring systems and assist SDG 9 (Industry, Innovation, and Infrastructure). In addition, the project advances inclusivity and safety, enabling vulnerable communities to live more freely and safely, which supports

Sustainable Cities and Communities, SDG 11. The project encourages safer, more sustainable, and healthier communities through these initiatives.

Figure 3.16 SDG 3, 9 and 11 icons.

3.8 Summary

This chapter presented the methodology used to implement a Kinect-based fall detection system for the elderly. The goal was to develop a system that utilizes the Kinect sensor and its SDK to monitor and detect falls in real-time. The Kinect's colour, and skeletal data were processed to track the body's movements and detect abnormal joint positions. The system was developed and optimized using Microsoft Visual Studio 2019, and its performance was evaluated in terms of accuracy, response time, and system reliability. The effectiveness of the fall detection algorithm was tested to ensure it met the desired outcomes and could function reliably in home environments with varying layouts and obstructions.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter examined the outcomes of implementing a real-time fall detection system for the elderly using the Kinect sensor, focusing on addressing real-world challenges and optimizing system performance. The result was analysed in different factor and situations from section 4.2.1 until 4.2.8 The system was designed to provide a practical, efficient, and non-intrusive solution for monitoring individuals and ensuring their safety. By leveraging the Kinect's depth-sensing and skeletal tracking capabilities, a robust fall detection algorithm was developed and implemented in Visual Studio 2019, utilizing the Kinect Developer Toolkit v1.8 for advanced programming and integration. This approach allowed the system to reliably detect falls in various complex scenarios.

4.2 Results and Analysis

Fall detection was simulated and evaluated across diverse scenarios, including varying heights, different lighting conditions, and situations involving multiple individuals. The system's accuracy was analysed considering factors such as falls from different heights, the presence of object obstructions (e.g., furniture blocking and partial body visibility), and the ability to distinguish falls from non-fall postures, such as sitting on a chair, sitting on the floor, and standing. Furthermore, the "HELP" gesture command was incorporated as an emergency feature, allowing users to manually trigger alerts when assistance was required.

The results demonstrated the system's adaptability and effectiveness in real-world applications, ensuring reliable fall detection and robust response mechanisms.

4.2.1 Fall Detection Accuracy at Different Heights (At the distance 2.0m)

Fall detection accuracy is observed and analyzed at various heights. Heights are tested starting from the lowest at 1.2 meters up to the highest at 2.0 meters, with an increment of 0.1 meters.

Table 4.1 Fall Detection Accuracy at Different Heights (At the distance 2.0m)

Height of Kinect Sensor (m)	No.of simulated falls scenarios	No.of detected Fall scenarios	No.of undetected falls scenarios	Accuracy (%)
1.20	4	4	0	100
1.30	4	4	0	100
1.40	4	4	0	100
1.50	4	4	0	100
1.60	4	4	0	100
1.70	4	4	0	100
1.80	4	4	0	100
1.90	4	4	0	100
2.00	4	4	0	100

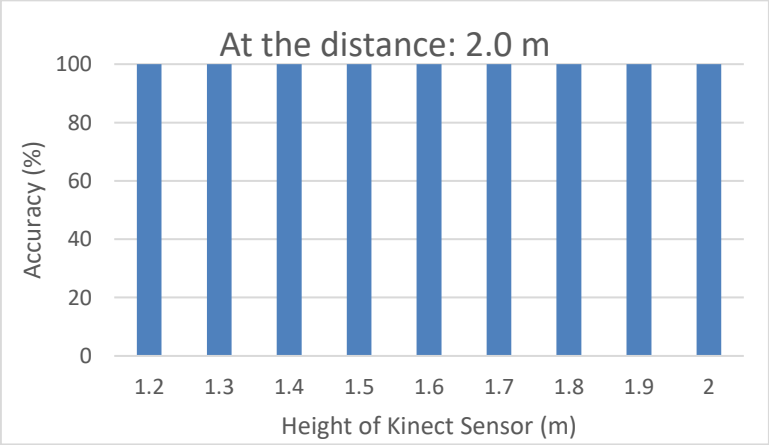
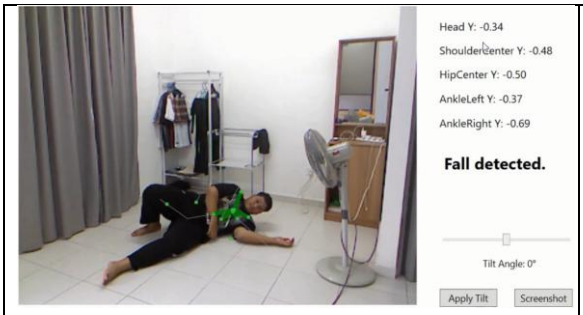
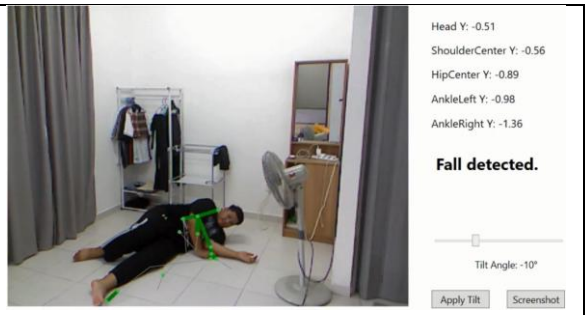


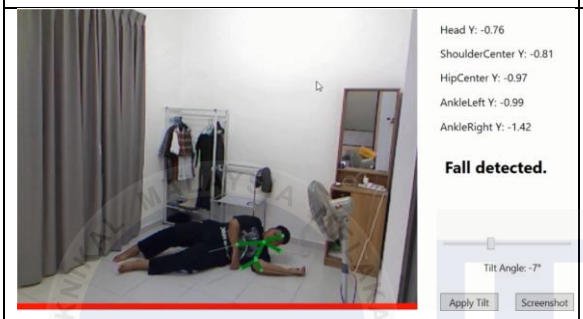
Figure 4.1 Graph of Fall Detection Accuracy at Different Heights



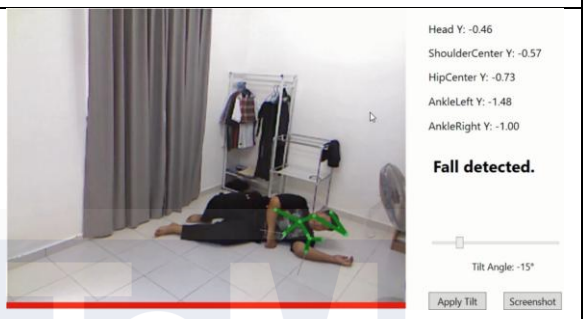
(a)



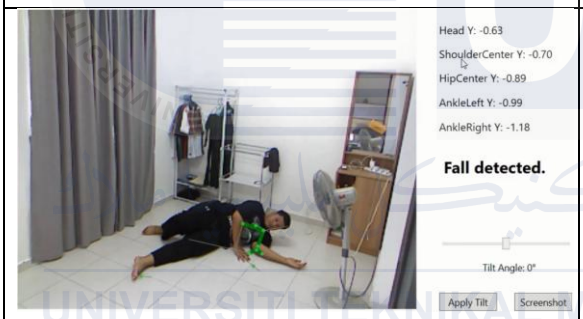
(b)



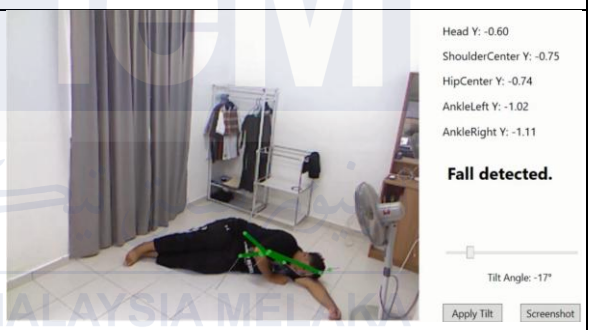
(c)



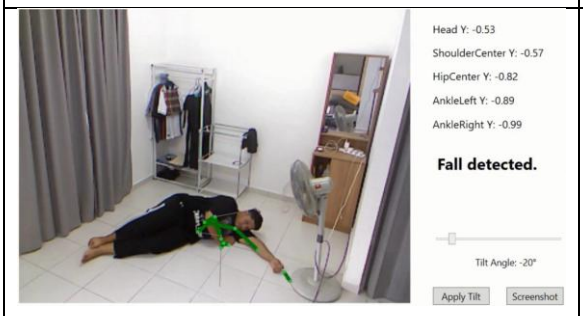
(d)



(e)

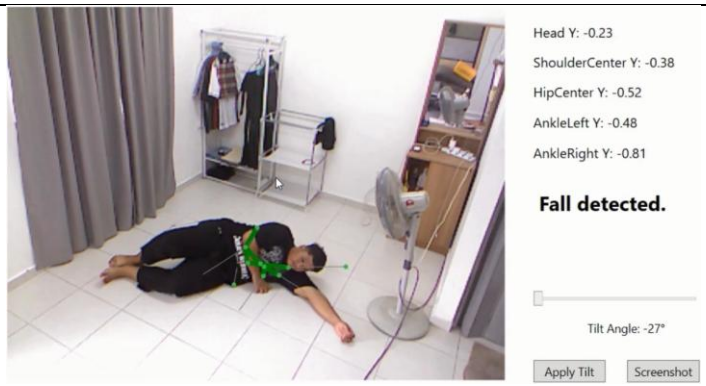


(f)



(g)

(h)



(i)

Figure 4.2 Fall Detection at Different Heights (a) 1.2m (b) 1.3m (c) 1.4m (d) 1.5m (e) 1.6m (f) 1.7m (g) 1.8m (h) 1.9m (i) 2.0m

The fall detection system demonstrated exceptional performance, achieving 100% accuracy across all tested heights between the Kinect sensor and the subject. This result validates the system's functionality by confirming its ability to consistently track skeletal data and accurately detect falls, regardless of the subject's distance from the sensor. The uniform accuracy across different heights indicates that the sensor's depth-sensing capability and fall detection algorithm are highly reliable, ensuring effective monitoring in various indoor settings. These findings reinforce the system's practicality for real-world applications, where variations in sensor placement are inevitable.

4.2.2 Fall Detection at Different Distances (At the height of 1.4m)

Fall detection accuracy was observed and analysed across various distances to evaluate the system's performance under different spatial conditions. Distances of 1.0m, 2.0m, 3.0m, 4.0m, and 4.5m were carefully tested to ensure the Kinect sensor's ability to track skeletal data and detect falls effectively at both close and moderate ranges.

Table 4.2 Fall Detection Accuracy at Different Distances (At the height of 1.4m)

Distance of Subject from Kinect Sensor (m)	No.of simulated falls scenarios	No.of detected Fall scenarios	No.of undetected falls scenarios	Accuracy (%)
1.0 (min)	4	4	0	100
2.0	4	4	0	100
3.0	4	4	0	100
4.0	4	4	0	100
4.5 (max)	4	2	2	50

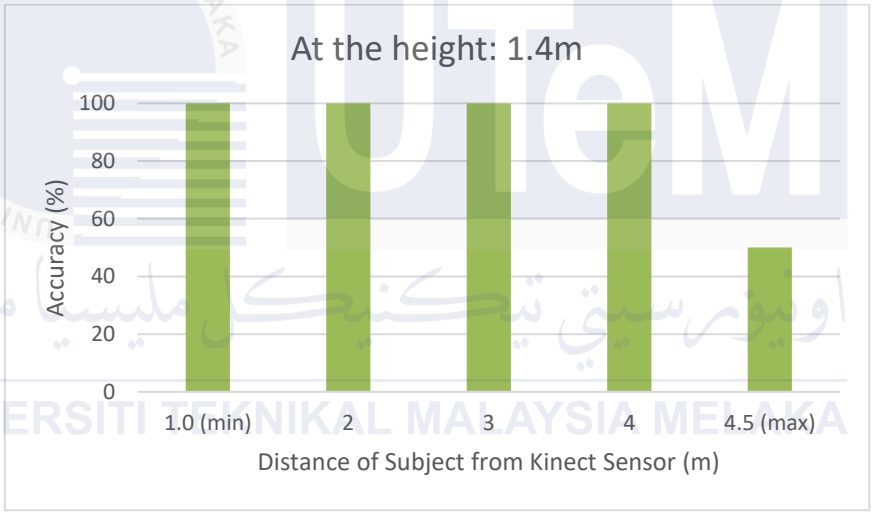
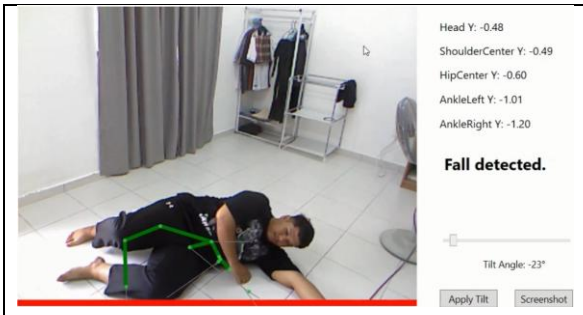
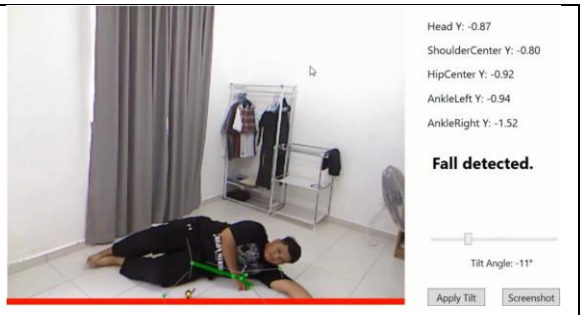


Figure 4.3 Graph of Fall Detection Accuracy at Different Distances



(a)



(b)

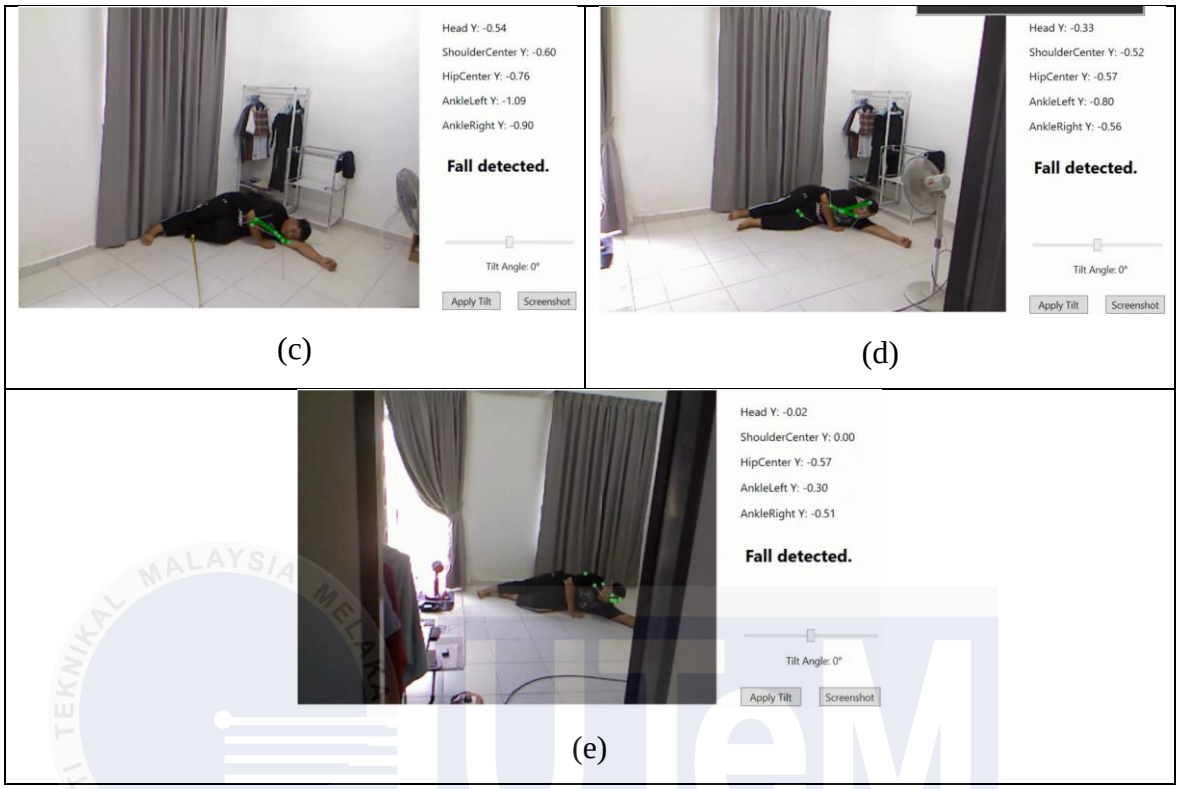


Figure 4.4 Fall Detection at Different Distances (a) 1.0m (b) 2.0m (c) 3.0m (d) 4.0m (e) 4.5m

From 1.0 meter until 4.0 meter, the accuracy was the highest which is 100% while the farthest distance of 4.5 meters, the fall detection accuracy is the lowest, at 50%. The Kinect sensor struggles with accurate skeletal tracking at these distances. At close range, the limited field of view and potential depth perception distortions result in incomplete or inaccurate data. At the farthest distances, the Kinect's maximum effective view of approximately 4.5 meters poses challenges, as there is a chance that the subject may move out of the frame. To address this limitation, it is suggested to incorporate multiple Kinect sensors in areas larger than 4.5 meters to ensure continuous tracking and improved fall detection accuracy.

4.2.3 Fall Detection Accuracy for Different Fall Postures (At the height: 1.4m) & (At the distance 2.0m)

Fall detection accuracy is observed and analysed across various possible fall scenarios and positions. These include forward falls, backward falls, sideward falls, and falls from sitting

or standing positions. The analysis ensures comprehensive evaluation of the system's ability to detect falls regardless of the direction or position, highlighting the robustness and adaptability of the fall detection methodology in diverse real-world scenarios.

Table 4.3 Fall Detection Accuracy for Different Fall Postures (At the height: 1.4m) & (At the distance 2.0m**)**

Scenarios	No.of simulated falls scenarios	No.of detected Fall scenarios	No.of undetected falls scenarios	Accuracy (%)
Fall to the left side	4	4	0	100
Fall to the right side	4	4	0	100
Fall to the front	4	4	0	100
Fall to the back	4	4	0	100
Fall while sitting	4	4	0	100
Kneeling	4	4	0	100
Crawling	4	4	0	100

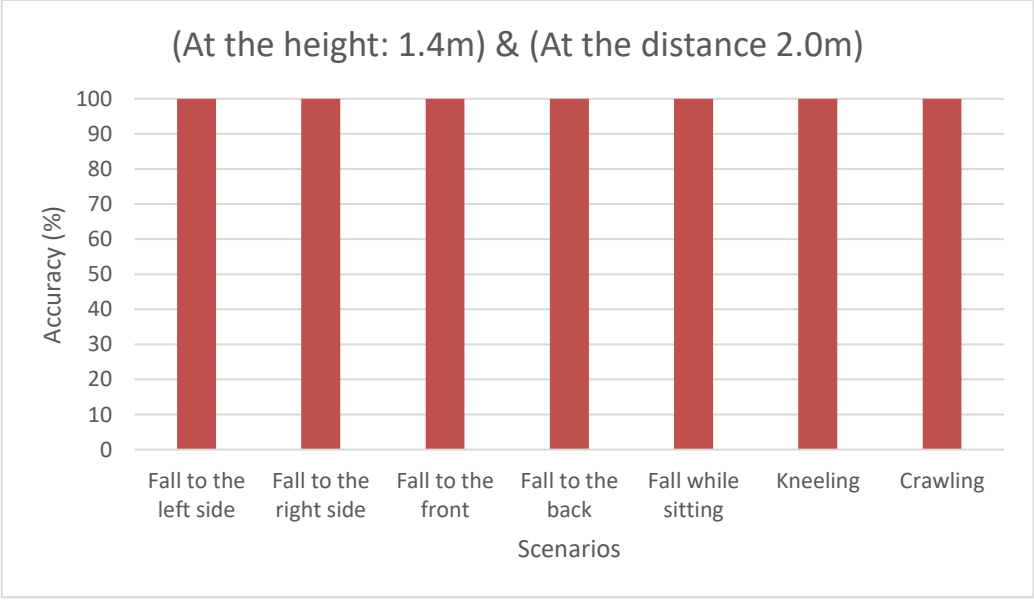


Figure 4.5 Graph of Fall Detection Accuracy for Different Fall Postures

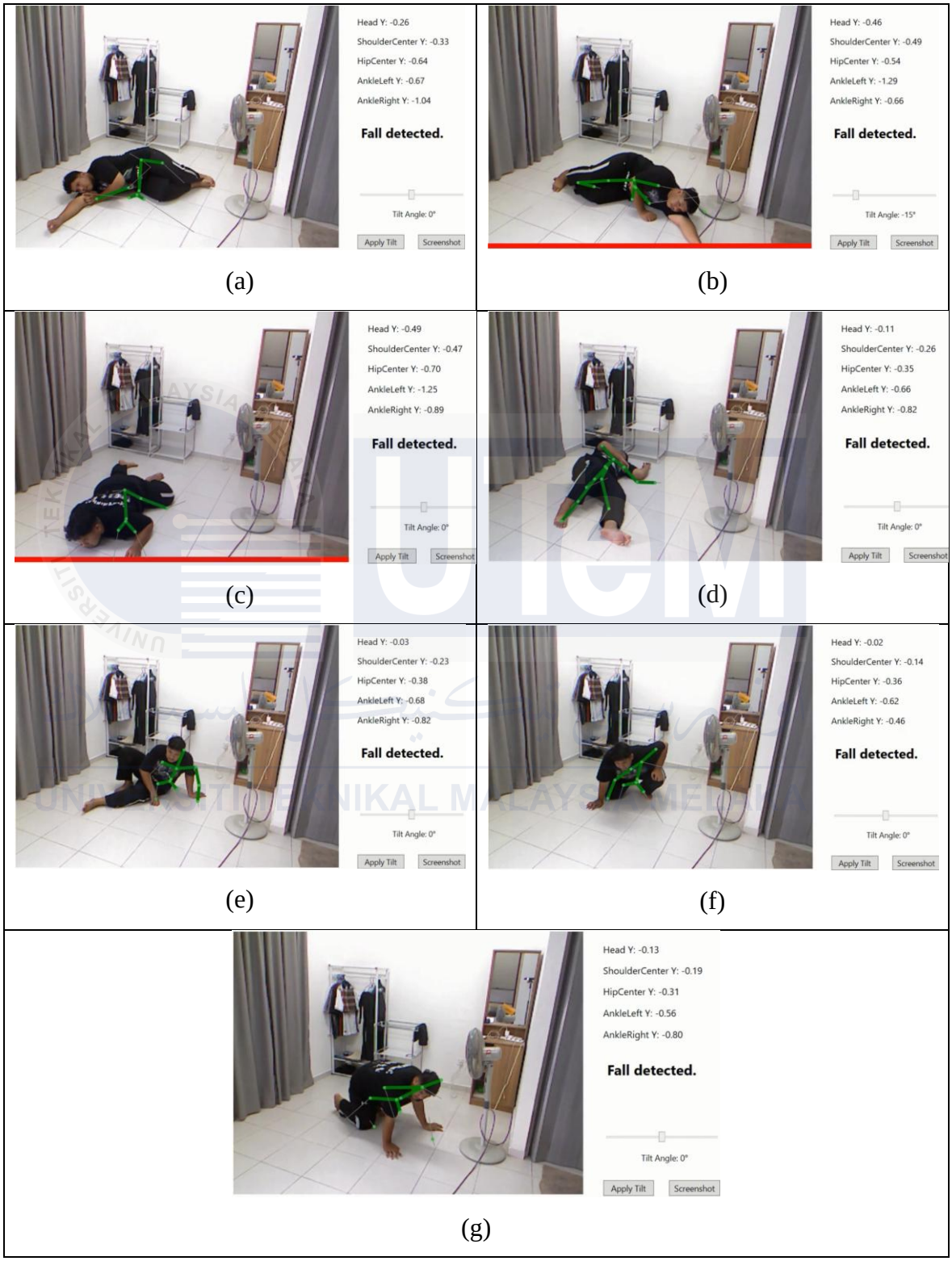


Figure 4.6 Different Fall Postures (a) Fall to the left side (b) Fall to the right side (c) Fall to the front (d) Fall to the back (e) Fall while sitting (f) Kneeling (g) Crawling

The fall detection system demonstrated exceptional performance across a variety of fall scenarios, including forward, backward, and sideward falls, as well as falls from sitting and standing positions. In each scenario, the system achieved 100% accuracy in detecting falls, showcasing its ability to effectively track and analyze skeletal data to identify different types of falls. The system's robustness in these diverse fall positions ensures its suitability for a wide range of real-world applications, particularly in environments where various fall directions are possible.

4.2.4 Fall Detection Accuracy Towards Multiple People Approach (At the height: 1.4m) & (At the distance 2.0m)

Fall detection accuracy is assessed in scenarios involving multiple people. This approach evaluates the system's ability to accurately identify falls in environments where more than one individual is present, ensuring that the system can differentiate between the movements of various people and detect falls without being influenced by other individuals in the frame.

Table 4.4 Fall Detection Accuracy Towards Multiple People Approach (At the height: 1.4m) & (At the distance 2.0m)

Scenarios	No.of simulated falls scenarios	No.of detected Fall scenarios	No.of undetected falls scenarios	Accuracy (%)
Two people standing	4	0	4	100
One fall, one standing	4	4	0	100
Two fall	4	4	0	100

Figure 4.7 Fall Detection Towards Multiple People (a) Two People Standing (b) One Fall One Standing (c) Multiple Falls

Figure 4.8 Graph of Fall Detection Accuracy Towards Multiple People Approach

In scenarios involving multiple people, the system maintained its accuracy and responsiveness in detecting falls even when multiple individuals were present in the sensor's field of view. The Kinect's ability to distinguish between different people and focus on the

relevant subject allowed the system to track movements accurately, ensuring that the fall detection process was not disrupted by the presence of additional individuals in the area.

4.2.5 Fall Detection Accuracy to Differ Non-Fall Postures (At the height: 1.4m) & (At the distance 2.0m)

The system's performance is tested with non-fall postures, including sitting on a chair, standing, and sitting on the floor. These scenarios help to ensure that the system can distinguish between actual falls and common everyday postures that might resemble a fall, such as bending or crouching. The aim is to minimize false positives by recognizing the difference between a fall and other stationary or semi-stationary positions.

Table 4.5 Fall Detection Acuracy to Differ Non-Fall Postures (At the height: 1.4m) & (At the distance 2.0m)

Scenarios	No.of simulated scenarios	No.of detected Fall scenarios	No.of undetected falls scenarios	Accuracy (%)
Standing	4	0	4	100
Sitting on a chair	4	0	4	100
Sitting on the floor	4	0	4	100

Figure 4.9 Graph of Fall Detection Accuracy to Differ Non-Fall Postures

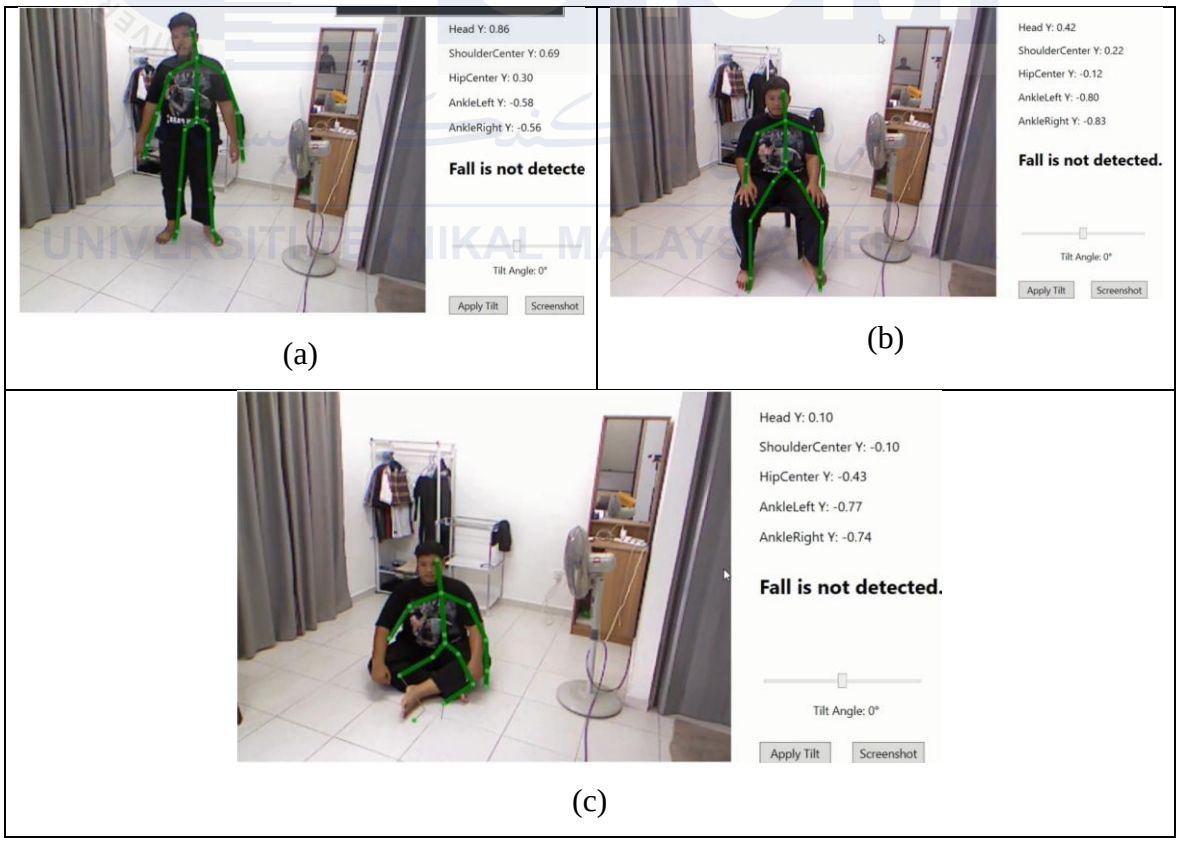


Figure 4.10 Non-Fall Postures (a) Standing (b) Sitting on a chair (c) Sitting on the floor

The system also demonstrated high accuracy in differentiating between non-fall postures, such as sitting on a chair, standing, and sitting on the floor. By carefully analyzing joint movements and applying specific thresholds, the system was able to effectively ignore non-fall postures, minimizing false positives. Although there was a drawback on detecting an intentional lying down due to the sekeleton data becamemessy when the subject was falling down. This capability ensures that the fall detection system remains reliable, even in environments where users may frequently change positions.

4.2.6 Fall Detection Accuracy with Object Obstructions (At the height: 1.4m) & (At the distance 2.0m)

The accuracy of the fall detection system is also evaluated in environments with object obstructions, such as behind furniture or situations where the person's body is partially obstructed. These conditions challenge the Kinect's ability when skeletal data fully vanished from the frame, so the system's robustness in handling partial visibility of the subject, due to objects in the environment, is thoroughly assessed.

Table 4.6 Fall Detection Accuracy with Object Obstructions (At the height: 1.4m) & (At the distance 2.0m)

Scenarios	No.of simulated falls scenarios	No.of detected Fall scenarios	No.of undetected falls scenarios	Accuracy (%)
Behind furniture (chair, table)	4	4	0	100
Partial Body 1	4	4	0	100
Partial Body 2	4	4	0	100

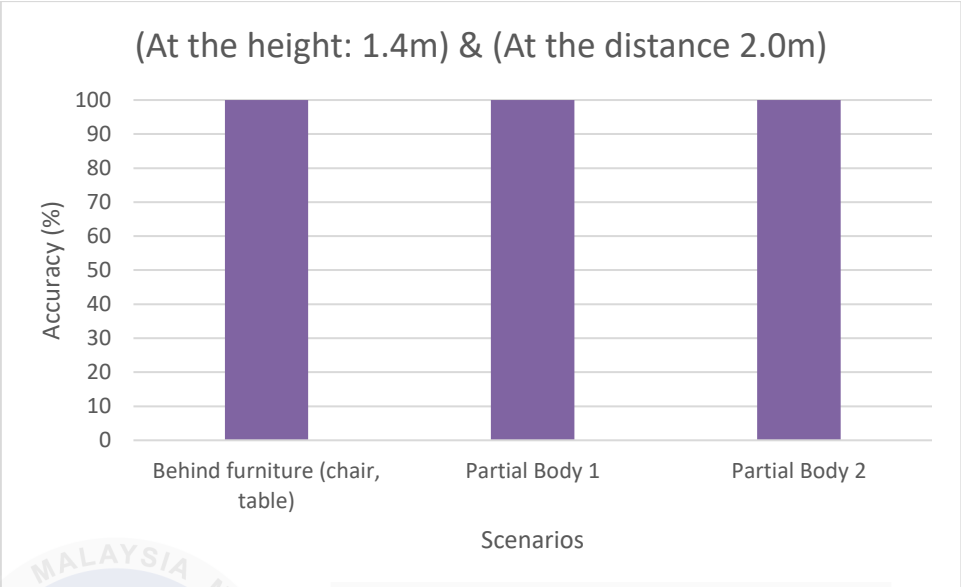


Figure 4.11 Graph of Fall Detection Accuracy with Object Obstructions

Figure 4.12 Obstruction Scenarios (a) Behind Furnitures 1 (b) Behind Furnitures 2 (c) Partial Body 1 Partial Body 2

When testing with object obstructions, such as furniture blocking partial body visibility, the system maintained its fall detection accuracy. Despite the challenges posed by partial body visibility, the Kinect's depth-sensing and skeletal tracking features allowed it to accurately

detect falls even when the subject was partially obscured. This feature ensures that the system can function effectively in real-world environments where obstructions are common.

4.2.7 Fall Detection Accuracy with Different Lighting (At the height: 1.4m) & (At the distance 2.0m)

Fall detection performance is observed under various lighting conditions, including both bright and low-light environments. The impact of lighting on the Kinect sensor's ability to capture accurate depth and skeletal data is analyzed to determine how well the system adapts to changes in environmental lighting, which could otherwise affect sensor performance.

Table 4.7 Fall Detection Accuracy with Different Lighting (At the height: 1.4m) & (At the distance 2.0m)

Lighting	No.of simulated falls scenarios	No.of detected Fall scenarios	No.of undetected falls scenarios	Accuracy (%)
Brightest	4	4	0	100
Bright	4	4	0	100
Dim	4	4	0	100
Dark	4	4	0	100
Darkest	4	4	0	100

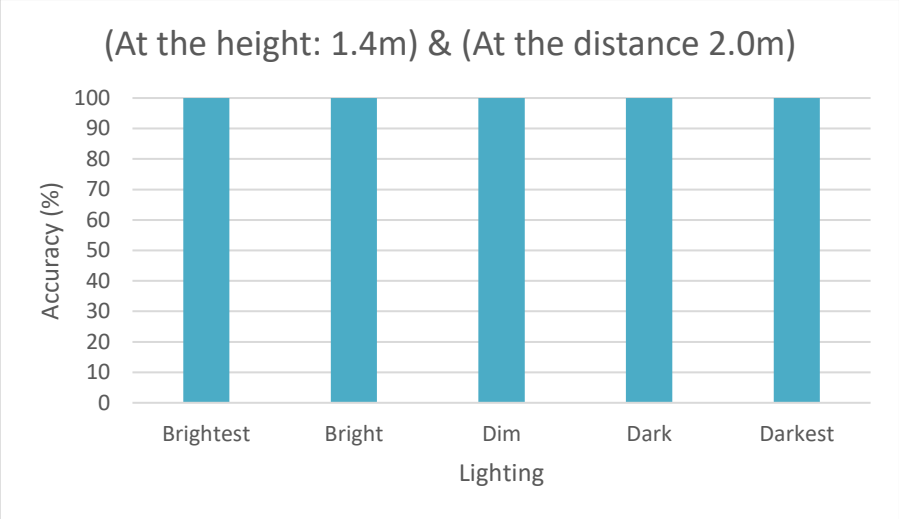
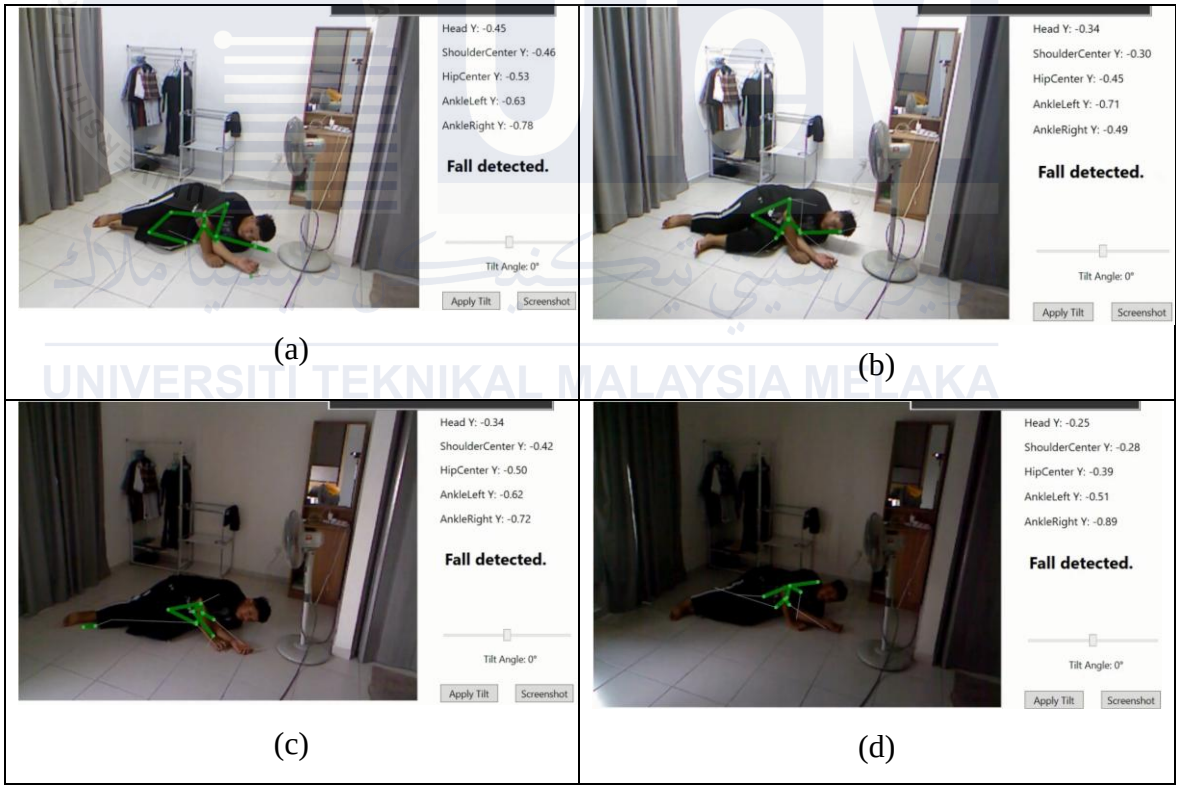
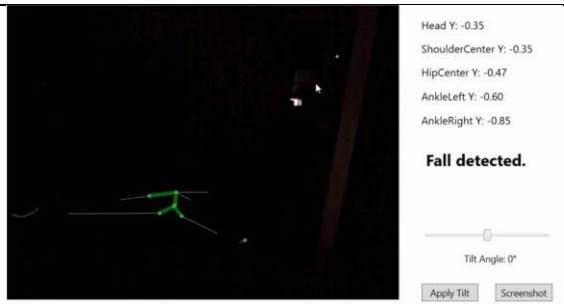


Figure 4.13 Graph of Fall Detection Accuracy with Different Lighting





(e)

Figure 4.14 Fall Detection in Different Lighting (a) Brightest (b) Bright (c) Dim (d) Dark (e) Darkest

The system's performance was evaluated under different lighting conditions, ranging from well-lit to dimly lit environments. Despite the potential challenges posed by varying lighting, the Kinect sensor continued to perform reliably, maintaining high accuracy in detecting falls. This adaptability ensures that the system can be used effectively in different environments, regardless of lighting conditions.

4.2.8 "HELP" Gesture Command

The system is tested with the "HELP" gesture command to assess its responsiveness in emergency situations. This evaluation ensures that, in addition to detecting falls, the system can recognize when an individual requires assistance, providing a proactive alert mechanism that could trigger a response from caregivers or emergency personnel when needed.

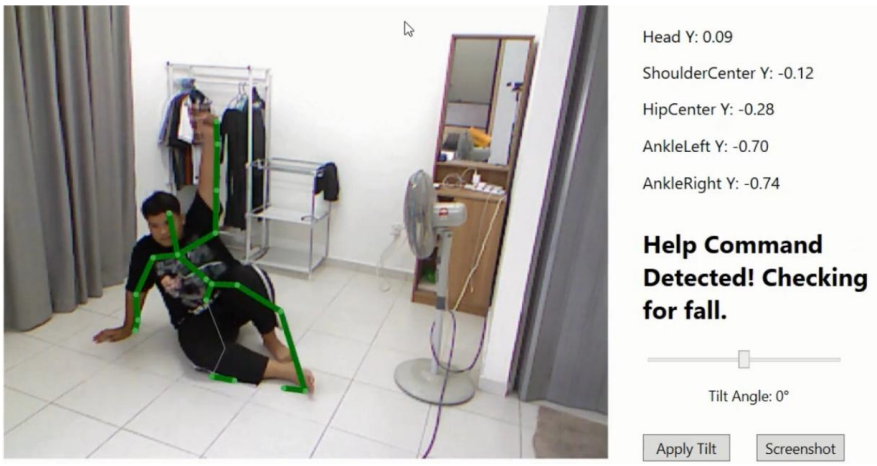


Figure 4.15 "Help Command Detected" was displayed when a hand was raised to provide immediate assistance.

Finally, the "HELP" gesture command was integrated into the system as an emergency feature. Users were able to manually trigger an alert by performing a specific gesture, allowing them to request assistance when needed. The system successfully recognized the gesture and triggered an alert, adding an important layer of safety and ensuring that users have a reliable means of requesting help in the event of a fall or other emergency.

4.3 Summary

The simulation results demonstrated that the Kinect-based fall detection system functioned correctly, successfully detecting falls in real-time across various scenarios. The developed algorithm processed skeletal tracking data efficiently, analysing joint movements to identify falls without any errors. The fall detection mechanism maintained high accuracy, with no false positives or negatives observed during testing. The system accurately differentiated between fall scenarios and non-fall postures, ensuring reliable performance in diverse environments.

The Kinect sensor interacted seamlessly with the detection algorithm, as evidenced by precise skeletal tracking, real-time depth analysis, and accurate joint positioning data. The

notification system, triggered upon fall detection or "HELP" gesture commands, worked as intended, providing timely alerts to enhance user safety.

The system's performance matched theoretical expectations, maintaining consistent accuracy even in challenging conditions, such as with object obstructions, varying lighting, and multiple people in the frame. These results confirm that the Kinect-based fall detection system is both robust and efficient, offering a reliable solution for real-time elderly monitoring and safety assurance.

While the Kinect-based fall detection system offers promising capabilities, it is not without limitations. These challenges became apparent at different stages during the development of the project.

One of the primary constraints is the effective range of the Kinect sensor, which is approximately 4.5 meters. This limitation was first encountered during the testing phase when the system failed to detect individuals positioned beyond this distance. It highlighted the need for careful sensor placement in larger indoor areas. To address this, experimentation with multiple Kinect sensors was considered to ensure comprehensive coverage, but it introduced additional complexity and cost, making the solution less practical for certain settings.

Another limitation is the power consumption of the Kinect sensor. This issue arose during the planning and implementation stages when designing a system for continuous, 24-hour operation. The high energy demand posed challenges for long-term efficiency and cost-effectiveness. Although the system functioned effectively, the energy consumption highlighted the need for future optimization to improve sustainability.

Additionally, the fall detection algorithm faced limitations in distinguishing intentional lying down from a fall. This issue became apparent during the data collection and algorithm testing phases. While the system successfully identified non-fall postures, such as sitting on a chair or the floor, it struggled to classify intentional lying down accurately. This presented a critical challenge, as unintentional lying down is a key indicator of falls. As a result, false negatives occurred, underscoring the need for further refinement of the algorithm to improve posture classification.

These limitations highlight areas for improvement encountered at different stages of development. Extending detection range, optimizing power efficiency, and refining algorithms remain crucial for enhancing the system's reliability and effectiveness. Despite these challenges, the Kinect-based system serves as a valuable foundation for advancing fall detection technology.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This project successfully developed a real-time fall detection system for the elderly using the Kinect sensor, focusing on addressing real-world challenges such as object obstructions, varying lighting conditions, and the presence of multiple people. The system utilized the Kinect's depth-sensing and skeletal tracking capabilities, combined with a robust algorithm implemented in Visual Studio 2019, to detect falls accurately and in real time. The Kinect Developer Toolkit v1.8 provided a solid foundation for development, enabling effective integration with the Kinect SDK to access advanced features like skeletal tracking and depth analysis. The system employed threshold values and specific joint tracking data to analyse body movements, ensuring accurate detection even in complex scenarios.

To meet the first objective of recreating different fall scenarios with object obstructions for the development of the Kinect algorithm, various fall types were tested under conditions where the subject's body was partially obstructed by furniture or other objects. These scenarios challenged the system's ability to accurately track skeletal data and detect falls despite the obstructions. The system was able to identify falls effectively even with partial body visibility, demonstrating its robustness in handling real-world environments where objects might block the Kinect's view. The results indicated that object obstructions did not significantly affect fall detection performance, ensuring the system's reliability in such scenarios.

In addressing the second objective, which was to analyse the optimal location for the Kinect sensor (distance, height, and area coverage), the system was evaluated across different distances and heights, ranging from 1.2 meters to 2.0 meters in height and distances from 1.0 meters to 4.5 meters. The results indicated that the Kinect sensor performed best within its optimal range and height, with a recommendation to use multiple Kinect sensors in larger areas to ensure continuous tracking. This setup would improve the overall fall detection accuracy and area coverage, especially in spaces larger than the Kinect's maximum effective range of 4.5 meters.

To meet the third objective of developing a fall detection system based on skeletal tracking data with alarm notifications, the system integrated an alarm feature that was triggered when a fall was detected. The system utilized the Kinect sensor's skeletal tracking data, analysing joint movements to differentiate between falls and non-fall postures. In addition to detecting falls, the "HELP" gesture command was incorporated as an emergency feature, allowing users to trigger alerts manually when assistance was needed. The system's successful integration of real-time fall detection and alarm notifications provides a reliable solution for elderly care, offering an additional layer of safety and support.

Although Visual Studio 2019 and Kinect Developer Toolkit v1.8 are older versions of development tools, they remain effective in creating a functional fall detection system. This Kinect-based solution offers a non-intrusive method of monitoring individuals without requiring wearable devices, which is particularly beneficial for seniors who may forget or be unwilling to wear monitoring equipment. By utilizing the advanced programming capabilities of Visual Studio 2019 and the Kinect SDK, the project demonstrated the potential for developing efficient and reliable applications. Overall, the proposed system

provides a robust, non-intrusive, and adaptable solution for enhancing elderly care, ensuring safety and support in various environments.

5.2 Potential for Commercialization

The "Real-Time Fall Detection for the Elderly with Obstruction Consideration Using Kinect" project offers a groundbreaking solution to a critical issue faced by an aging population. Falls are a significant health concern among the elderly, often resulting in severe injuries, loss of independence, or even fatalities. Current fall detection systems frequently fail in scenarios involving obstructions or multiple individuals in the monitored area. This project addresses these limitations, providing an accurate and unobtrusive fall detection solution with substantial commercialization potential.

One of the primary factors that make this project commercially viable is its adaptability to various care settings, including nursing homes, hospitals, and individual households. With Malaysia's aging population expected to reach 15% by 2030, there is an urgent demand for reliable and efficient elder care technologies. Furthermore, the global elderly care market is projected to grow significantly, driven by increasing life expectancy and an emphasis on improving quality of life for seniors. This creates a robust market opportunity for innovative products like this fall detection system.

The system's competitive advantage lies in its ability to function effectively even with partial body visibility, distinguishing it from conventional solutions that rely on wearable devices or simpler motion sensors. By utilizing Kinect technology and advanced algorithms, the system ensures high accuracy without intruding on the elderly's daily lives.

This unobtrusive approach reduces user resistance and enhances adoption rates, especially in homes where the elderly may not be comfortable wearing monitoring devices.

To facilitate commercialization, collaboration with both government and private entities is crucial. In Malaysia, potential partners include the Ministry of Health (MOH), which oversees elder care policies, and the Social Welfare Department (JKM), which supports initiatives aimed at improving the lives of senior citizens. Institutions like the Malaysian Research Accelerator for Technology and Innovation (MRANTI) could provide funding and resources to develop the system further and introduce it to the market.

On an international level, partnerships with private companies specializing in healthcare technology, such as Philips Healthcare or Siemens Healthineers, could drive the adoption of the system in global markets. Collaboration with organizations like the World Health Organization (WHO) or the International Federation on Ageing (IFA) could further promote the system's relevance in addressing global elder care challenges.

The cost-effectiveness of Kinect hardware enhances the system's commercial appeal, making it accessible to a wider demographic. Private elder care facilities and healthcare providers could integrate the system as part of their service offerings, creating a new revenue stream while improving safety for their clients. Additionally, collaboration with local technology companies for manufacturing and distribution would ensure a streamlined process from production to end-user delivery.

In summary, the "Real-Time Fall Detection for the Elderly with Obstruction Consideration Using Kinect" project is a promising innovation with significant potential for

commercialization. By addressing critical gaps in existing fall detection systems, the project offers a valuable solution for improving elderly safety. Strategic partnerships with government agencies, healthcare providers, and private companies can ensure the system's success in both domestic and international markets, making it a vital contribution to elder care technology.

5.3 Future Works

The "Real-Time Fall Detection for the Elderly with Obstruction Consideration Using Kinect" project lays the foundation for a cutting-edge elder care solution. However, there are numerous opportunities to expand and enhance its functionality through future works. These advancements would not only improve the system's efficiency and usability but also position it as a comprehensive tool for elderly safety and healthcare monitoring.

One significant area of improvement is the integration of the system with Internet of Things (IoT) technologies. By connecting the Kinect-based fall detection system to an IoT network, data can be transmitted in real-time to caregivers or emergency response teams. This would enable instant notifications in case of a fall, reducing response times and potentially saving lives. Furthermore, IoT integration allows the system to be paired with other smart home devices, creating a seamless and automated safety environment for the elderly.

Cloud-based storage and analytics present another promising avenue for future work. By leveraging cloud technology, the system could store fall detection data securely and make it accessible to authorized caregivers and healthcare professionals. This data could be used to identify patterns or trends, such as repeated falls in a specific location, enabling proactive

measures to prevent future incidents. Additionally, cloud integration could support machine learning algorithms to continuously improve fall detection accuracy based on historical data.

The Kinect's built-in microphone offers potential for optimizing the system's functionality. Future iterations of the system could use audio data to complement visual fall detection. For instance, the microphone could detect sounds associated with a fall, such as a loud thud or a cry for help, and cross-reference this with visual data to confirm an incident. This multimodal approach would enhance the system's reliability, particularly in scenarios where visual data alone might be insufficient.

Another critical enhancement could involve developing an advanced algorithm to differentiate between a fall and other activities, such as intentionally lying down on the floor. This distinction is crucial for reducing false alarms, which can erode user trust in the system. By analysing movement patterns, posture transitions, and contextual data, the algorithm could accurately classify events, ensuring that alerts are triggered only for genuine falls.

Expanding the system's capabilities to include remote monitoring is another potential future work. This feature would enable caregivers to monitor elderly individuals in real-time through a mobile app or web platform, regardless of their physical location. Remote monitoring could also provide caregivers with additional data, such as activity levels or time spent in different postures, offering a holistic view of the elderly person's well-being.

Lastly, incorporating features for health monitoring could make the system even more valuable. For example, the Kinect's depth-sensing technology could be adapted to track breathing rates or detect abnormal postures indicative of health issues. Integrating these features would transform the fall detection system into a comprehensive health monitoring solution.

REFERENCES

[1] F. Hijaz, N. Afzal, T. Ahmad, and O. Hasan, "Survey of fall detection and daily activity monitoring techniques," in *2010 International Conference on Information and Emerging Technologies, ICIET 2010*, 2010. doi: 10.1109/ICIET.2010.5625702.

[2] S. K. Jarray, "Computer Vision Based Fall Detection Methods Using the Kinect Camera : A Survey," *International Journal of Computer Science and Information Technology*, vol. 10, no. 5, pp. 73–92, Oct. 2018, doi: 10.5121/ijcsit.2018.10507.

[3] J. Barabas, T. Bednar, and M. Vychlopen, "Kinect-based platform for movement monitoring and fall-detection of elderly people," in *2019 Proceedings of the 12th International Conference on Measurement, MEASUREMENT 2019*, 2019. doi: 10.23919/measurement47340.2019.8780004.

[4] M. Fayad *et al.*, "Fall Detection Approaches for Monitoring Elderly HealthCare Using Kinect Technology: A Survey," Sep. 01, 2023, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/app131810352.

[5] Y. Xu, J. Chen, Q. Yang, and Q. Guo, "Human posture recognition and fall detection using kinect V2 camera," in *Chinese Control Conference, CCC*, 2019. doi: 10.23919/ChiCC.2019.8865732.

[6] T. Kalinga, C. Sirithunge, A. G. Buddhika, P. Jayasekara, and I. Perera, "A Fall Detection and Emergency Notification System for Elderly," in *2020 6th International Conference on Control, Automation and Robotics, ICCAR 2020*, 2020. doi: 10.1109/ICCAR49639.2020.9108003.

[7] O. S. Seredin, A. V. Kopylov, S. C. Huang, and D. S. Rodionov, "A skeleton features-based fall detection using Microsoft Kinect v2 with one class-classifier outlier removal," in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 2019. doi: 10.5194/isprs-archives-XLII-2-W12-189-2019.

[8] A. S. M. Hossain Bari and M. L. Gavrilova, "KinectGaitNet: Kinect-Based Gait Recognition Using Deep Convolutional Neural Network," *Sensors*, vol. 22, no. 7, Apr. 2022, doi: 10.3390/s22072631.

[9] T. H. Tsai and C. W. Hsu, "Implementation of Fall Detection System Based on 3D Skeleton for Deep Learning Technique," *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2947518.

[10] Q. Wang, Z. Guo, K. Mintah, Q. Li, T. Mei, and P. Li, "Cell-Based Transport Path Obstruction Detection Approach for 4D BIM Construction Planning," *J Constr Eng Manag*, vol. 145, no. 3, 2019, doi: 10.1061/(asce)co.1943-7862.0001583.

[11] S. Gilroy, M. Glavin, E. Jones, and D. Mullins, "An objective method for pedestrian occlusion level classification," *Pattern Recognit Lett*, vol. 164, 2022, doi: 10.1016/j.patrec.2022.10.028.

[12] Y. Liu, M. Gao, H. Zong, X. Wang, and J. Li, "Real-Time Object Detection for the Running Train Based on the Improved YOLO V4 Neural Network," *J Adv Transp*, vol. 2022, 2022, doi: 10.1155/2022/4377953.

[13] Y. Peng, J. Peng, J. Li, P. Yan, and B. Hu, “Design and Development of the Fall Detection System based on Point Cloud,” in *Procedia Computer Science*, 2019. doi: 10.1016/j.procs.2019.01.253.
[14] X. Zi, K. Chaturvedi, A. Braytee, J. Li, and M. Prasad, “Detecting Human Falls in Poor Lighting: Object Detection and Tracking Approach for Indoor Safety,” *Electronics (Switzerland)*, vol. 12, no. 5, 2023, doi: 10.3390/electronics12051259.
[15] Y. Harari, N. Shawen, C. K. Mummidisetty, M. V. Albert, K. P. Kording, and A. Jayaraman, “A smartphone-based online system for fall detection with alert notifications and contextual information of real-life falls,” *J Neuroeng Rehabil*, vol. 18, no. 1, 2021, doi: 10.1186/s12984-021-00918-z.
[16] A. Amini, K. Banitsas, and W. R. Young, “Kinect4FOG: monitoring and improving mobility in people with Parkinson’s using a novel system incorporating the Microsoft Kinect v2,” *Disabil Rehabil Assist Technol*, vol. 14, no. 6, pp. 566–573, Aug. 2019, doi: 10.1080/17483107.2018.1467975.
[17] C. A. Q. Burgos, D. F. Q. Benavidez, E. R. Omen, and J. L. N. Semanate, “Fall detection system for people using video surveillance,” *Ingeniare*, vol. 28, no. 4, pp. 684–693, 2020, doi: 10.4067/S0718-33052020000400684.
[18] A. Amini, K. Banitsas, and W. R. Young, “Kinect4FOG: monitoring and improving mobility in people with Parkinson’s using a novel system incorporating the Microsoft Kinect v2,” *Disabil Rehabil Assist Technol*, vol. 14, no. 6, pp. 566–573, Aug. 2019, doi: 10.1080/17483107.2018.1467975.
[19] B. Pękala, T. Mroczek, D. Gil, and M. Kepski, “Application of Fuzzy and Rough Logic to Posture Recognition in Fall Detection System,” *Sensors*, vol. 22, no. 4, Feb. 2022, doi: 10.3390/s22041602.
[20] B. H. Wang, J. Yu, K. Wang, X. Y. Bao, and K. M. Mao, “Fall Detection Based on Dual-Channel Feature Integration,” *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.2999503.
[21] J. Kim, B. Kim, and H. Lee, “Fall Recognition Based on Time-Level Decision Fusion Classification,” *Applied Sciences*, vol. 14, no. 2, 2024, doi: 10.3390/app14020709.
[22] D. Ros and R. Dai, “A Flexible Fall Detection Framework Based on Object Detection and Motion Analysis,” in *5th International Conference on Artificial Intelligence in Information and Communication, ICAIIC 2023*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 63–68. doi: 10.1109/ICAIIC57133.2023.10066990.
[23] M. Bundele, H. Sharma, M. Gupta, and P. S. Sisodia, “An Elderly Fall Detection System using Depth Images,” in *2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering, ICRAIE 2020 - Proceeding*, 2020. doi: 10.1109/ICRAIE51050.2020.9358330.
[24] D. Yacchirema, J. S. De Puga, C. Palau, and M. Esteve, “Fall detection system for elderly people using IoT and Big Data,” in *Procedia Computer Science*, 2018. doi: 10.1016/j.procs.2018.04.110.
[25] K. L. Lu and E. T. H. Chu, “An image-based fall detection system for the elderly,” *Applied Sciences (Switzerland)*, vol. 8, no. 10, 2018, doi: 10.3390/app8101995.
[26] Z. Qiu, X. Liang, Q. Chen, X. Huang, and Y. Wang, “Old man fall detection based on surveillance video object tracking,” in *Communications in Computer and Information Science*, 2020. doi: 10.1007/978-981-15-2767-8_15.

[27] U. Masud, T. Saeed, H. M. Malaikah, F. U. Islam, and G. Abbas, “Smart Assistive System for Visually Impaired People Obstruction Avoidance Through Object Detection and Classification,” *IEEE Access*, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3146320.

[28] X. Zhang, J. Ji, L. Wang, Z. He, and S. Liu, “Image-based fall detection in bus compartment scene,” *IET Image Process*, vol. 17, no. 4, 2023, doi: 10.1049/ipr2.12705.

[29] M. Fayad *et al.*, “Fall Detection Approaches for Monitoring Elderly HealthCare Using Kinect Technology: A Survey,” Sep. 01, 2023, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/app131810352.

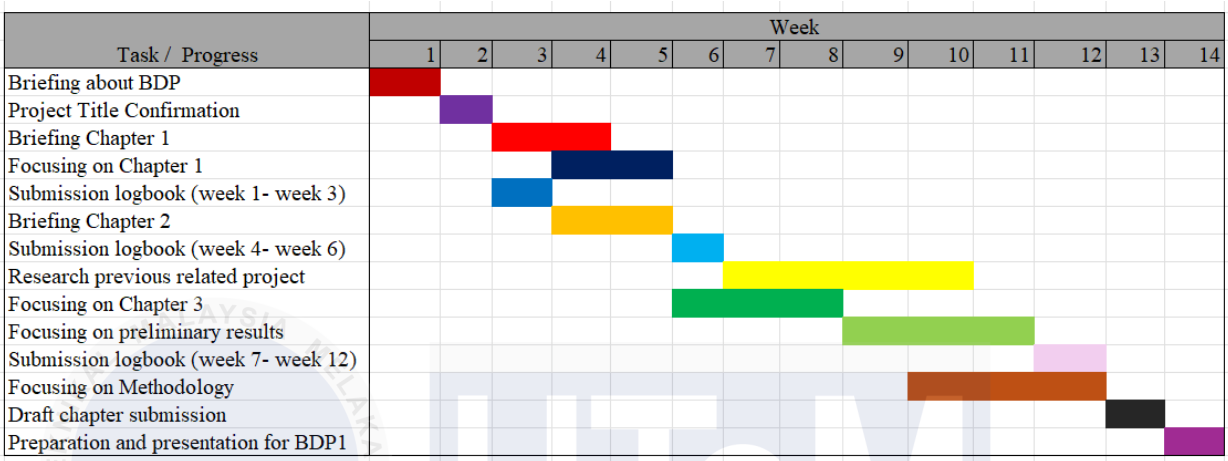
[30] X. Zhang, J. Ji, L. Wang, Z. He, and S. Liu, “Image-based fall detection in bus compartment scene,” *IET Image Process*, vol. 17, no. 4, 2023, doi: 10.1049/ipr2.12705.

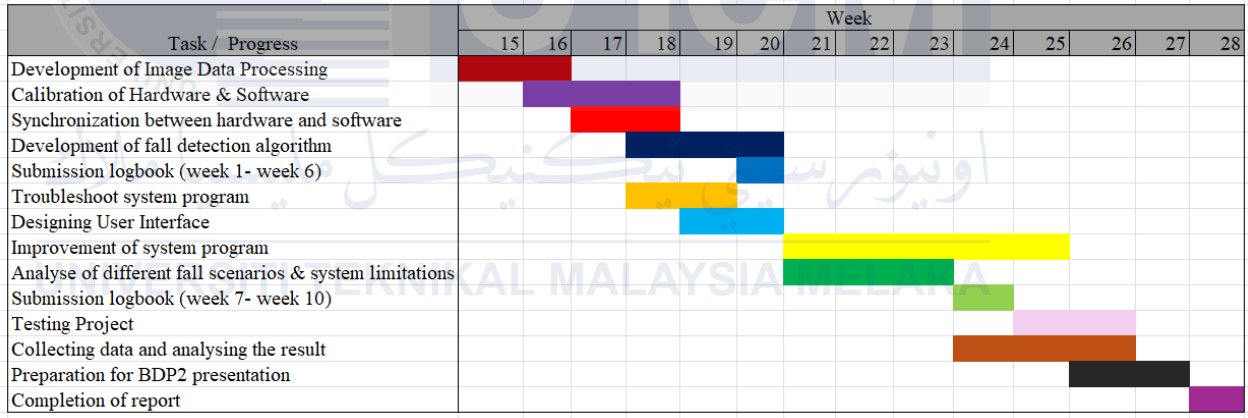
[31] Y. Harari, N. Shawen, C. K. Mummidisetty, M. V. Albert, K. P. Kording, and A. Jayaraman, “A smartphone-based online system for fall detection with alert notifications and contextual information of real-life falls,” *J Neuroeng Rehabil*, vol. 18, no. 1, 2021, doi: 10.1186/s12984-021-00918-z.

[32] J. Kim, B. Kim, and H. Lee, “Fall Recognition Based on Time-Level Decision Fusion Classification,” *Applied Sciences*, vol. 14, no. 2, 2024, doi: 10.3390/app14020709.

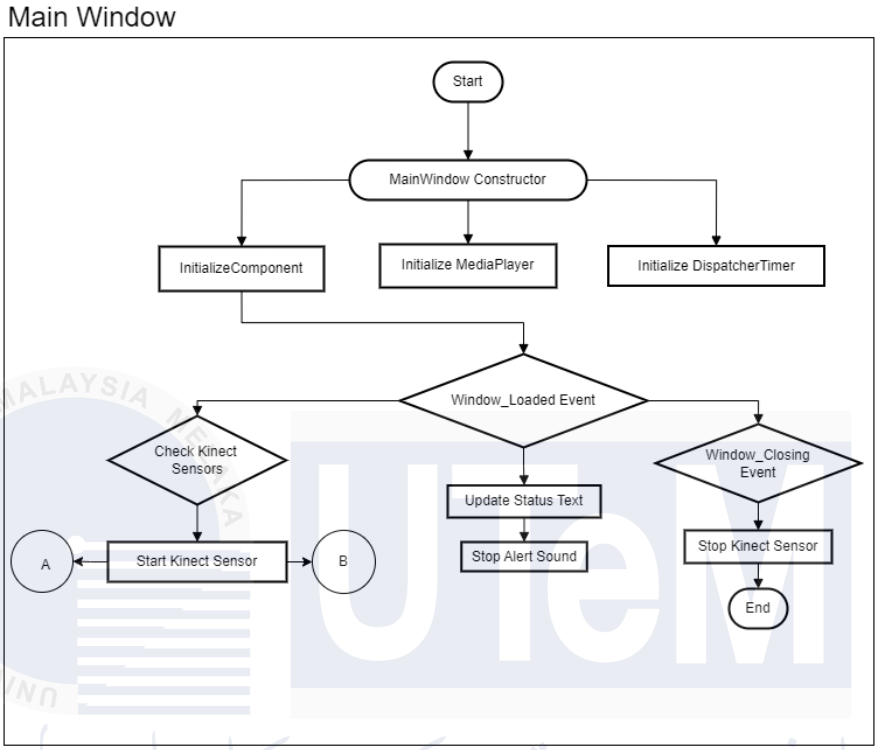
APPENDICES

Appendix A Gantt Chart PSM 1 & PSM 2

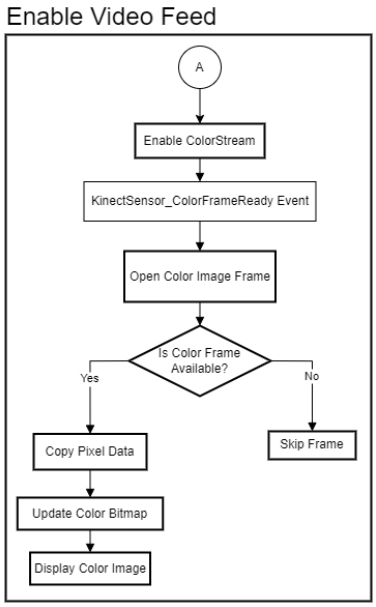




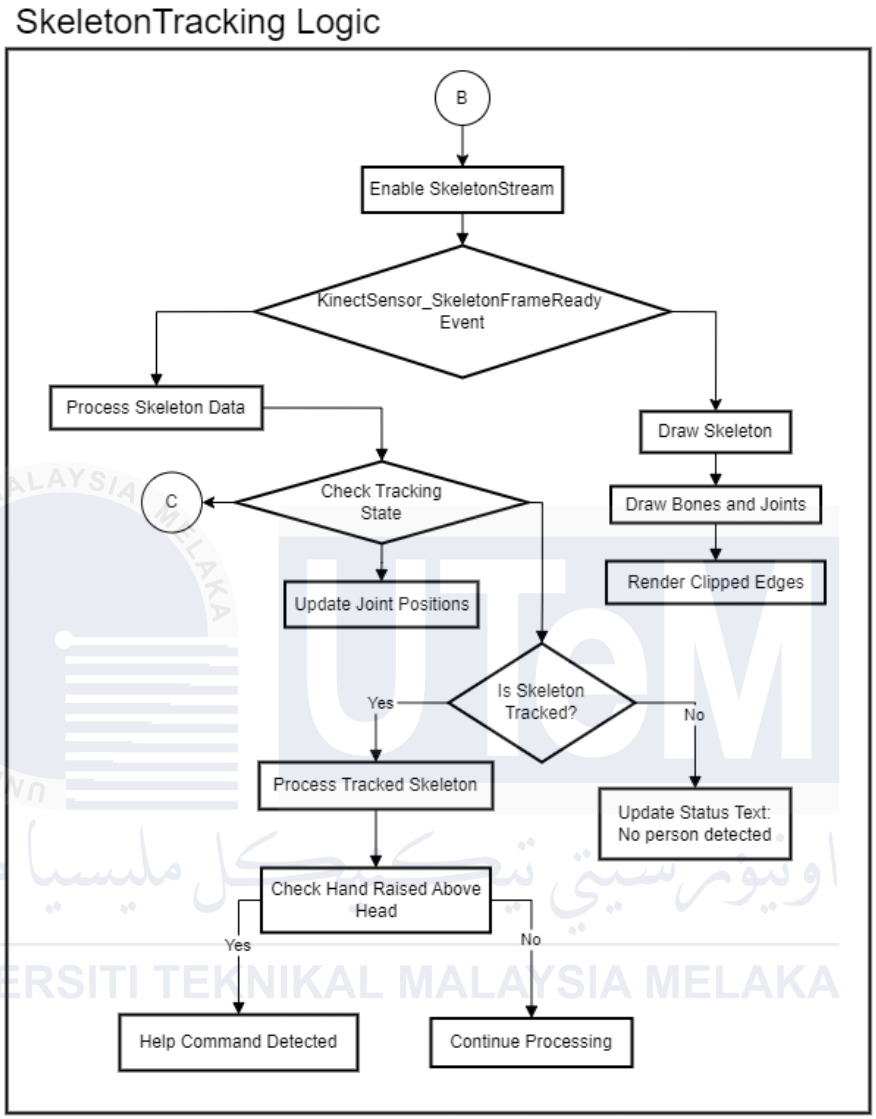
Appendix B Project Flowchart (1)



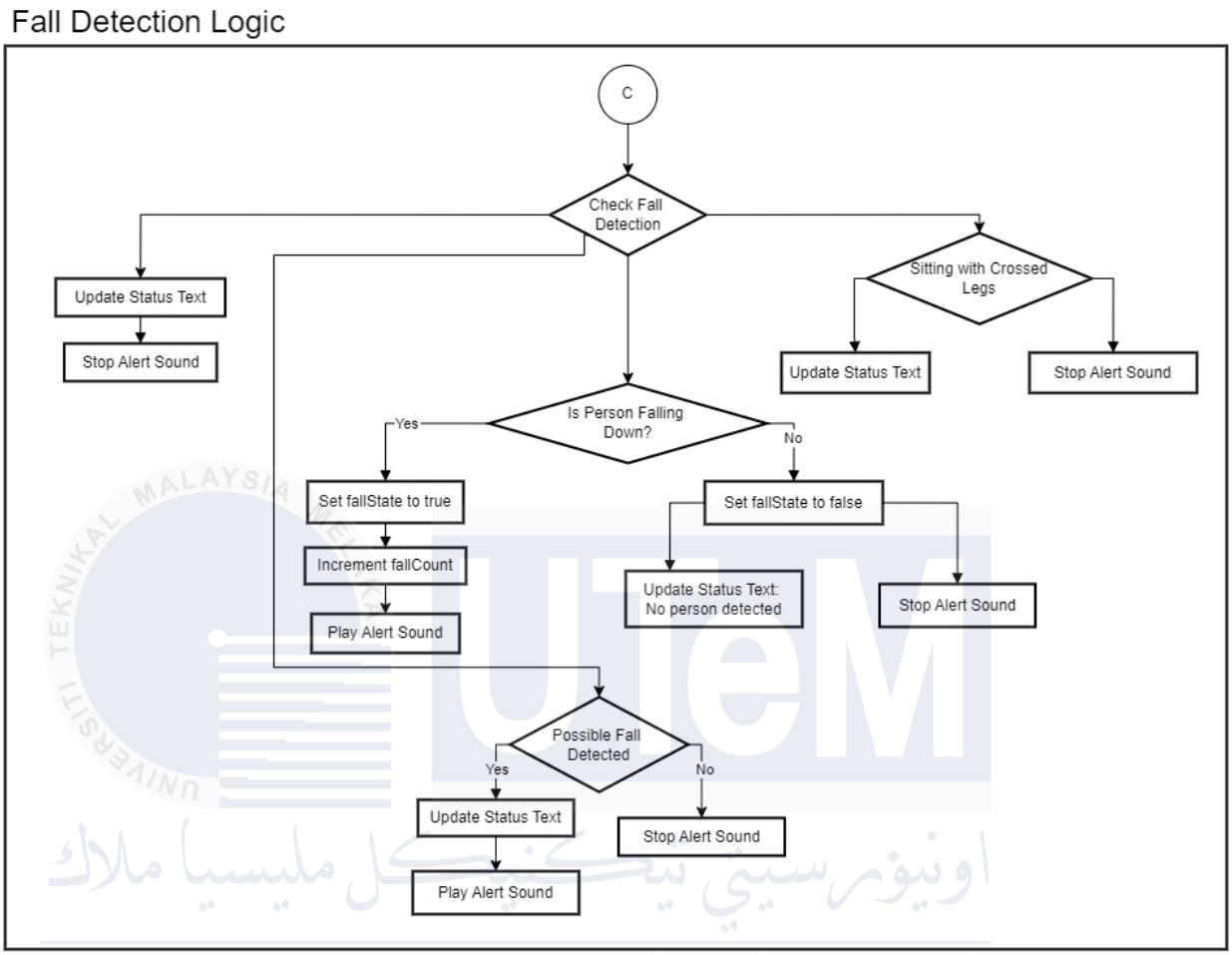
Appendix C Project Flowchart (2)



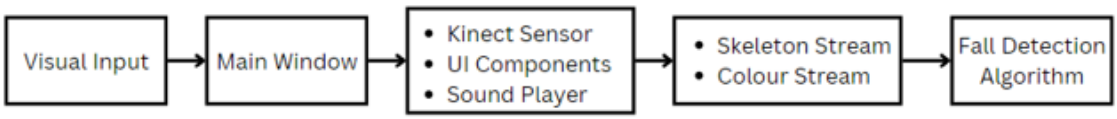
Appendix D Project Flowchart (3)



Appendix E Project Flowchart (4)



Appendix F Simple Block Diagram



Appendix G MainWindow.xaml.cs

```
using System;
using System.Globalization;
using System.Windows;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using Microsoft.Kinect;
using System.IO;
using System.Windows.Media.Media3D;
using System.Media;
using System.Threading.Tasks;
using System.Windows.Threading;

namespace FallDetection
{
   public partial class MainWindow : Window
   {
      private KinectSensor kinectSensor;
      private const float RenderWidth = 640.0f;
      private const float RenderHeight = 480.0f;
      private const double JointThickness = 3;
      private const double BodyCenterThickness = 10;
      private const double ClipBoundsThickness = 10;
      private readonly Brush centerPointBrush = Brushes.Blue;
      private readonly Brush trackedJointBrush = new SolidColorBrush(Color.FromArgb(255, 68, 192, 68));
      private readonly Brush inferredJointBrush = Brushes.Yellow;
      private readonly Pen trackedBonePen = new Pen(Brushes.Green, 6);
      private readonly Pen inferredBonePen = new Pen(Brushes.Gray, 1);
      private DrawingGroup drawingGroup;
      private DrawingImage imageSource;

      private WriteableBitmap colorBitmap;
```

```
// Check if the person is laying down (head and hip positions are low, ankles are at a higher level)
         bool isLaying = yHead < yHipCenter && yHipCenter < Math.Min(yAnkleLeft, yAnkleRight) - layingThreshold;

         // Check if knees are elevated above the floor
         bool kneesElevated = yKneeLeft > Math.Min(yAnkleLeft, yAnkleRight) + kneeElevatedThreshold &&
                              yKneeRight > Math.Min(yAnkleLeft, yAnkleRight) + kneeElevatedThreshold;

         return isLaying && kneesElevated;
      }

      private bool IsPersonFallingDown(Skeleton skeleton)
      {
         double yHead = skeleton.Joints[JointType.Head].Position.Y;
         double yShoulderCenter = skeleton.Joints[JointType.ShoulderCenter].Position.Y;
         double yHipCenter = skeleton.Joints[JointType.HipCenter].Position.Y;
         double yAnkleLeft = skeleton.Joints[JointType.AnkleLeft].Position.Y;
         double yAnkleRight = skeleton.Joints[JointType.AnkleRight].Position.Y;

         // Threshold factors
         double HeightThresholdFactor = 0.7;
         double distanceThreshold = 0;

         // Initialize head to ankle distance for the first time
         if (initialHeadToAnkleDistance == 0)
```

```
        private double initialHeadToAnkleDistance = 0;
        private const double positionThreshold = 0.3;
        private MediaPlayer fallAlertPlayer;
        private bool isSoundPlaying = false;
        private DispatcherTimer fallSoundTimer;
        private double lastHeadY = 0;
        private double lastShoulderCenterY = 0;
        private double lastHipCenterY = 0;
        private DateTime lastSkeletonTimestamp = DateTime.MinValue;
        private bool possibleFallDetected = false;
        private readonly TimeSpan fallDetectionTimeout = TimeSpan.FromSeconds(3); // Timeout for possible fall state
        private readonly Brush setAColor = Brushes.Green;
        private readonly Brush setBColor = Brushes.Blue;
        private bool fallState = false; // Persistent fall state flag

        public MainWindow()
        {
            InitializeComponent();

            // Initialize SoundPlayer with the path to the alert sound file
            fallAlertPlayer = new MediaPlayer();
            fallAlertPlayer.Open(new Uri(@"C:\Users\AMIAQ\Desktop\kinect_project\Police Siren Sound Effect.wav", UriKind.Absolute));

            // Initialize DispatcherTimer for looping sound during fall detection
            fallSoundTimer = new DispatcherTimer();
            fallSoundTimer.Interval = TimeSpan.FromSeconds(1);
            fallSoundTimer.Tick += (s, e) =>
            {
                initialHeadToAnkleDistance = yHead - Math.Min(yAnkleLeft, yAnkleRight); // Consider lower ankle position
                distanceThreshold = initialHeadToAnkleDistance * HeightThresholdFactor;
            }

            double currentHeadToAnkleDistance = yHead - Math.Min(yAnkleLeft, yAnkleRight);

            // Check if the person is standing or sitting with crossed legs
            if (IsPersonStanding(skeleton) || IsPersonSittingCrossedLegs(skeleton))
            {
                fallState = false; // Reset the fall state
                return false; // No fall detected
            }

            // Check if fall detected based on head position and body orientation
            bool isFall = Math.Abs(initialHeadToAnkleDistance - currentHeadToAnkleDistance) > distanceThreshold &&
                          yHead < positionThreshold &&
                          yShoulderCenter < positionThreshold &&
                          yHipCenter < positionThreshold;

            if (isFall)
            {
                // Additional check for limbs (arms and legs)
                if (IsLimbPositionAbnormal(skeleton))
                {
                    // Update last known position and timestamp
                    lastHeadY = yHead;
                    lastShoulderCenterY = yShoulderCenter;
                    lastHipCenterY = yHipCenter;
```

```
            fallAlertPlayer.Position =
TimeSpan.Zero;
            fallAlertPlayer.Play();
        };
    }


    // Play alert sound asynchronously to
prevent blocking
    private async Task
PlayAlertSoundAsync()
    {
        if (!isSoundPlaying)
        {
            fallAlertPlayer.Position =
TimeSpan.Zero;
            fallAlertPlayer.Play();
            isSoundPlaying = true;

            await Task.Delay(1000);
        }
    }

    // Stop alert sound and reset flag
    private void StopAlertSound()
    {
        if (isSoundPlaying)
        {
            fallAlertPlayer.Stop();
            isSoundPlaying = false;
        }
    }

    private void Window_Loaded(object
sender, RoutedEventArgs e)
    {
        drawingGroup = new
DrawingGroup();
        imageSource = new
DrawingImage(drawingGroup);
        SkeletonImage.Source =
imageSource;

        if
(KinectSensor.KinectSensors.Count > 0)
        {
            kinectSensor =
KinectSensor.KinectSensors[0];

            if (kinectSensor != null)
```

```
            lastSkeletonTimestamp =
DateTime.Now;

            fallState = true; // Confirm fall
            return true;
        }
    }

    // Update last known skeleton state
for possible fall detection
    lastHeadY = yHead;
    lastShoulderCenterY =
yShoulderCenter;
    lastHipCenterY = yHipCenter;
    lastSkeletonTimestamp =
DateTime.Now;

    return false; // Safe
}


private bool
IsLimbPositionAbnormal(Skeleton
skeleton)
    {
    // Retrieve limb joint positions
    double yElbowLeft =
skeleton.Joints[JointType.ElbowLeft].Posit
ion.Y;
    double yElbowRight =
skeleton.Joints[JointType.ElbowRight].Pos
ition.Y;
    double yKneeLeft =
skeleton.Joints[JointType.KneeLeft].Positi
on.Y;
    double yKneeRight =
skeleton.Joints[JointType.KneeRight].Posi
tion.Y;

    double yAnkleLeft =
skeleton.Joints[JointType.AnkleLeft].Posit
ion.Y;
    double yAnkleRight =
skeleton.Joints[JointType.AnkleRight].Pos
ition.Y;
```

```
            {
                kinectSensor.Start();


kinectSensor.ColorStream.Enable();

                colorBitmap = new
WriteableBitmap(kinectSensor.ColorStrea
m.FrameWidth,
kinectSensor.ColorStream.FrameHeight,
96.0, 96.0, PixelFormats.Bgr32, null);

                ColorImage.Source =
colorBitmap;


kinectSensor.ColorFrameReady +=
KinectSensor_ColorFrameReady;


kinectSensor.SkeletonStream.Enable();


kinectSensor.SkeletonFrameReady +=
KinectSensor_SkeletonFrameReady;

                StatusTextBlock.Text =
"Kinect sensor connected";
            }
            else
            {
                StatusTextBlock.Text = "No
Kinect sensor found";
            }
        }
        else
        {
            StatusTextBlock.Text = "No
Kinect sensor found";
        }
    }



    private void Window_Closing(object
sender,
System.ComponentModel.CancelEventArg
s e)
    {

        if (kinectSensor != null)
```

```
        // Check for abnormal limb
positions (e.g., limbs significantly lower
than the torso)
        if (yElbowLeft < positionThreshold
|| yElbowRight < positionThreshold ||
            yKneeLeft < positionThreshold ||
yKneeRight < positionThreshold ||
            yAnkleLeft < positionThreshold
|| yAnkleRight < positionThreshold)
        {
            return true; // Abnormal position
detected
        }

        return false;
    }


    private void
UpdateJointPositionsUI(double headY,
double shouldercenterY, double
hipcenterY, double ankleleftY, double
anklerightY)
    {
        HeadYText.Text = $"Head Y:
{headY:F2}";
        ShoulderCenterYText.Text =
$"ShoulderCenter Y:
{shouldercenterY:F2}";
        HipCenterYText.Text =
$"HipCenter Y: {hipcenterY:F2}";
        AnkleLeftYText.Text =
$"AnkleLeft Y: {ankleleftY:F2}";
        AnkleRightYText.Text =
$"AnkleRight Y: {anklerightY:F2}";
    }



    private bool
IsHandRaisedAboveHead(Skeleton
skeleton)
    {
        // Ensure both hands and head
joints are tracked before checking their
positions
        Joint head =
skeleton.Joints[JointType.Head];
```

```
        {
            kinectSensor.Stop();
            kinectSensor = null;
        }
    }


    private void
KinectSensor_ColorFrameReady(object
sender, ColorImageFrameReadyEventArgs
e)
    {
        using (ColorImageFrame
colorFrame = e.OpenColorImageFrame())
        {
            if (colorFrame != null)
            {
                byte[] colorData = new
byte[colorFrame.PixelDataLength];


colorFrame.CopyPixelDataTo(colorData);

                colorBitmap.WritePixels(new
Int32Rect(0, 0, colorFrame.Width,
colorFrame.Height), colorData,
colorFrame.Width *
colorFrame.BytesPerPixel, 0);
            }
        }
    }


    private async void
KinectSensor_SkeletonFrameReady(object
sender, SkeletonFrameReadyEventArgs e)
    {

        Skeleton[] skeletons = new
Skeleton[0];

        using (SkeletonFrame
skeletonFrame = e.OpenSkeletonFrame())
        {
            if (skeletonFrame != null)
            {
```

```
        Joint handLeft =
skeleton.Joints[JointType.HandLeft];
        Joint handRight =
skeleton.Joints[JointType.HandRight];

        if (head.TrackingState ==
JointTrackingState.Tracked &&
            ((handLeft.TrackingState ==
JointTrackingState.Tracked &&
handLeft.Position.Y > head.Position.Y) ||
             (handRight.TrackingState ==
JointTrackingState.Tracked &&
handRight.Position.Y > head.Position.Y)))
        {
            return true; // Hand is raised
above head
        }
        return false; // Hand is not above
head or joints not reliably tracked
    }


    private void UpdateStatusText(string
newText)
    {
        if (StatusTextBlock.Text !=
newText)
        {
            StatusTextBlock.Text =
newText;
        }
    }


    private void PlayFallAlertSound()
    {
        try
        {
            fallAlertPlayer.Play();  // Play
the sound asynchronously
        }
        catch (Exception ex)
        {
            StatusTextBlock.Text = "Error
playing alert sound: " + ex.Message;
        }
    }
```

```
            skeletons = new
Skeleton[skeletonFrame.SkeletonArrayLen
gth];

skeletonFrame.CopySkeletonDataTo(skele
tons);

            int fallCount = 0; // Track how
many people are falling
            int trackedCount = 0; // Track
how many people are being tracked
            bool helpCommandDetected =
false; // Track if help command is detected
            bool anySkeletonTracked =
false; // Track if any skeleton is tracked in
the frame

            foreach (Skeleton skeleton in
skeletons)
            {

                if (skeleton.TrackingState
== SkeletonTrackingState.Tracked)
                {
                    anySkeletonTracked =
true;
                    trackedCount++;

                    double headY =
skeleton.Joints[JointType.Head].Position.
Y;
                    double shouldercenterY
=
skeleton.Joints[JointType.ShoulderCenter].
Position.Y;
                    double hipcenterY =
skeleton.Joints[JointType.HipCenter].Posit
ion.Y;
                    double ankleleftY =
skeleton.Joints[JointType.AnkleLeft].Posit
ion.Y;
                    double anklerightY =
skeleton.Joints[JointType.AnkleRight].Pos
ition.Y;


UpdateJointPositionsUI(headY,
shouldercenterY, hipcenterY, ankleleftY,
anklerightY);
```

```
        private void
TestSoundButton_Click(object sender,
RoutedEventArgs e)
        {
            PlayFallAlertSound();  // Manually
trigger sound playing
        }

        private void
StopSoundButton_Click(object sender,
RoutedEventArgs e)
        {
            // Stop the sound
            if (fallAlertPlayer != null)
            {
                fallAlertPlayer.Stop();
                StatusTextBlock.Text = "Sound
stopped.";
            }
        }


        private void NotifyFallEvent()
        {
            StatusTextBlock.Text = "Help
Command Detected!! \fChecking for fall";

        }


        private void
DrawBonesAndJoints(Skeleton skeleton,
DrawingContext drawingContext)
        {
            // Render Torso
            DrawBone(skeleton,
drawingContext, JointType.Head,
JointType.ShoulderCenter);
            DrawBone(skeleton,
drawingContext,
JointType.ShoulderCenter,
JointType.Spine);
            DrawBone(skeleton,
drawingContext, JointType.Spine,
JointType.HipCenter);
            DrawBone(skeleton,
drawingContext,
JointType.ShoulderCenter,
JointType.ShoulderLeft);
```

```
                    if
(IsPersonFallingDown(skeleton))
                    {
                        fallState = true;
                        fallCount++;

                    }
                    else
                    {
                        fallState = false;
                        UpdateStatusText("No
person detected.");
                        StopAlertSound();
                    }

                    if
(IsHandRaisedAboveHead(skeleton))
                    {

helpCommandDetected = true;

UpdateStatusText("Help Command
Detected! Checking for fall.");
                        await
PlayAlertSoundAsync();
                    }
                }
            }
            if (helpCommandDetected)
            {
                UpdateStatusText("Help
Command Detected! Checking for fall.");
                await
PlayAlertSoundAsync();
            }

            else if (fallCount == 0 &&
trackedCount == 1)
            {
                UpdateStatusText("Fall is
not detected.");
                StopAlertSound();
            }

            else if (fallCount == 1 &&
trackedCount == 1)
            {
```

```
        DrawBone(skeleton,
drawingContext,
JointType.ShoulderCenter,
JointType.ShoulderRight);
        DrawBone(skeleton,
drawingContext, JointType.HipCenter,
JointType.HipLeft);
        DrawBone(skeleton,
drawingContext, JointType.HipCenter,
JointType.HipRight);

        // Left Arm
        DrawBone(skeleton,
drawingContext, JointType.ShoulderLeft,
JointType.ElbowLeft);
        DrawBone(skeleton,
drawingContext, JointType.ElbowLeft,
JointType.WristLeft);
        DrawBone(skeleton,
drawingContext, JointType.WristLeft,
JointType.HandLeft);

        // Right Arm
        DrawBone(skeleton,
drawingContext, JointType.ShoulderRight,
JointType.ElbowRight);
        DrawBone(skeleton,
drawingContext, JointType.ElbowRight,
JointType.WristRight);
        DrawBone(skeleton,
drawingContext, JointType.WristRight,
JointType.HandRight);

        // Left Leg
        DrawBone(skeleton,
drawingContext, JointType.HipLeft,
JointType.KneeLeft);
        DrawBone(skeleton,
drawingContext, JointType.KneeLeft,
JointType.AnkleLeft);
        DrawBone(skeleton,
drawingContext, JointType.AnkleLeft,
JointType.FootLeft);

        // Right Leg
        DrawBone(skeleton,
drawingContext, JointType.HipRight,
JointType.KneeRight);
```

```
                UpdateStatusText(" Fall
detected.");
                await
PlayAlertSoundAsync();
                possibleFallDetected = true;
// Set possible fall state
                lastSkeletonTimestamp =
DateTime.Now; // Update timestamp
            }

            else if (fallCount == 1 &&
trackedCount > 1)
            {

                UpdateStatusText("One fall
detected with multiple people in frame.");
                await
PlayAlertSoundAsync();
            }

            else if (fallCount > 1)
            {
                UpdateStatusText("Multiple
falls detected.");
                await
PlayAlertSoundAsync(); // Asynchronous
sound playback
            }

            else if (!anySkeletonTracked)
            {
                if (fallState)
                {

UpdateStatusText("Possible fall detected
(skeleton lost).");
                    await
PlayAlertSoundAsync();
                }
                else
                {
                    UpdateStatusText("No
person detected.");
                    StopAlertSound();
                    fallState = false;
                }
            }
```

```
        DrawBone(skeleton,
drawingContext, JointType.KneeRight,
JointType.AnkleRight);
        DrawBone(skeleton,
drawingContext, JointType.AnkleRight,
JointType.FootRight);



        foreach (Joint joint in
skeleton.Joints)
        {
            if (joint.TrackingState ==
JointTrackingState.Tracked)
            {
                Brush drawBrush = null;

                if (joint.TrackingState ==
JointTrackingState.Tracked)
                {
                    drawBrush =
trackedJointBrush;
                }
                else if (joint.TrackingState ==
JointTrackingState.Inferred)
                {
                    drawBrush =
inferredJointBrush;
                }
                if (drawBrush != null)
                {

drawingContext.DrawEllipse(drawBrush,
null,
SkeletonPointToScreen(joint.Position),
JointThickness, JointThickness);
                }
            }
        }
    }



    private Point
SkeletonPointToScreen(SkeletonPoint
skelpoint)
    {

        DepthImagePoint depthPoint =
kinectSensor.CoordinateMapper.MapSkele
tonPointToDepthPoint(skelpoint,
```

```
            else if (trackedCount > 1 && fallCount == 0)
            {
                UpdateStatusText("More than one person detected");
                StopAlertSound();
            }

            else
            {
                UpdateStatusText("Fall is not detected.");
                StopAlertSound();
            }
        }
    }


    using (DrawingContext dc = drawingGroup.Open())
    {

dc.DrawRectangle(Brushes.Transparent, null, new Rect(0.0, 0.0, RenderWidth, RenderHeight));

        if (skeletons.Length != 0)
        {
            foreach (Skeleton skel in skeletons)
            {

                RenderClippedEdges(skel, dc, colorBitmap);

                if (skel.TrackingState == SkeletonTrackingState.Tracked)
                {

DrawBonesAndJoints(skel, dc);
                }
                else if (skel.TrackingState == SkeletonTrackingState.PositionOnly)
                {
                    dc.DrawEllipse(
                    centerPointBrush,
                    null,

SkeletonPointToScreen(skel.Position),
                    BodyCenterThickness,
```

```
DepthImageFormat.Resolution640x480Fps30);
        return new Point(depthPoint.X, depthPoint.Y);
    }


    private void DrawBone(Skeleton skeleton, DrawingContext drawingContext, JointType jointType0, JointType jointType1)
    {
        Joint joint0 = skeleton.Joints[jointType0];
        Joint joint1 = skeleton.Joints[jointType1];

        // If can't find either of these joints, exit
        if (joint0.TrackingState == JointTrackingState.NotTracked ||
            joint1.TrackingState == JointTrackingState.NotTracked)
        {
            return;
        }

        // Assume all drawn bones are inferred unless BOTH joints are tracked
        if (joint0.TrackingState == JointTrackingState.Inferred &&
            joint1.TrackingState == JointTrackingState.Inferred)
        {
            return;
        }

        Pen drawPen = this.inferredBonePen;
        if (joint0.TrackingState == JointTrackingState.Tracked && joint1.TrackingState == JointTrackingState.Tracked)
        {
            drawPen = trackedBonePen;
        }


drawingContext.DrawLine(drawPen,
```

```
                    BodyCenterThickness);
                }
            }
        }

        drawingGroup.ClipGeometry =
new RectangleGeometry(new Rect(0.0,
0.0, RenderWidth, RenderHeight));

    }
}



private bool
IsPersonStanding(Skeleton skeleton)
{
    double yHead =
skeleton.Joints[JointType.Head].Position.
Y;
    double yShoulderCenter =
skeleton.Joints[JointType.ShoulderCenter].
Position.Y;
    double yHipCenter =
skeleton.Joints[JointType.HipCenter].Posit
ion.Y;
    double yAnkleLeft =
skeleton.Joints[JointType.AnkleLeft].Posit
ion.Y;
    double yAnkleRight =
skeleton.Joints[JointType.AnkleRight].Pos
ition.Y;

    double
currentHeadToAnkleDistance = yHead -
Math.Min(yAnkleLeft, yAnkleRight);

    // Detect if the person is standing
upright
    return
currentHeadToAnkleDistance >
initialHeadToAnkleDistance * 0.9 &&
            yHead > positionThreshold
&&
            yShoulderCenter >
positionThreshold &&
            yHipCenter >
positionThreshold;
}
```

```
SkeletonPointToScreen(joint0.Position),
SkeletonPointToScreen(joint1.Position));
    }



    private static void
RenderClippedEdges(Skeleton skeleton,
DrawingContext drawingContext,
WriteableBitmap colorBitmap)
    {
        double actualRenderWidth =
colorBitmap.PixelWidth;



        if
(skeleton.ClippedEdges.HasFlag(FrameEd
ges.Bottom))
        {
            drawingContext.DrawRectangle(
                Brushes.Red,
                null,
                new Rect(0, RenderHeight -
ClipBoundsThickness, actualRenderWidth,
ClipBoundsThickness));
        }

        if
(skeleton.ClippedEdges.HasFlag(FrameEd
ges.Top))
        {
            drawingContext.DrawRectangle(
                Brushes.Red,
                null,
                new Rect(0, 0,
actualRenderWidth,
ClipBoundsThickness));
        }

        if
(skeleton.ClippedEdges.HasFlag(FrameEd
ges.Left))
        {
            drawingContext.DrawRectangle(
                Brushes.Red,
                null,
                new Rect(0, 0,
ClipBoundsThickness, RenderHeight));
        }
```

```
        private bool IsPersonSittingCrossedLegs(Skeleton skeleton)
        {
            double yHipCenter = skeleton.Joints[JointType.HipCenter].Position.Y;
            double yKneeLeft = skeleton.Joints[JointType.KneeLeft].Position.Y;
            double yKneeRight = skeleton.Joints[JointType.KneeRight].Position.Y;
            double yAnkleLeft = skeleton.Joints[JointType.AnkleLeft].Position.Y;
            double yAnkleRight = skeleton.Joints[JointType.AnkleRight].Position.Y;

            double crossingThreshold = 0.1; // Threshold for crossed-leg sitting (adjust as needed)

            // Check if knees and ankles are at similar heights and hips are above them
            return Math.Abs(yAnkleLeft - yAnkleRight) < crossingThreshold &&
                   Math.Abs(yKneeLeft - yKneeRight) < crossingThreshold &&
                   yHipCenter > Math.Max(yKneeLeft, yKneeRight);
        }

        private bool IsLayingWithKneesUp(Skeleton skeleton)
        {
            double yHead = skeleton.Joints[JointType.Head].Position.Y;
            double yHipCenter = skeleton.Joints[JointType.HipCenter].Position.Y;
            double yKneeLeft = skeleton.Joints[JointType.KneeLeft].Position.Y;
            double yKneeRight = skeleton.Joints[JointType.KneeRight].Position.Y;
            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Right))
            {
                drawingContext.DrawRectangle(
                    Brushes.Red,
                    null,
                    new Rect(actualRenderWidth - ClipBoundsThickness, 0, ClipBoundsThickness, RenderHeight));
            }
        }


        private void Button_Click(object sender, RoutedEventArgs e)
        {
            RenderTargetBitmap renderTargetBitmap = new RenderTargetBitmap((int)this.ActualWidth, (int)this.ActualHeight, 96d, 96d, PixelFormats.Pbgra32);
            renderTargetBitmap.Render(this);

            BitmapEncoder encoder = new PngBitmapEncoder();

            encoder.Frames.Add(BitmapFrame.Create(renderTargetBitmap));

            string time = DateTime.Now.ToString("hh'-'mm'-'ss", CultureInfo.CurrentUICulture.DateTimeFormat);
            string myPhotos = Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);
            string path = Path.Combine(myPhotos, "KinectWindowSnapshot-" + time + ".png");

            try
            {

                Directory.CreateDirectory(Path.GetDirectoryName(path));
                using (FileStream fs = new FileStream(path, FileMode.Create))
                {
```

```
            double yAnkleLeft =
skeleton.Joints[JointType.AnkleLeft].Posit
ion.Y;
            double yAnkleRight =
skeleton.Joints[JointType.AnkleRight].Pos
ition.Y;

            double layingThreshold = 0.5; //
Threshold for laying down (adjust as
needed)
            double kneeElevatedThreshold =
0.2; // Threshold for knees not being on the
floor
```

```
                encoder.Save(fs);
            }
            StatusTextBlock.Text =
"Screenshot saved.";
        }
        catch (IOException)
        {
            StatusTextBlock.Text = "Error
saving screenshot.";
        }
    }


    private void
ApplyTiltButton_Click(object sender,
RoutedEventArgs e)
    {
        int selectedAngle =
(int)TiltAngleSlider.Value;

AdjustKinectTiltAngle(selectedAngle);
    }


    private void
AdjustKinectTiltAngle(int angle)
    {
        if (kinectSensor != null &&
kinectSensor.IsRunning)
        {
            angle = Math.Max(-27,
Math.Min(27, angle));

            try
            {
                kinectSensor.ElevationAngle
= angle;
                StatusTextBlock.Text = $"Tilt
angle set to {angle} degrees.";
            }
            catch
(InvalidOperationException)
            {
                StatusTextBlock.Text = "Error
adjusting tilt angle.";
            }
        }
    }
  }
}
```

Appendix H MainWindow.xaml

```
<Window x:Class="FallDetection.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Fall Detection App" Height="600" Width="1000"
        Loaded="Window_Loaded" Closing="Window_Closing" >


    <Grid>
        <Grid>

            <Image x:Name="ColorImage" HorizontalAlignment="Stretch"
VerticalAlignment="Stretch" Margin="113,10,285.2,52" />

            <Image x:Name="SkeletonImage" HorizontalAlignment="Stretch"
VerticalAlignment="Stretch" Margin="112,10,285.2,52" />

            <TextBlock x:Name="StatusTextBlock" HorizontalAlignment="Left"
Margin="734,255,0,0" TextWrapping="Wrap" Text="Status" VerticalAlignment="Top"
FontSize="24" FontWeight="Bold" RenderTransformOrigin="1.12,-0.242" />

            <TextBlock x:Name="HeadYText" HorizontalAlignment="Left"
TextWrapping="Wrap" VerticalAlignment="Top" Text="Head" Margin="734,59,0,0"
FontSize="16" RenderTransformOrigin="0.6,-0.612"/>

            <TextBlock x:Name="ShoulderCenterYText" HorizontalAlignment="Left"
Margin="734,94,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
Text="Shoulder" FontSize="16" RenderTransformOrigin="-0.165,-1.862"/>

            <TextBlock x:Name="HipCenterYText" HorizontalAlignment="Left"
Margin="734,129,0,0" TextWrapping="Wrap"  VerticalAlignment="Top" Text="Hip"
FontSize="16" RenderTransformOrigin="0.522,-1.712"/>

            <TextBlock x:Name="AnkleLeftYText" HorizontalAlignment="Left"
TextWrapping="Wrap" VerticalAlignment="Top" Text="AnkleLeft"
Margin="734,165,0,0" FontSize="16" RenderTransformOrigin="0.502,-0.677"/>

            <TextBlock x:Name="AnkleRightYText" HorizontalAlignment="Left"
Margin="734,201,0,0" TextWrapping="Wrap" Text="AnkleRight"
VerticalAlignment="Top" FontSize="16" RenderTransformOrigin="-0.139,0.727"/>

            <Button x:Name="ScreenshotBtn" Content="Screenshot"
HorizontalAlignment="Left" Margin="845,457,0,0" VerticalAlignment="Top"
Width="85" Click="Button_Click" Height="26" FontSize="14"
RenderTransformOrigin="1.308,-0.321"/>
```

```
        <Slider x:Name="TiltAngleSlider" Minimum="-27" Maximum="27"
TickFrequency="1" SmallChange="1" LargeChange="5" Value="0"
Margin="734,375,54.6,160.4" />

        <TextBlock x:Name="TiltAngleValueText" Text="{Binding
ElementName=TiltAngleSlider, Path=Value, StringFormat='Tilt Angle: {0:F0}°'}"
HorizontalAlignment="Left" Margin="798,410,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" FontSize="14" RenderTransformOrigin="0.367,0.476"/>

        <Button x:Name="ApplyTiltButton" Content="Apply Tilt"
HorizontalAlignment="Left" Click="ApplyTiltButton_Click" VerticalAlignment="Top"
Margin="734,457,0,0" Height="26" Width="85" FontSize="14"
RenderTransformOrigin="3.968,-3.336"/>

        <Button x:Name="TestSoundButton" Content="Test Sound"
Click="TestSoundButton_Click" Margin="734,509,173,29" />

        <Button x:Name="StopSoundButton" Content="Stop Sound"
Click="StopSoundButton_Click" Margin="845,508,62,27"/>


    </Grid>
  </Grid>
</Window>
```